

# Appendix C

## User Guide

### C.1 Common Functions

- click Home button to refresh the page
- click Tutorial button to open the github repository of the web page in a new tab, and the tutorial is demonstrated in the README file
- click Run in Swish to open SWISH in a new tab

### C.2 Start Programming on Workflow Graph

In this section, we will illustrate with an example to demonstrate what to do if we want to express with the workflow interface what will happen after a person feels hungry.

#### C.2.1 Initialize

First of all, we would like to show the observations and initial states: bob feels hungry from time 3 to time 4, and he intends to buy some food to eat. Then we find that available food includes burger, whose price is 10 pounds. Bob does not have any cash with him, but his account balance is 100 pounds.

The above facts can be expressed with a Initialization Graph. The steps are as follows:

1. hover over Graph button and click on Initialize to generate the graph paper

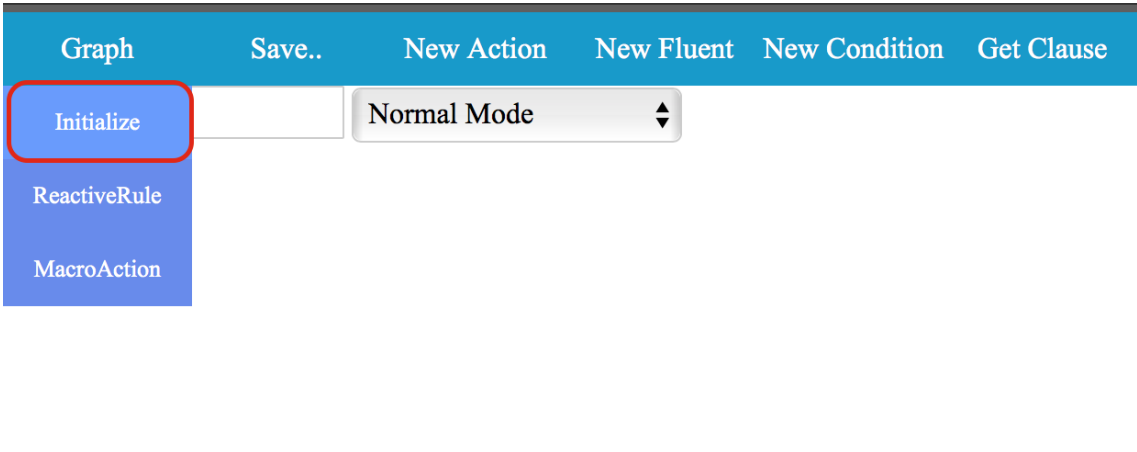


Figure C.1: Initialize

2. add all observations: click on New Action button, a green box for action will be generated. Select Text Mode and insert `feel_hungry(bob)` to the input area and right click the green box to insert text.

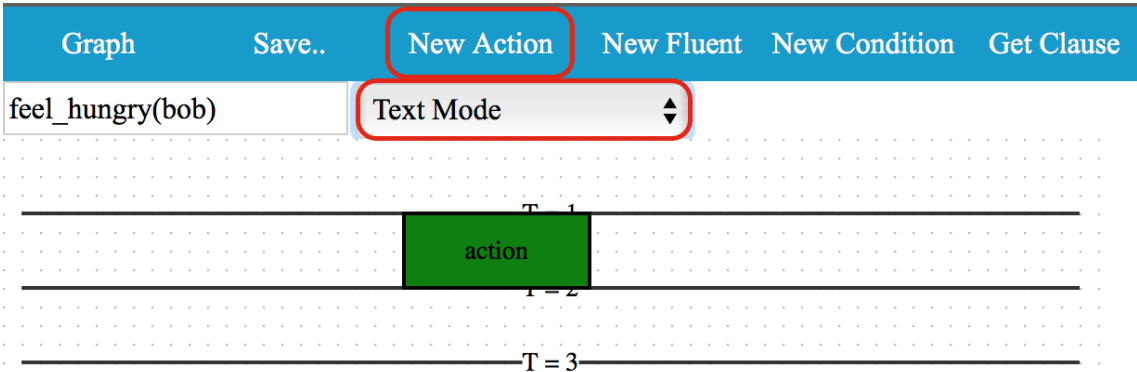


Figure C.2: Observation1

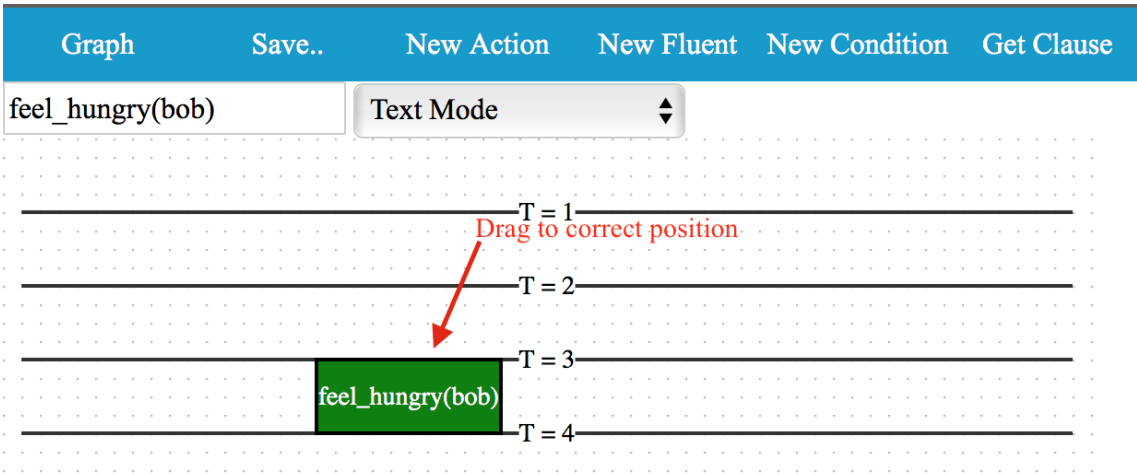


Figure C.3: Observation2

Drag the box to a position between  $T = 3$  and  $T = 4$ . Right now the below graph means that bob feels hungry from time 3 to time 4.

3. add all initial states: click New Fluent button twice to generate two blue rectangles for fluent. Type in `balance(bob, 100)`, `cash(bob, 0)` respectively and right click those two blue rectangles to insert the text.

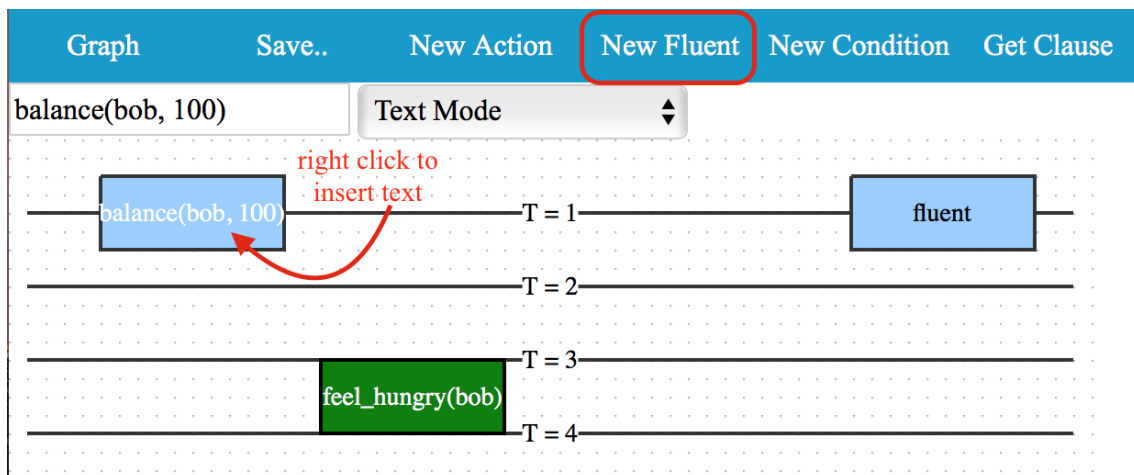


Figure C.4: Initial states

The two blue rectangles express that Bob does not have any cash with him, but his account balance is 100 pounds.

4. add all timeless facts: click on New Condition twice and insert the text in the same way to illustrate that available food includes burger, whose price is 10 pounds.

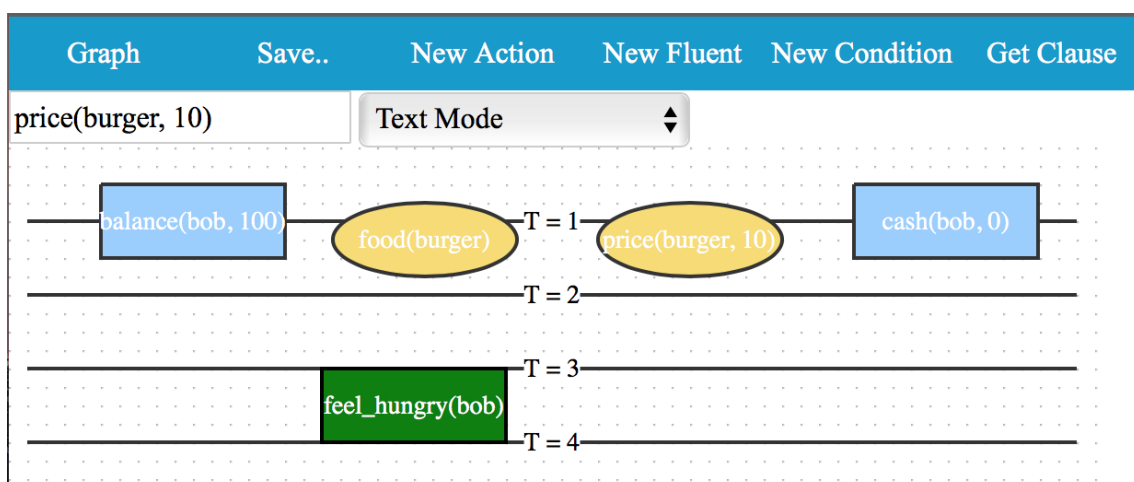


Figure C.5: Timeless Facts

5. hover over Save.. button and click on save

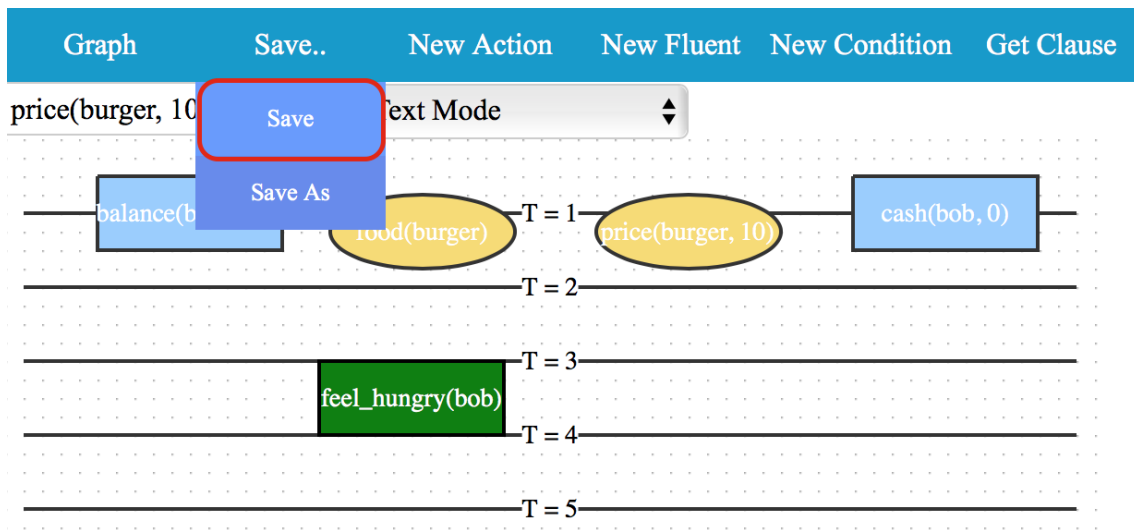


Figure C.6: Save1

- input a random name for the initialization graph and save

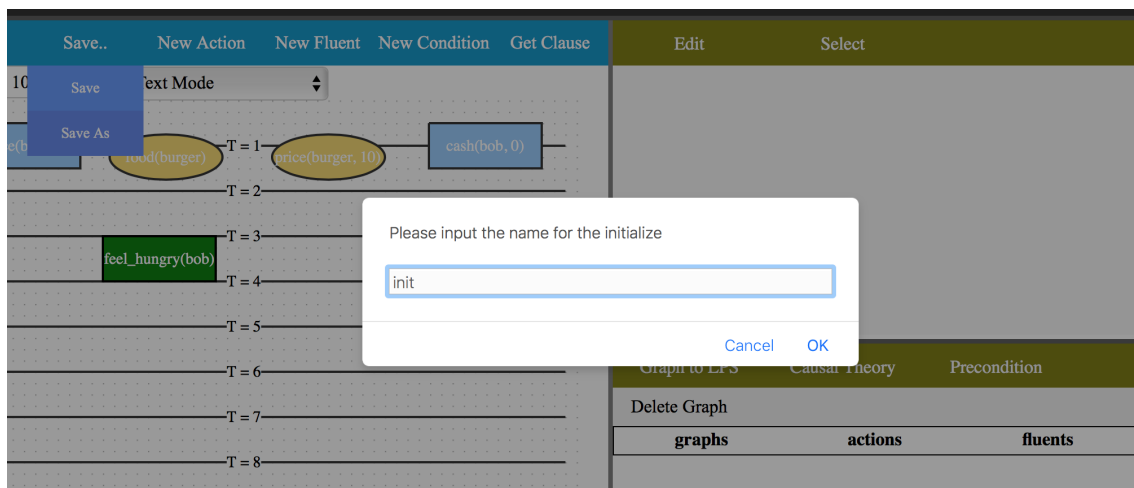


Figure C.7: Save2

- click Get Clause button to get LPS code for the graph. The LPS code will be demonstrate in the code area. The button in the clause panel will store all the information about the graph.

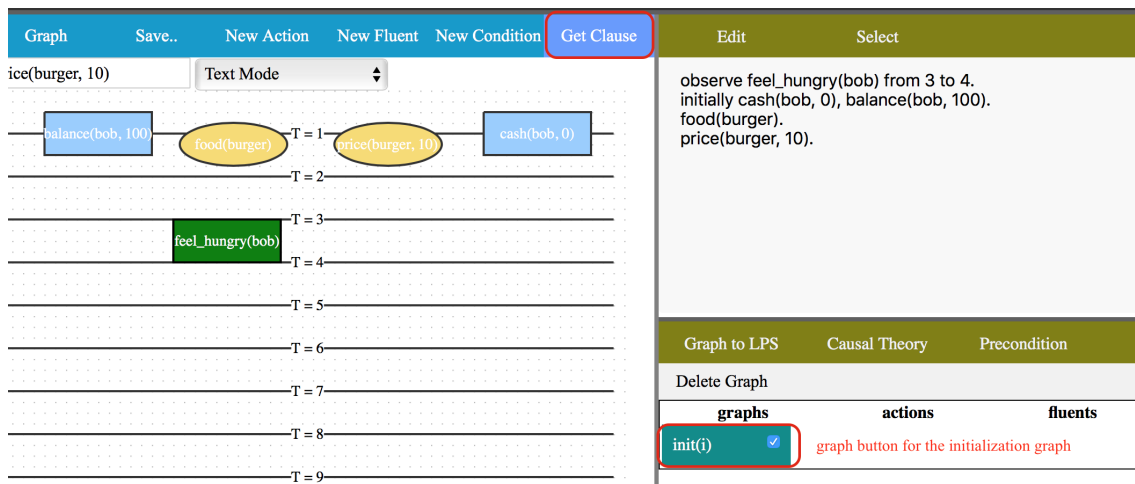


Figure C.8: Get Clause

### C.2.2 Reactive Rule

After setting the initial states, we would like to move on to state some rules: if a person feels hungry, he or she will try to buy some food. If a person pays for food, and he or she is hungry, he or she will eat the food immediately. Each rule will be expressed by a Reactive Rule Graph. The steps are as follows:

1. hover over Graph button and click on ReactiveRule

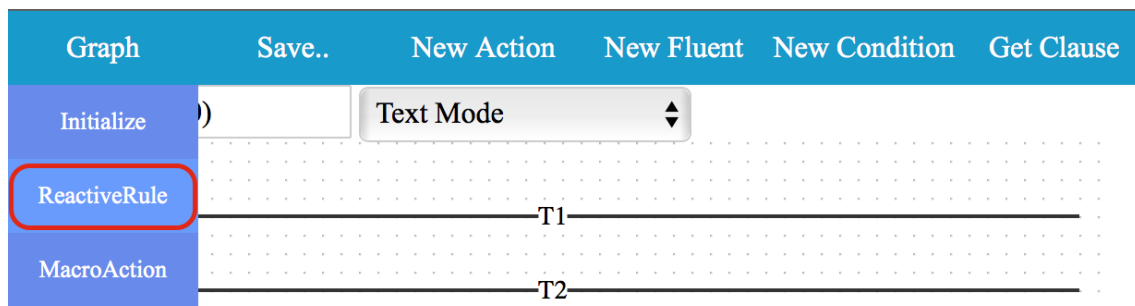


Figure C.9: Reactive Rule

2. add all antecedent: feel\_hungry(P) here means variable P (a person) feels hungry.

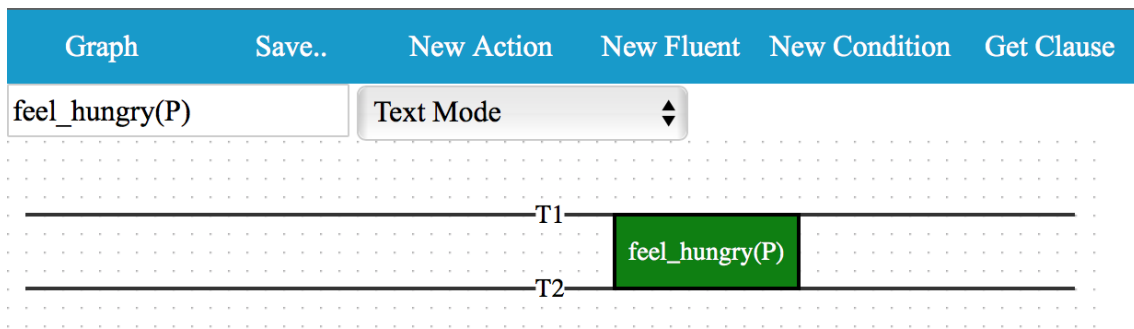


Figure C.10: Antecedent

3. add all consequent: after creating element and inserting text, select Conclusion Mode and right click to set the cell to be consequent. If we want to reset a consequent to be an antecedent, right click the red font colour again to make it white.

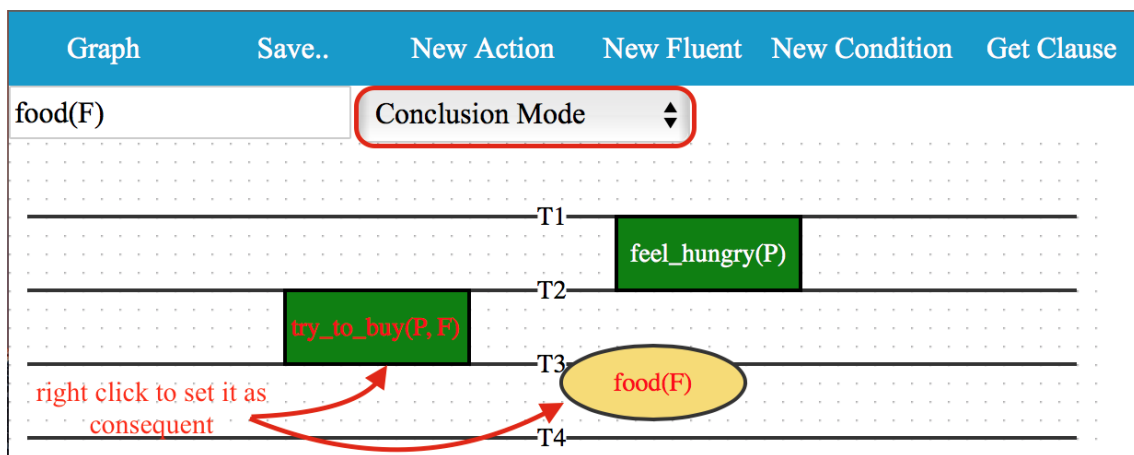


Figure C.11: Consequent

4. hover over Save.. button and click on save
5. input a random name for the reactive rule graph and save. This graph means that if a person feels hungry, he or she will try to buy some food. Please note that a Reactive Rule graph must include at least one antecedent and at least one consequent, and all antecedents should appear earlier (higher in position in graph) than all consequents. The sequence of each cell determines the sequence of predicates in the LPS clause. For instance, `try_to_buy(P, F)` appears before `food(F)` in the LPS clause only because `try_to_buy(P, F)` is higher in position in graph than `food(F)`.

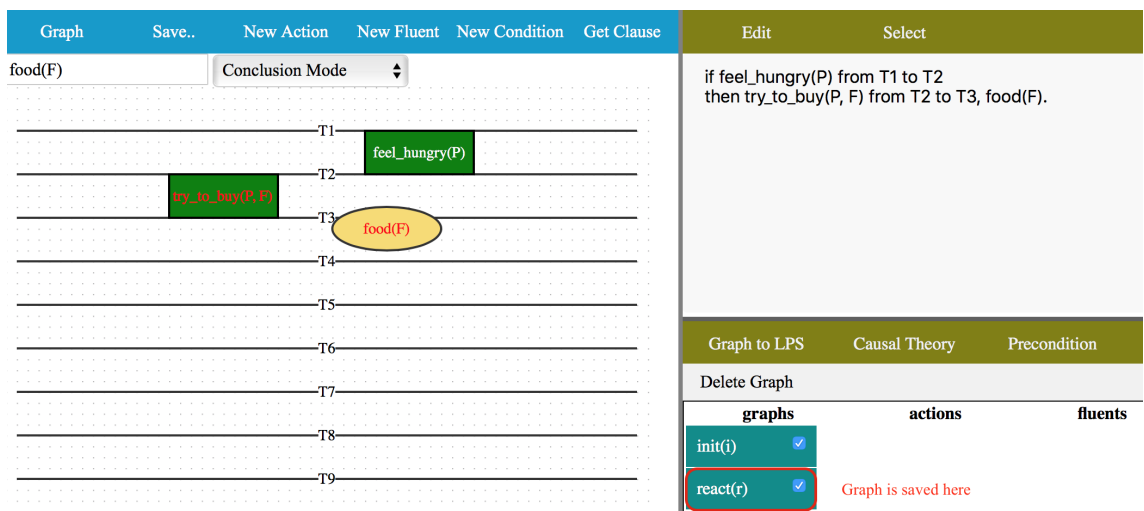


Figure C.12: Save3

6. repeat step 1-5 above to generate more reactive rules.

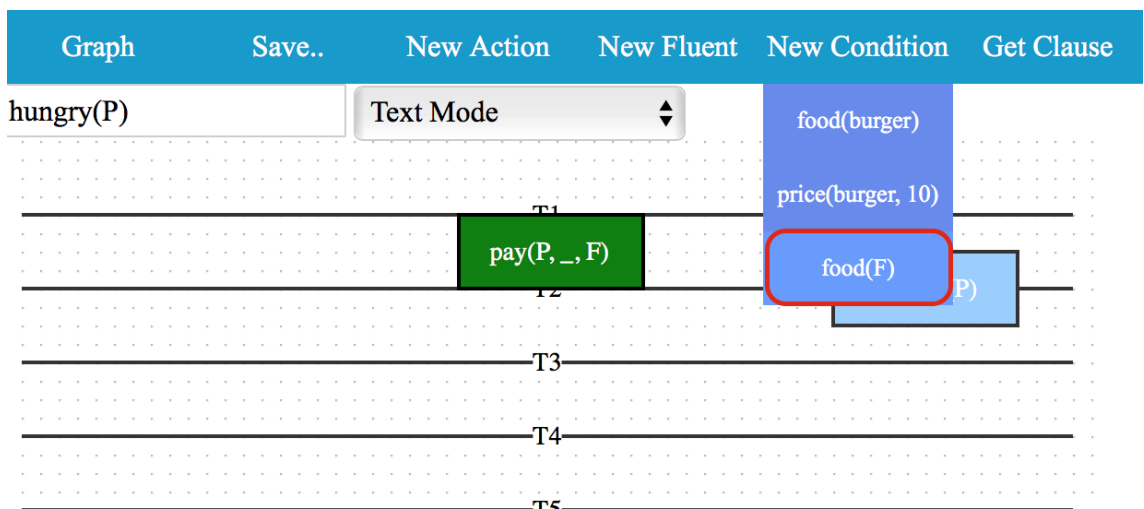


Figure C.13: Quick Generation of Cells

Since we have input before a condition that variable F belongs to food, we can hover over New Condition and click the "Food(F)" directly to generate the condition already with the same text. This is the same with New Action and New Fluent.

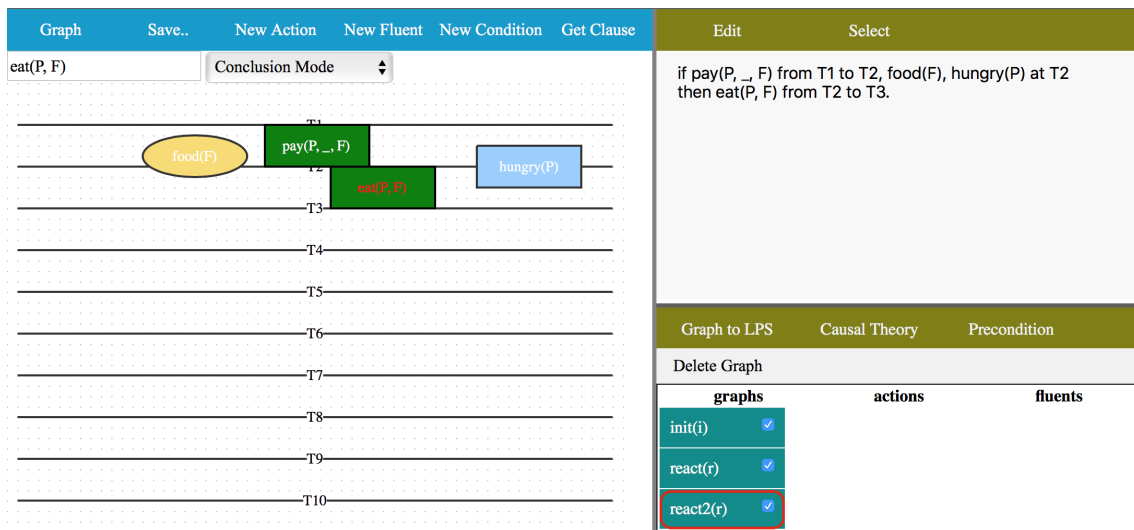


Figure C.14: Save4

The above graph illustrates that if a person pays for food, and he or she is hungry, he or she will eat the food immediately.

### C.2.3 Macro Action

Some of the actions might consist of a series of actions, fluents, and conditions, and those are called macro actions. For instance, the `try_to_buy(P, F)` indicating the a person `P` tries to buy something `F` is actually a macro action. It includes the following three situations:

- the food `F` is of price `Pr`, and the person `P` pay with cash
- the food `F` is of price `Pr`, and the person `P` has to withdraw some money `M` to pay the bill
- the food `F` is of price `Pr`, and `Pr` is greater than person `P`'s cash plus deposit, so `P` does nothing.

The steps are as follows:

1. hover over Graph button and click on MacroAction

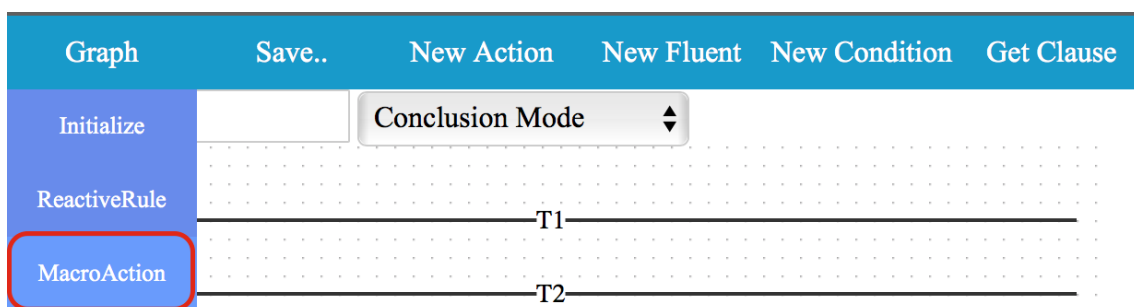
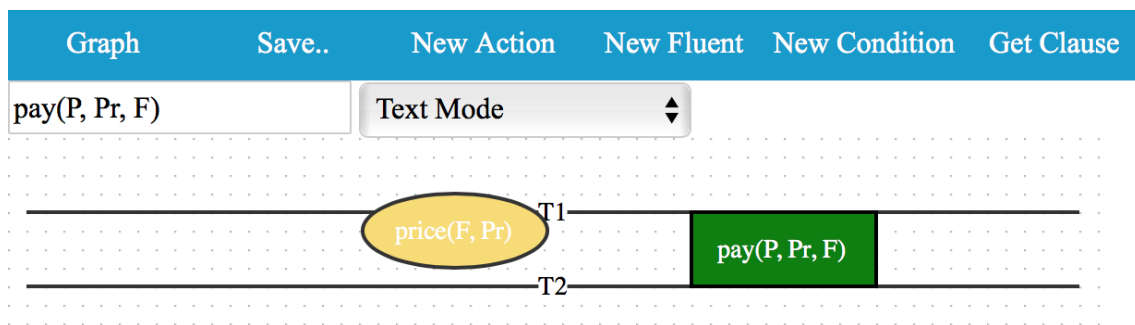


Figure C.15: Macro Action Graph

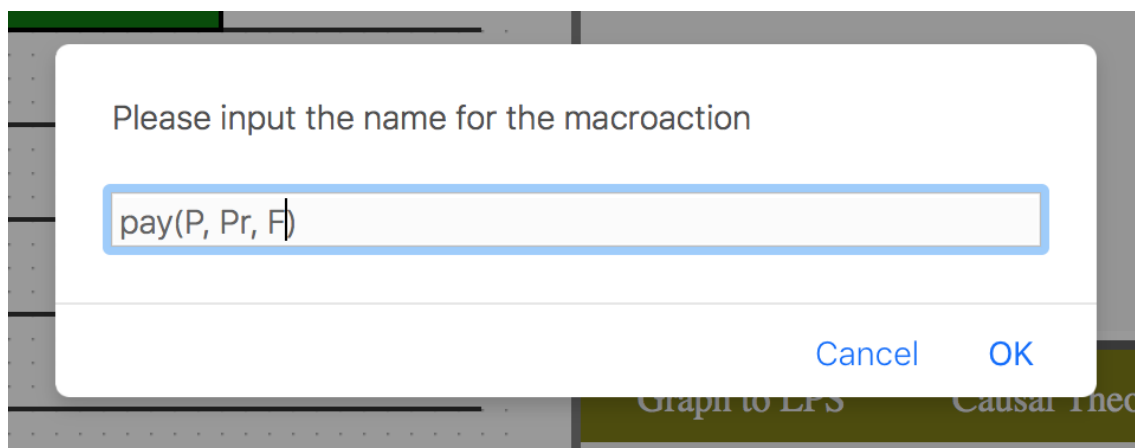


2. add the antecedents sequentially



**Figure C.16:** Macro Action antecedent

3. save the graph with name of the macro action predicate (in this case same with name "try\_to\_buy(P, F)") and click "Get Clause" to check



**Figure C.17:** Save5

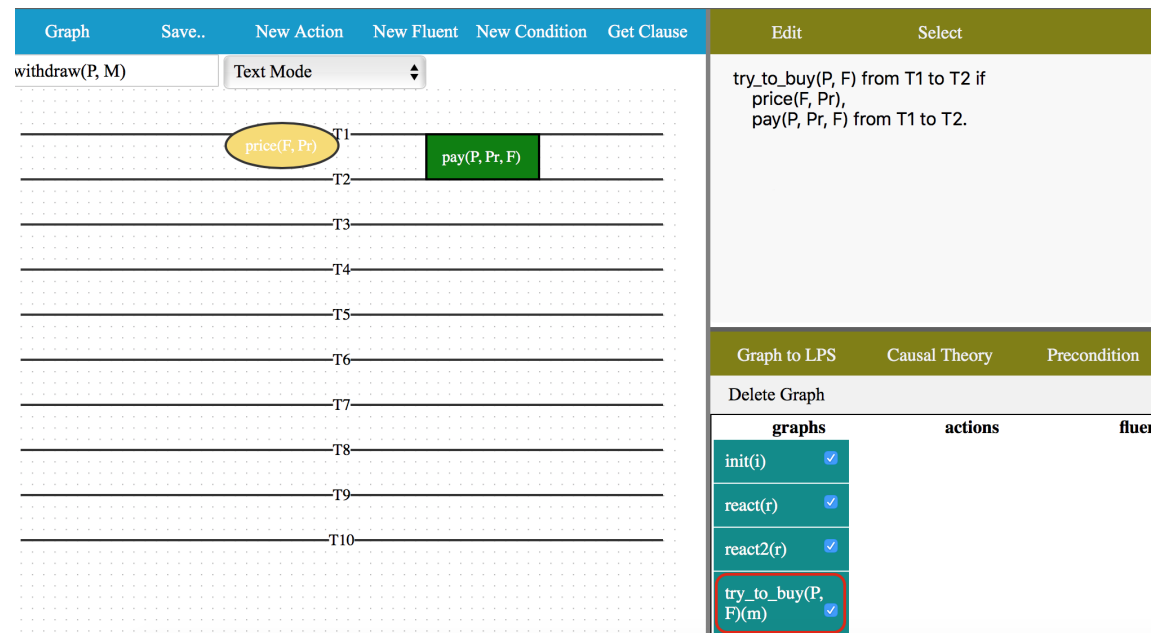


Figure C.18: Save6

4. repeat step 1-3 to generate more graphs

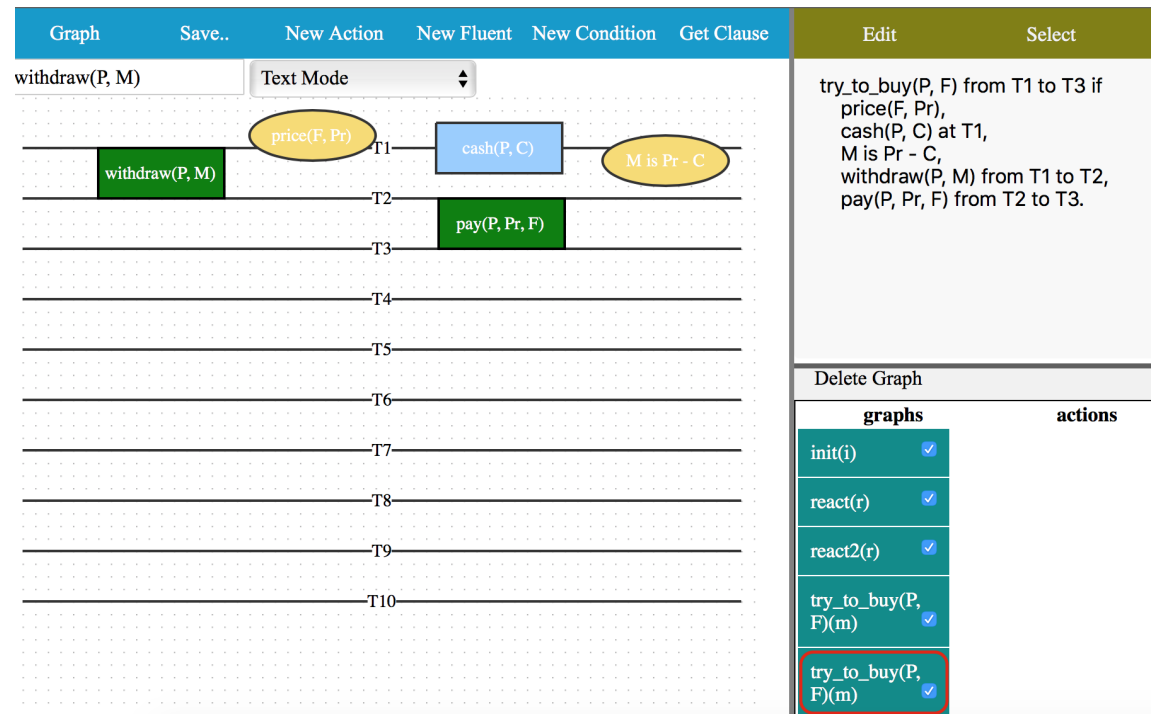


Figure C.19: Save7

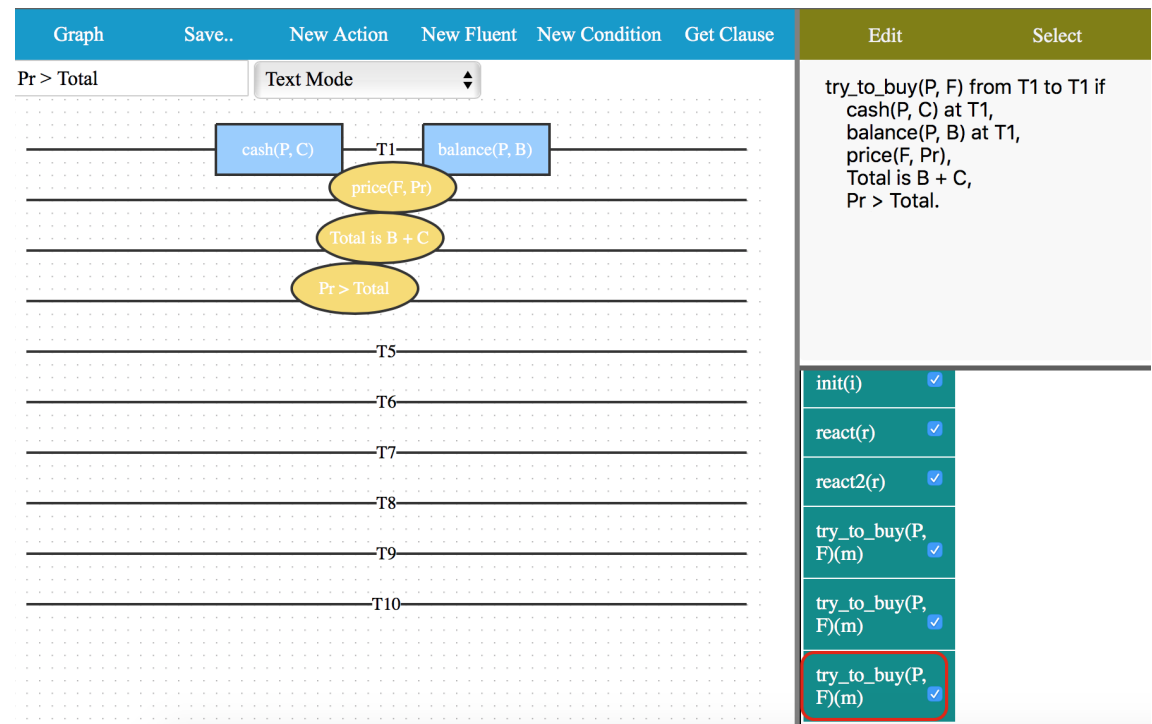


Figure C.20: Save8

### C.3 Generating Time-independent Clause

#### C.3.1 Checking

select the graph buttons we want to translate to final LPS code and click "to LPS" button. Confirm that in "Graph to LPS" panel predicates under actions column are all atomic.

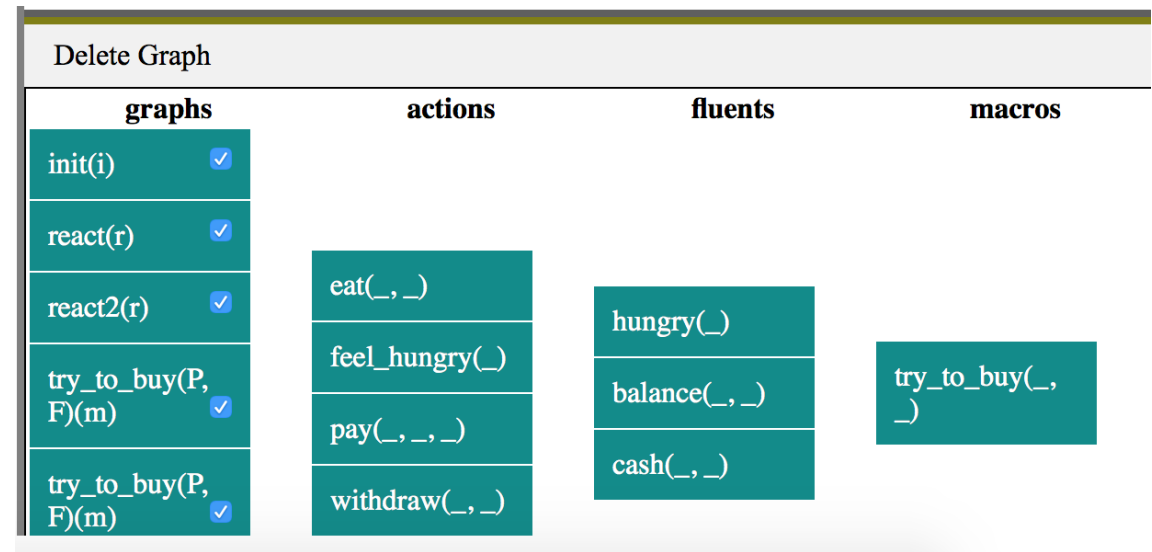


Figure C.21: Checking

The "delete graph" button delete all selected graphs in the graphs column.

### C.3.2 Causal Theory

Fluents cannot be directly modified, they can only be instantiated in the initial states, or be affected by actions in causal theory. For example, if a person P starts to feel hungry, fluent hungry(P) will be initiated. If the person eat something, hungry(P) will no longer be valid. In addition, if the person P withdraw some money M, his or her balance B will be updated to  $B - M$ , and cash C will be updated to  $C + M$ .

The steps are as follows.

1. click the Causal Theory button to enter the panel
2. for normal causal theory, select or input the action, fluent, and select the cause




Graph to LPS	Causal Theory	Precondition
add condition	reset	submit
		delete
<b>action</b>	feel_hungry(P)	feel_hungry(P) 
<b>cause</b>	initiates 	
<b>fluent</b>	hungry(P)	hungry(P) 
<input checked="" type="checkbox"/> feel_hungry(P) initiates hungry(P).		

Figure C.22: Causal Theory 1

3. for causal theory with condition, select or input the action, fluent, and select the cause, then click the add condition button and input or select the condition.

	Graph to LPS	Causal Theory	Precondition
	add condition	reset	submit delete
<b>action</b>	eat(P, F)		eat(P, F)
<b>cause</b>	terminates		
<b>fluent</b>	hungry(P)		hungry(P)
<b>condition</b>	food(F)		food(F)  delete

☒ feel\_hungry(P) initiates hungry(P).  
☒ eat(P, F) terminates hungry(P) if food(F).

Figure C.23: Causal Theory 2

4. if the cause is selected to be update, a new line will appear in the middle for users to input the original state and the new state of a variable in the fluent.

	Graph to LPS	Causal Theory	Precondition
	add condition	reset	submit delete
<b>action</b>	withdraw(P, M)		withdraw(P, M)
<b>cause</b>	updates		
	B	to	NewB in
<b>fluent</b>	balance(P, B)		balance(P, B)
<b>condition</b>	NewB is B - M		food(F)  delete

☒ feel\_hungry(P) initiates hungry(P).  
☒ eat(P, F) terminates hungry(P) if food(F).  
☒ withdraw(P, M) updates B to NewB in balance(P, B) if NewB is B - M.

Figure C.24: Causal Theory 3

5. finally, click the submit button to generate the clause.

☒ feel\_hungry(P) initiates hungry(P).  
☒ eat(P, F) terminates hungry(P) if food(F).  
☒ withdraw(P, M) updates B to NewB in balance(P, B) if NewB is B - M.  
☒ withdraw(P, M) updates C to NewC in cash(P, C) if NewC is C + M.  
☒ pay(P, Pr, F) updates C to NewC in cash(P, C) if NewC is C - Pr.

Figure C.25: Causal Theory 4

Reset button reset the panel to its original state, the delete button in the button bar delete selected clauses at the bottom, and the delete button at the end of each row of condition delete the entire row. The contents of the select area and the autocomplete function of the input area is presented only if we generate the LPS code for the graphs previously.

### C.3.3 Precondition

Apparently there are some predicates that cannot happen together: if person P holds less amount of cash than the price Pr of product F, P cannot pay for the product; Person P cannot withdraw more money than the balance of the account.

The steps for the generating preconditions are:

1. click the Precondition button to enter the panel
2. click add action, add fluent, or add condition several times to create the input area
3. select or directly input the action, fluent, or condition
4. click submit button and the clause will be generated at the bottom

Graph to LPS    Causal Theory    **Precondition (1)**

add action (2)    add fluent (3)    add condition (4)    reset    submit (8)    delete

Please input actions, fluents, and conditions which cannot happen together

<b>action</b>	pay(P, Pr, F)	pay(P, Pr, F) (5) select if exists	delete
<b>fluent</b>	cash(P, C)	cash(P, C) (6) select if exists	delete
<b>condition</b>	Pr > C (7) directly input if not exists in the select area	please select a conditic	delete

☒ false pay(P, Pr, F), cash(P, C), Pr > C. (9) Clause is generated

Figure C.26: Precondition1

Graph to LPS		Causal Theory		Precondition	
add action	add fluent	add condition	reset	submit	delete

Please input actions, fluents, and conditions which cannot happen together

<b>action</b>	<input type="text" value="withdraw(P, M)"/>	<input type="text" value="withdraw(P, M)"/>	<input type="text" value="delete"/>
<b>fluent</b>	<input type="text" value="balance(P, B)"/>	<input type="text" value="balance(P, B)"/>	<input type="text" value="delete"/>
<b>condition</b>	<input type="text" value="B &lt; M"/>	<input type="text" value="please select a conditio"/>	<input type="text" value="delete"/>

☒ false pay(P, Pr, F), cash(P, C), Pr > C.  
☒ false withdraw(P, M), balance(P, B), B < M.

**Figure C.27: Precondition2**

The reset button clear the input field and set the Precondition panel to its initial state, the delete button in the button bar delete selected clauses at the bottom, and the delete button at the end of the row of input field delete the entire row. The contents of the select area and the autocomplete function of the input area is presented only if we generate the LPS code for the graphs previously.

## C.4 Modification

### C.4.1 Locate the Graph

There are two ways to locate the graph, one way is to click the graph button in "Graph to LPS" panel directly, and the other is to double click the code in the select panel of code area, and the corresponding graph will be shown in the graph region.

### C.4.2 Modify the Graph

click the corresponding graph button in "Graph to LPS" panel or code in select panel, and the graph will be demonstrated in the graph region. After the modification, hover over "Save.." and click the "Save" button, and the modification will be saved.

### C.4.3 Duplicate the Graph

click the corresponding graph button in "Graph to LPS" panel or code in select panel, and the graph will be demonstrated in the graph region. After the modification, hover over "Save.." and click the "Save As" button to save the graph to a new name, and the new graph will be saved separately, and the original graph is not modified.

## C.5 Download and Upload

When we get all graphs, causal theories, and preconditions ready, we can select those we want, and click "Download" button to save the json string of the page into a text file.

The next time we open easyLPS, we can upload the saved file, and all graphs, causal theories, and preconditions are preserved. (Clicking to LPS button this time can regenerate the LPS code).

## C.6 Run in Swish

When we get all graphs, causal theories, and preconditions ready, we can select those we want, and click "to LPS" button again to generate the final LPS code.

		Download	Upload file	Run in Swish
Edit	Select			
<pre> actions eat(_, _), feel_hungry(_), pay(_, _, _), withdraw(_, _). fluents hungry(_), balance(_, _), cash(_, _). observe feel_hungry(bob) from 3 to 4. initially cash(bob, 0), balance(bob, 100). food(burger). price(burger, 10).  try_to_buy(P, F) from T1 to T2 if   price(F, Pr),   pay(P, Pr, F) from T1 to T2. try_to_buy(P, F) from T1 to T3 if   price(F, Pr),   cash(P, C) at T1           </pre>				

Figure C.28: Final Code

After the code is generated, we copy the code, click "Run in Swish" button, create a new program in Swish, and paste the code. Then type in "go(Timeline)" in the command line, and click "Run!" to get the running result of LPS.



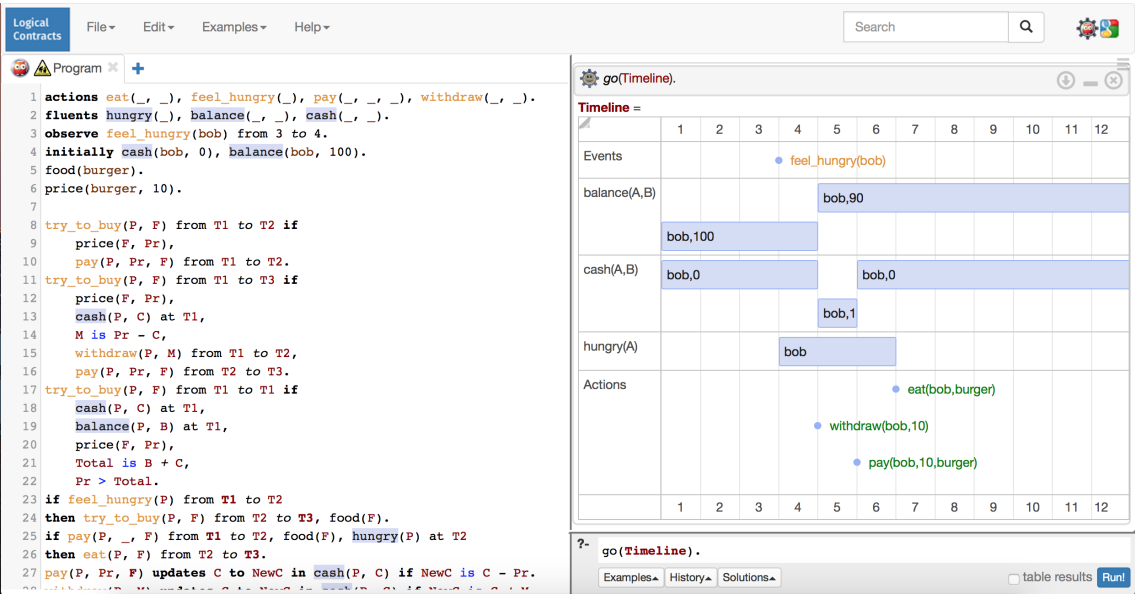


Figure C.29: Run in Swish