Warsaw University of Technology's
Faculty of Mathematics and Information Science

Knowledge Representation and Reasoning

Project number 2:
Deterministic Action With Cost
Supervisor: Dr Anna Radzikowska

Test case Documentation

Created By
Rishabh Jain, Rahul Tomer, Kuldeep Shankar,
Alaa Abboushi, Haran Dev Murugan,
Bui Tuan Anh.

Contents

1 Introduction

This document is to Show the test cases of the application that is used to implement the language $\mathcal{A}$ with cost statements. The Languages assumptions are as follows. Let C2 be a class of dynamic systems satisfying the following assumptions:

1. Inertia law
2. Complete information about all actions and fluent.
3. Only Determinism
4. Only sequential actions are allowed.
5. Characterizations of actions:
   - Precondition represented by set of literals(a fluent or its negation);if a precondition does not hold, the action is executed but with empty effect
   - Postcondition (effect of an action) represented by a set of literals.
   - Cost $k \in N$ of an action, actions with empty effects cost 0. Each action has a fixed cost, if it leads to non-empty effects.
6. Effects of an action depends on the state where the action starts.
7. All actions are performed in all states.
8. Partial description of any state of the system are allowed.
9. No constraints are defined.

2 Examples

2.0.1 Description

Andrew wants to travel by his car to a place. Travelling costs him 50$ when there is fuel in car tank. Travelling costs him 50$ when there is fuel in reserve. When there is no fuel in any of it, Andrew can fuy fuel. Fuel costs him 40$

2.0.2 Representation

Fluents: F = {fuel, reserve}
Actions: Ac = {buy, travel}
Costs: K = {40, 50}

initially fuel;
initially ¬reserve;
travel causes ¬fuel if fuel, reserve;
travel causes ¬reserve if ¬fuel, reserve;
travel causes ¬fuel if fuel, ¬reserve
travel costs 50;
buy causes fuel if ¬ fuel,reserve;
buy causes fuel if ¬ fuel,¬reserve;
buy causes reserve if fuel,¬reserve;
buy costs 40;

2.0.3 Calculation

$\sum$ = { σ_0,σ_1,σ_2,σ_3 }

σ_0 = { fuel, ¬reserve } σ_1 = { ¬fuel, ¬reserve }
σ_2 = { ¬fuel, reserve } σ_3 = { fuel,reserve }
Ψ(buy, σ_0)=σ_3
Ψ(travel, σ_0)=σ_1
Γ(buy, σ_0) = 40
Γ(travel, σ_0) = 50

Ψ(buy, σ_1)=σ_0
Ψ(travel, σ_1)=σ_1
Γ(buy, σ_1) = 40
Γ(travel, σ_1) = 0

Ψ(buy, σ_2)=σ_3
Ψ(travel, σ_2)=σ_1
Γ(buy, σ_2) = 40
Γ(travel, σ_2) = 50

Ψ(buy, σ_3)=σ_3

Ψ(travel, σ_3)=σ_2
Γ(buy, σ_3) = 0
Γ(travel, σ_3) = 50

2.0.4 Graph

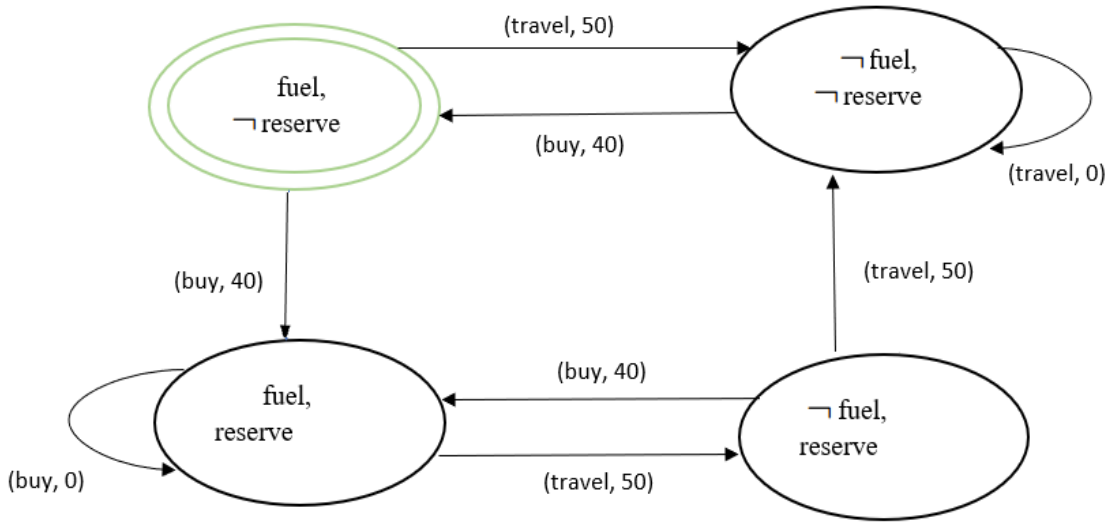

Figure 1: Example 01

2.0.5 Queries

reserve holds after (travel, travel, buy, buy): True
fuel holds after (buy, travel, buy,travel): False

150 sufficient for (travel, travel, buy, travel): True
90 sufficient for (buy, travel, buy): False

Test Images

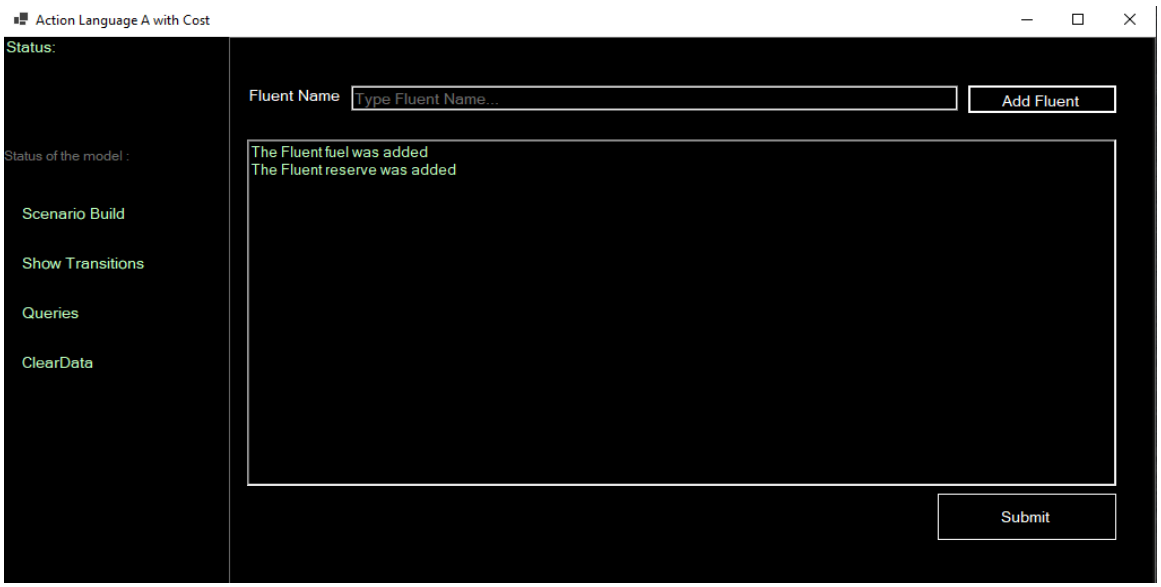

Figure 2: Add Fluents

Figure 3: Add Actions

Figure 4: Set Initial State fluents

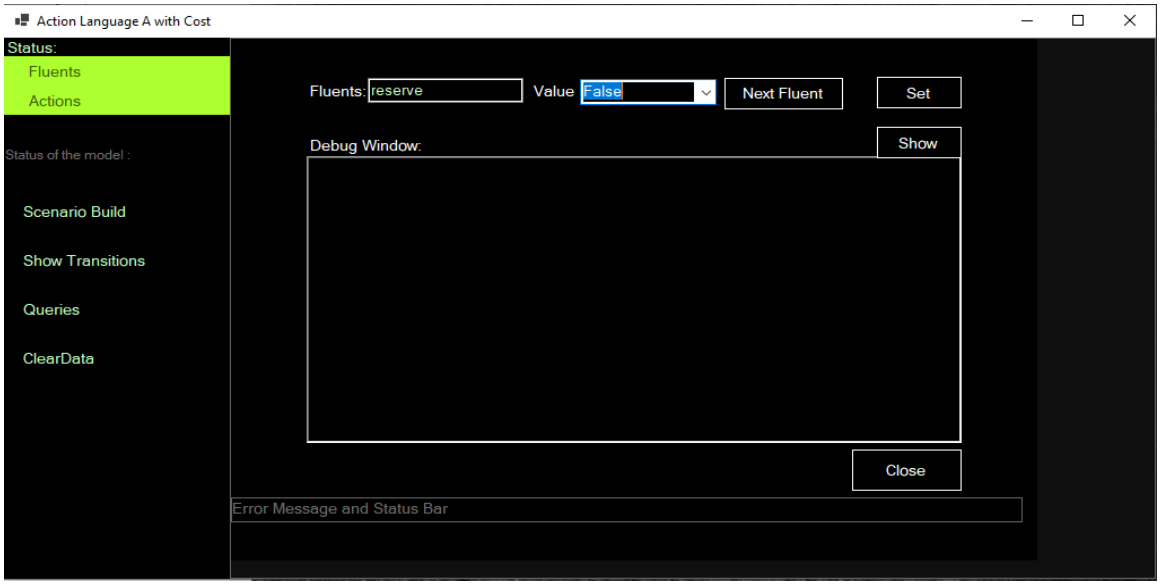

Figure 5: Set Initial state

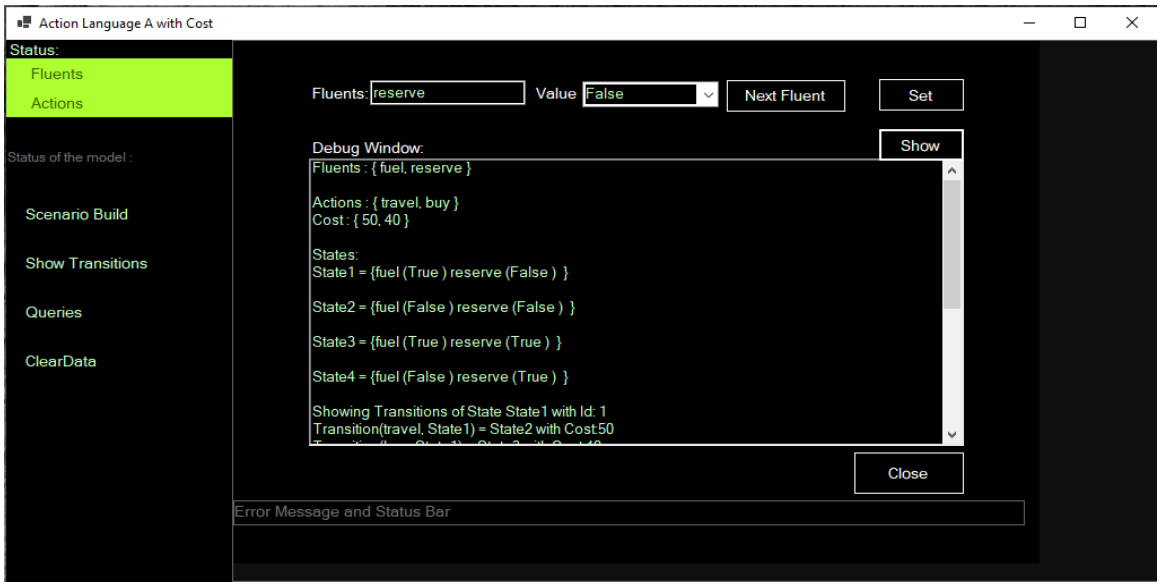

Figure 6: Show Transitions

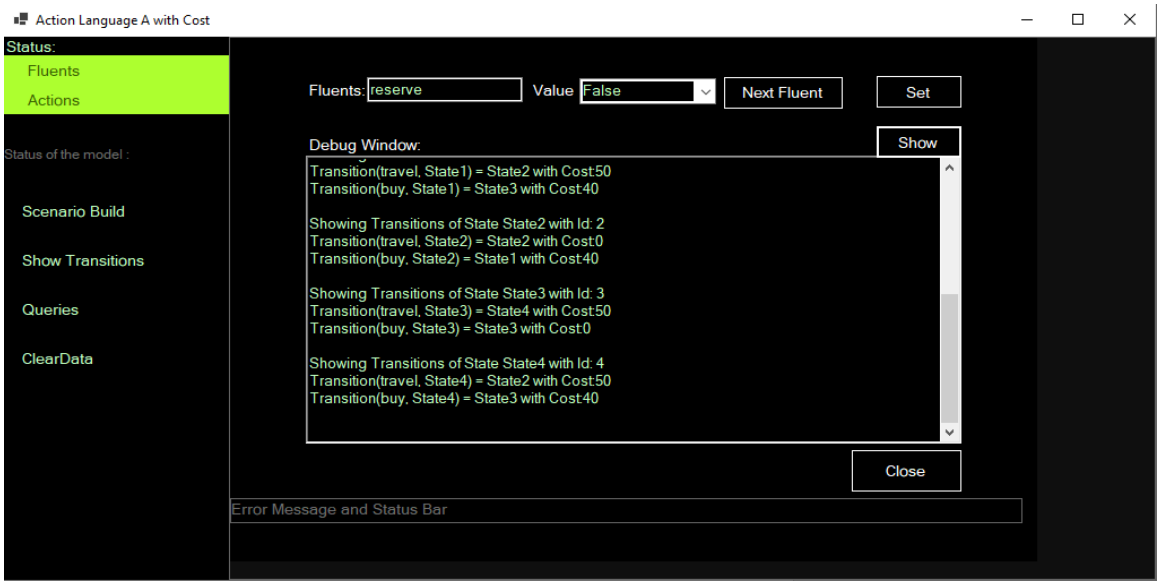

Figure 7: Show Transition contd..

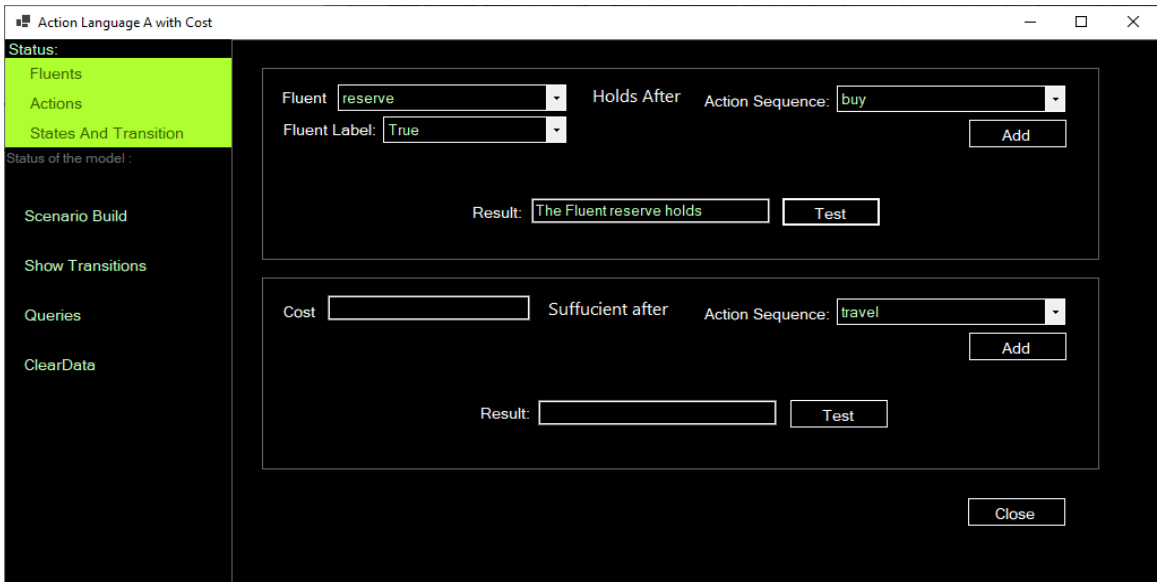

Figure 8: Queries(Fluents)

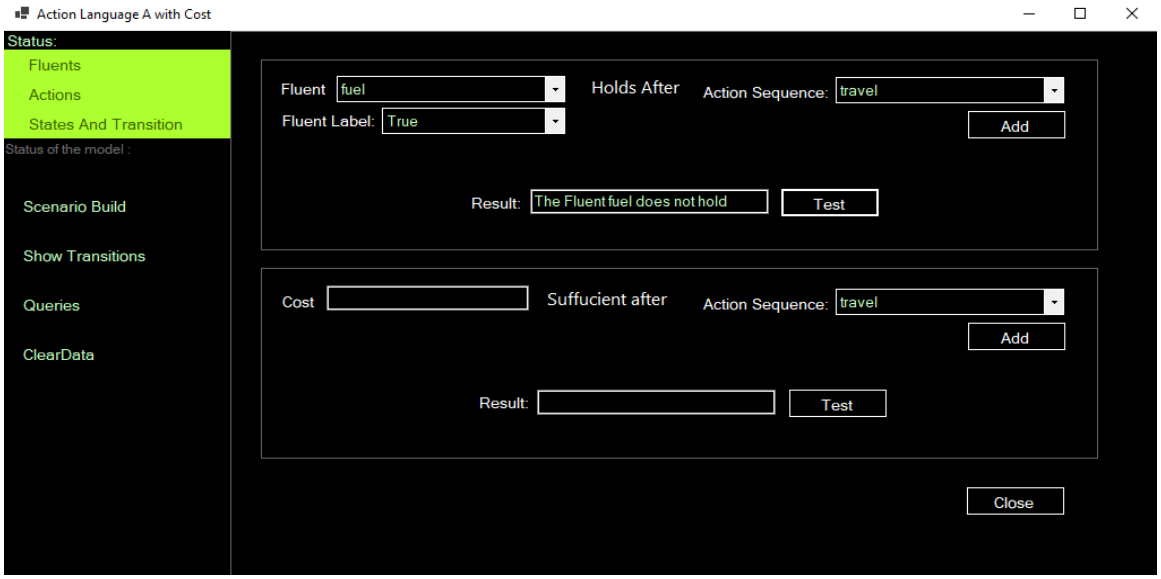

Figure 9: Queries(Fluents)

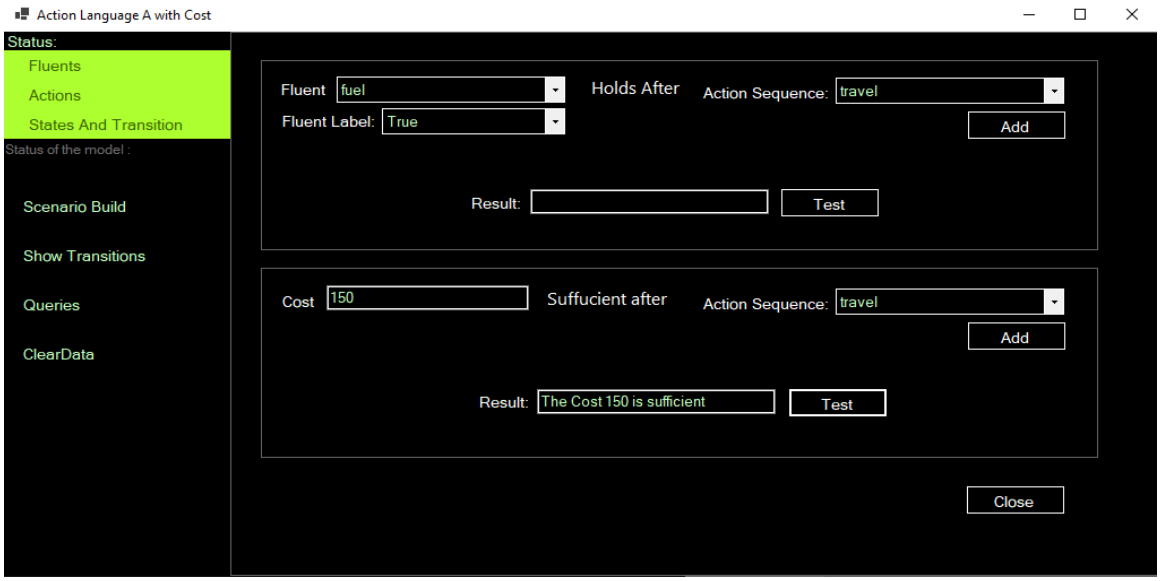

Figure 10: Queries(Cost)

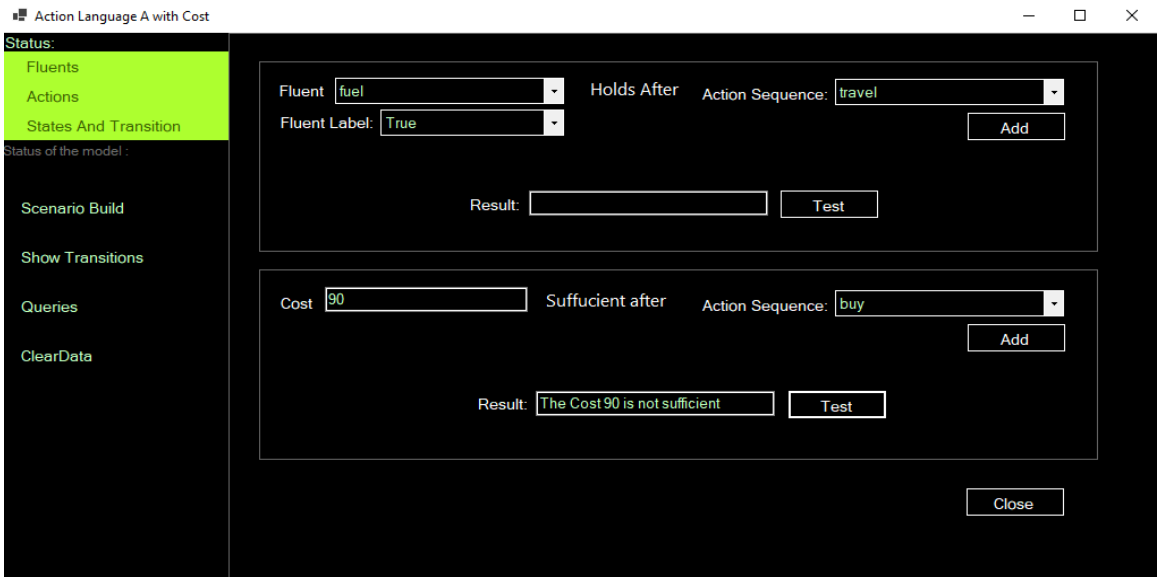

Figure 11: Queries(Cost)

2.1 Example 02

2.1.1 Description

John visits a painter to buy a specific painting. The cost of painting is 200$ if its available in the shop. But if painting is not available then John needs to order a new one to be painted and will buy once its available.Order costs 50$ At any time only one copy of painting is available and another one to be ordered once sold.

2.1.2 Representation:

Fluents: F = {available, sold}
Actions: Ac = {buy, order}
Costs: K = {200, 50}
initially ¬available;
initially ¬sold;
buy causes sold if available;
buy causes ¬available;
buy costs 200$;
order causes available if ¬available;

order costs 50$;

2.1.3 Calculation:

$\sum = \{ \sigma_0, \sigma_1, \sigma_2, \sigma_3\}$
$\sigma_0 = \{ \neg$available, $\neg$sold$\}$
$\sigma_1 = \{$ available, $\neg$sold$\}$
$\sigma_2 = \{ \neg$available, sold$\}$
$\sigma_3 = \{$ available, sold$\}$

Ψ (buy, $\sigma 0$) = $\sigma 0$
Ψ (order, $\sigma 0$) = $\sigma 1$
Γ(buy, σ_0) = 0
Γ(order, σ_0) = 50

Ψ (buy $\sigma 1$) = $\sigma 2$
Ψ (order, $\sigma 1$) = $\sigma 1$
Γ(buy, σ_1) = 200
Γ(order, σ_1) = 0

Ψ (buy, $\sigma 2$) = $\sigma 2$
Ψ (order, $\sigma 2$) = $\sigma 3$
Γ(buy, σ_2) = 0
Γ(order, σ_2) = 50

Ψ (buy, $\sigma 3$) = $\sigma 2$
Ψ (order, $\sigma 3$) = $\sigma 3$
Γ(buy, σ_3) = 200
Γ(order, σ_3) = 0

2.1.4 Graph

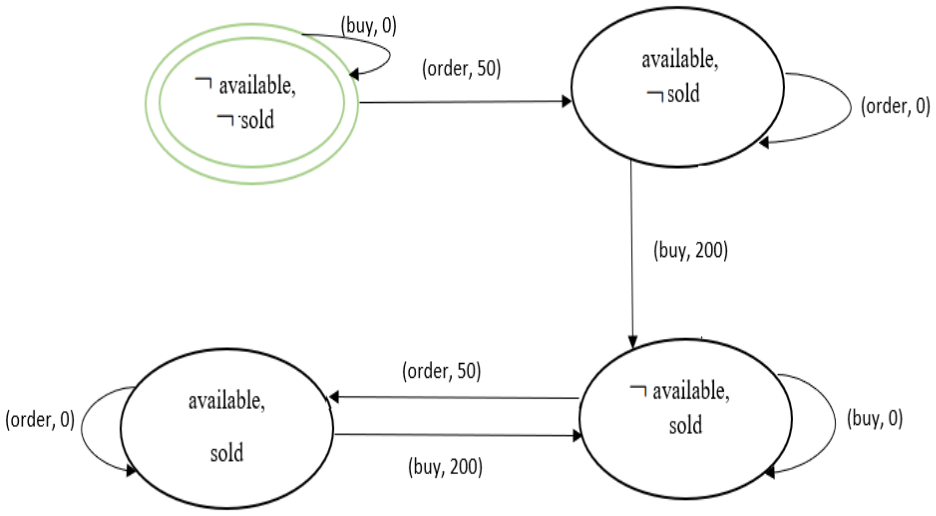

Figure 12: Example 02

2.1.5 Queries

available holds after (buy, order, buy, order): True
available holds after (buy, order, buy): False

275 sufficient for (buy, order, order, buy): True
190 sufficient for (order, buy, buy, order): False

Test Images

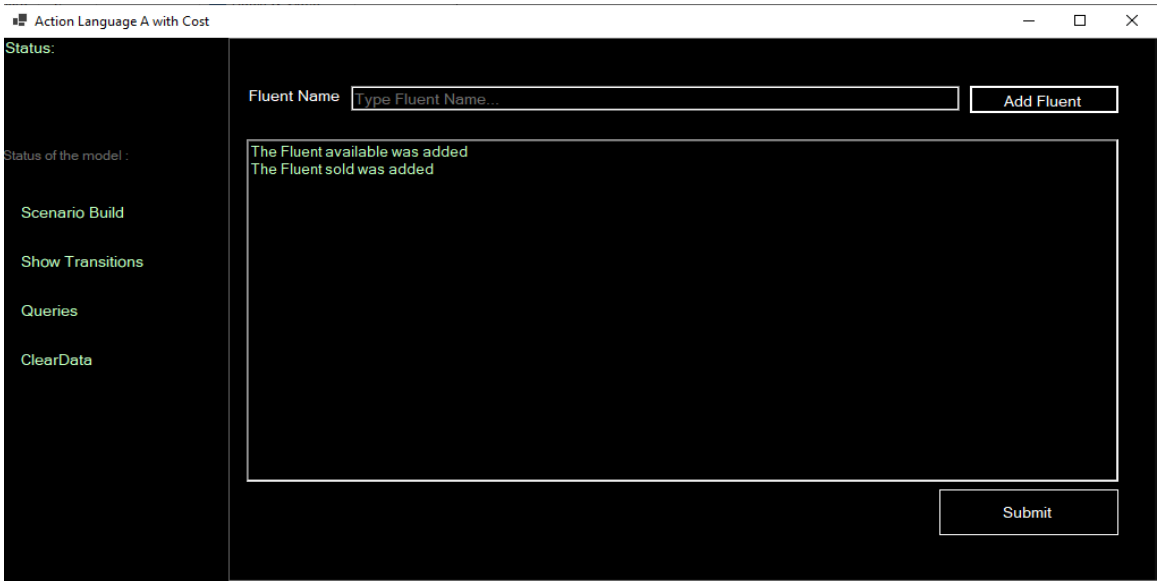

Figure 13: Add Fluents

Figure 14: Add Actions

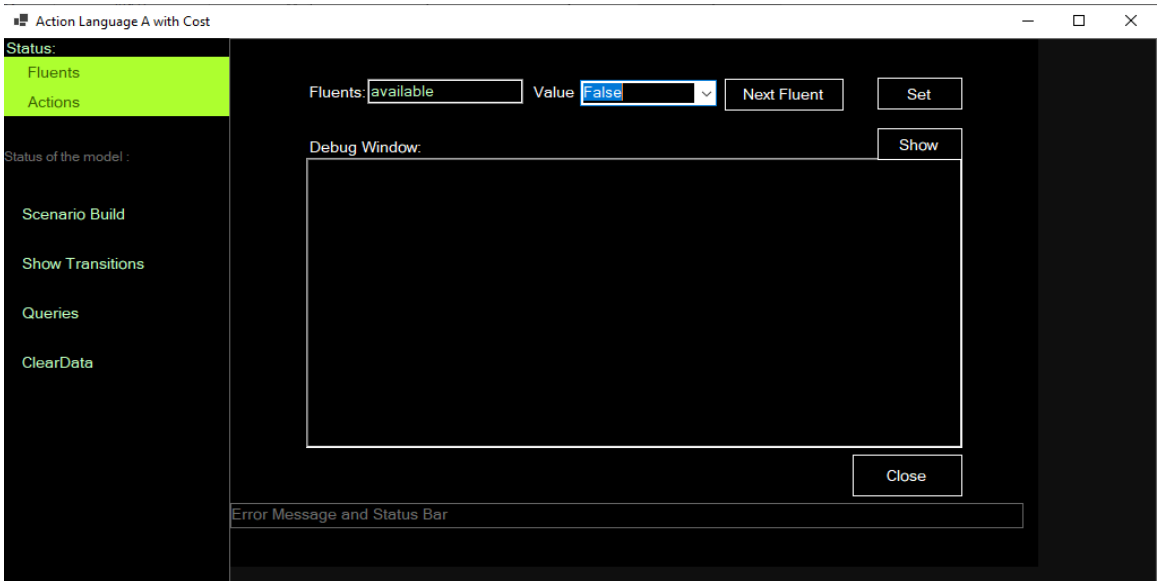

Figure 15: Set Initial State fluents

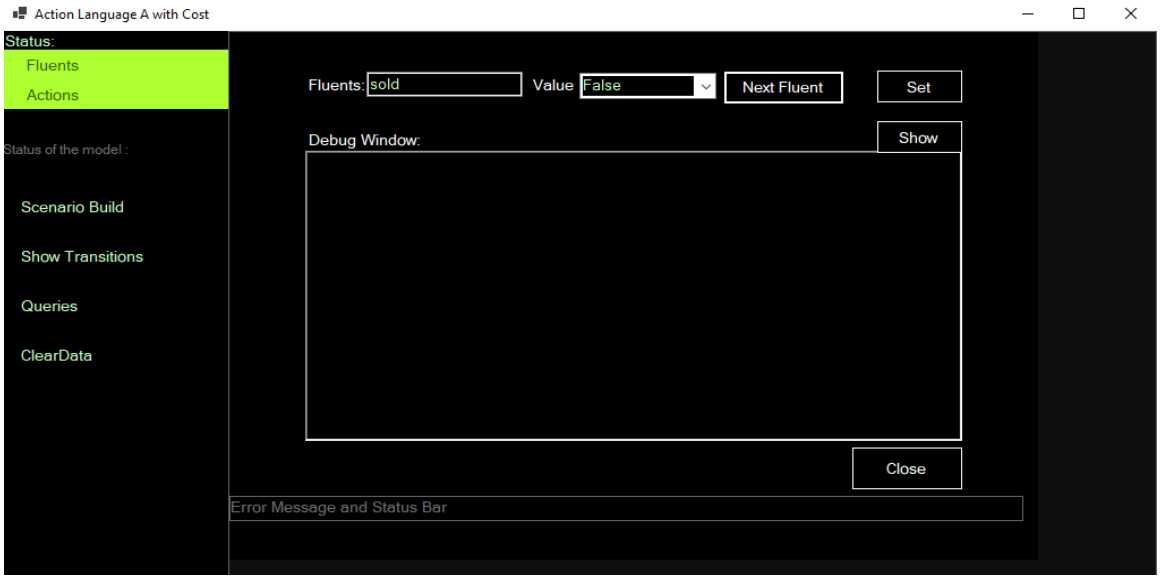

Figure 16: Set Initial state

Figure 17: Show Transitions

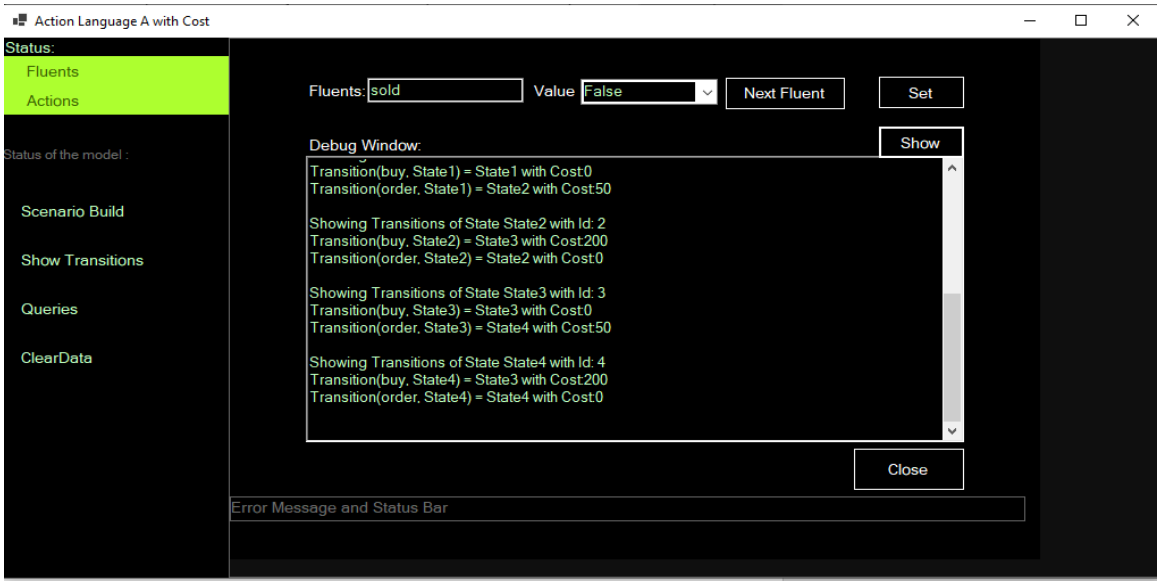

Figure 18: Show Transition contd..

Figure 19: Queries(Fluents)

Figure 20: Queries(Fluents)

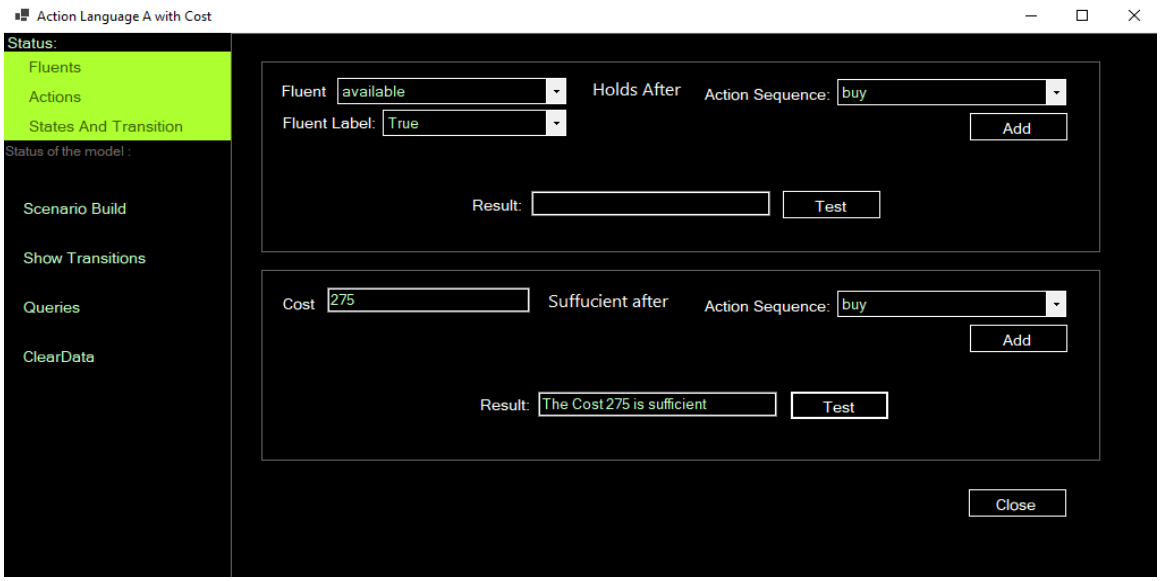

Figure 21: Queries(Cost)

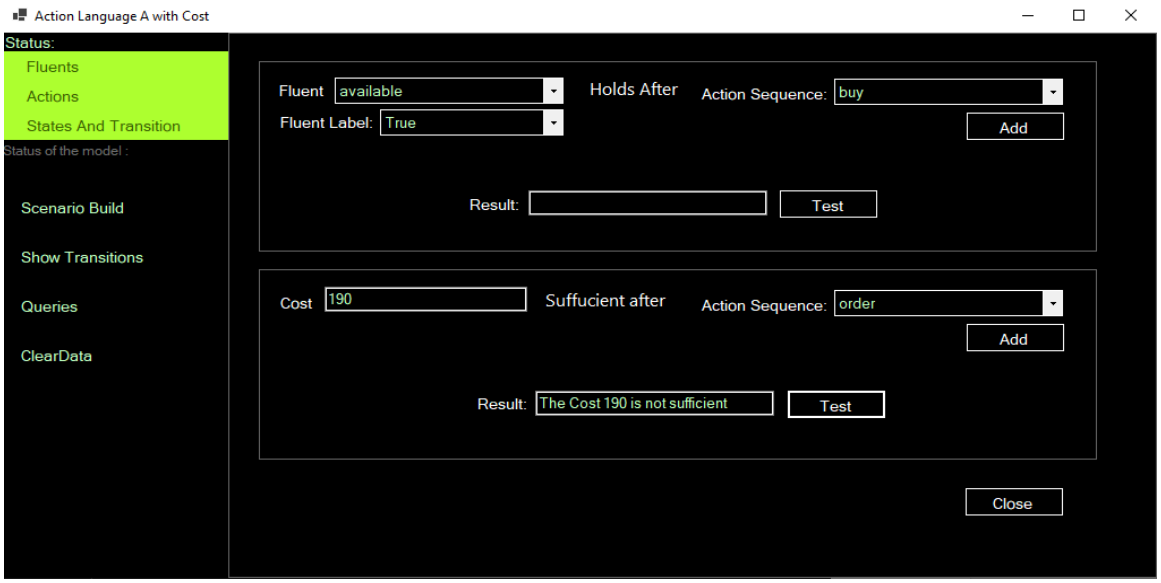

Figure 22: Queries(Cost)

2.2 Example 03

2.2.1 Description

There is a man. He can cook, eat, and play. Cooking makes food cooked. he can eat food if it is cooked. After eating he feels not hungry, and food is not cooked again. He can play. Playing makes him hungry. He just can play if he is not hungry. He just cooks when there is no food is cooked. Initially, he is hungry, and no food is cooked. In terms of energy, eating costs 5, cooking costs 15, playing costs 20.

2.2.2 Representation in language

Fluents: F = {cooked, hungry}
Actions: Ac = {cook, eat, play}
Costs: K = {15, 5, 20}

initially ¬cooked;
initially hungry;

cook causes cooked if ¬cooked;
cook costs 15;
eat causes ¬cooked if cooked;
eat causes ¬hungry if cooked;
eat costs 5;
play causes hungry if ¬hungry;
play costs 20;

2.2.3 Calculation

$\sum = \{\sigma_0 , \sigma_1, \sigma_2, \sigma_3\}$

$\sigma_0 = \{\neg\text{cooked, hungry}\}$
$\sigma_1 = \{\text{cooked, hungry}\}$
$\sigma_2 = \{\neg\text{cooked, }\neg\text{hungry}\}$
$\sigma_3 = \{\text{cooked, }\neg\text{hungry}\}$

$\Psi(\text{eat}, \sigma_0) = \sigma_0$
$\Psi(\text{cook}, \sigma_0) = \sigma_1$
$\Psi(\text{play}, \sigma_0) = \sigma_0$
$\Gamma(\text{eat}, \sigma_0) = 0$
$\Gamma(\text{cook}, \sigma_0) = 15$
$\Gamma(\text{play}, \sigma_0) = 0$

$\Psi(\text{eat}, \sigma_1) = \sigma_2$
$\Psi(\text{cook}, \sigma_1) = \sigma_1$
$\Psi(\text{play}, \sigma_1) = \sigma_1$
$\Gamma(\text{eat}, \sigma_1) = 5$
$\Gamma(\text{cook}, \sigma_1) = 0$
$\Gamma(\text{play}, \sigma_1) = 0$

$\Psi(\text{eat}, \sigma_2) = \sigma_2$
$\Psi(\text{cook}, \sigma_2) = \sigma_3$
$\Psi(\text{play}, \sigma_2) = \sigma_0$
$\Gamma(\text{eat}, \sigma_2) = 0$
$\Gamma(\text{cook}, \sigma_2) = 15$
$\Gamma(\text{play}, \sigma_2) = 20$

$\Psi(\text{eat}, \sigma_3) = \sigma_2$
$\Psi(\text{cook}, \sigma_3) = \sigma_3$
$\Psi(\text{play}, \sigma_3) = \sigma_1$
$\Gamma(\text{eat}, \sigma_3) = 5$
$\Gamma(\text{cook}, \sigma_3) = 0$
$\Gamma(\text{play}, \sigma_3) = 20$

2.2.4 Graph

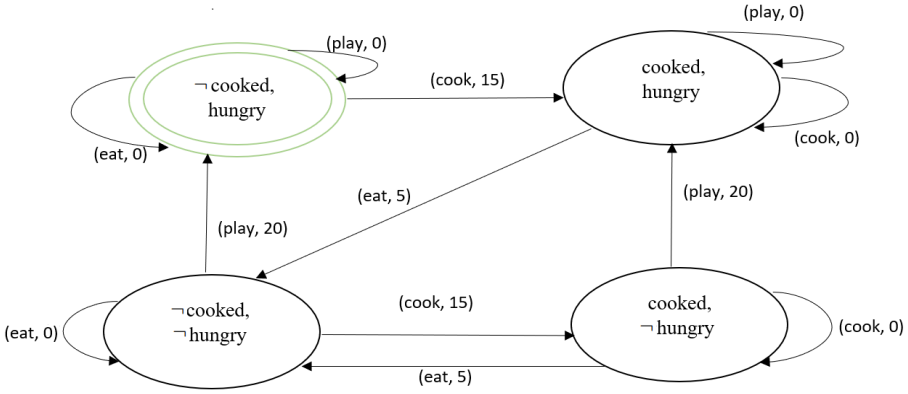

Figure 23: Example 03

2.2.5 Queries

cooked holds after (play, play, eat, cook): True
cooked holds after (cook, eat play, cook): False

40 sufficient for (play, cook, eat, cook): True
20 sufficient for (cook, eat, cook, play): False

Test Images

Figure 24: Add Fluents

Figure 25: Add Actions

Figure 26: Set Initial State fluents

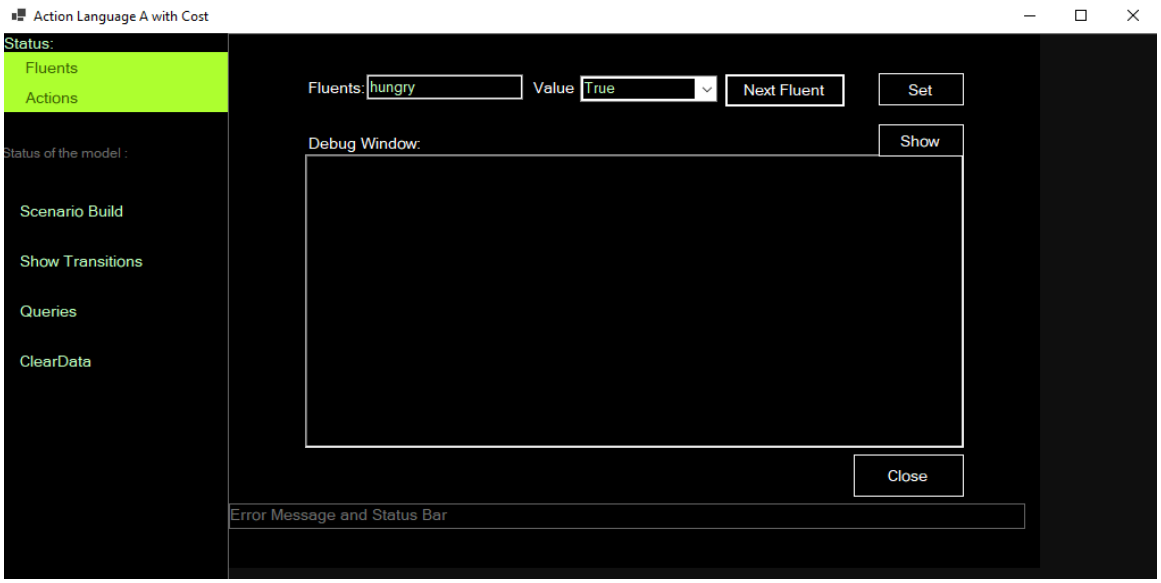

Figure 27: Set Initial state

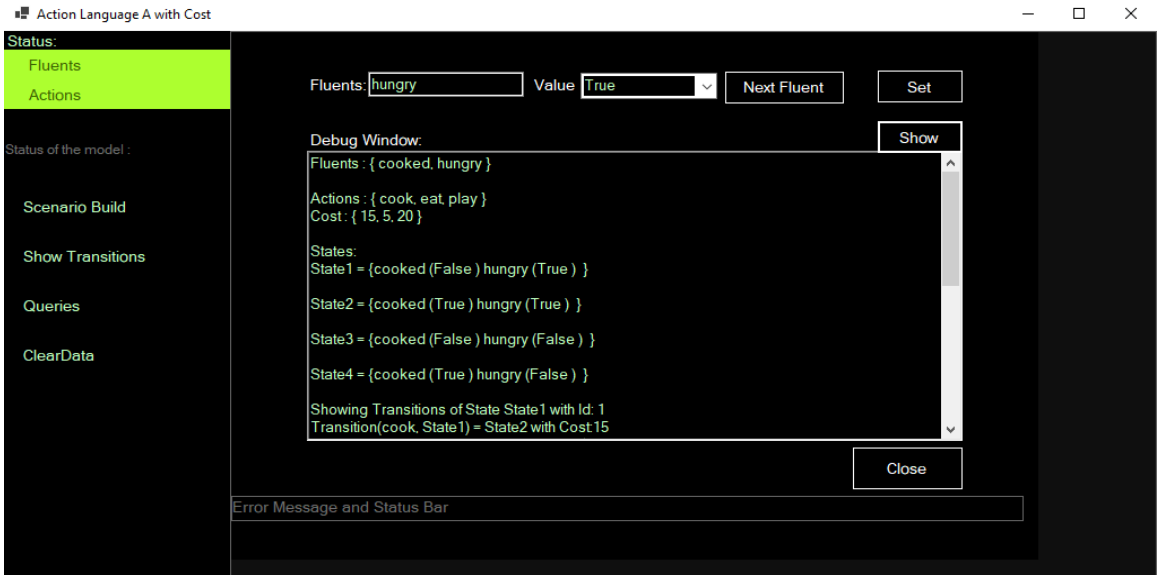

Figure 28: Show Transitions

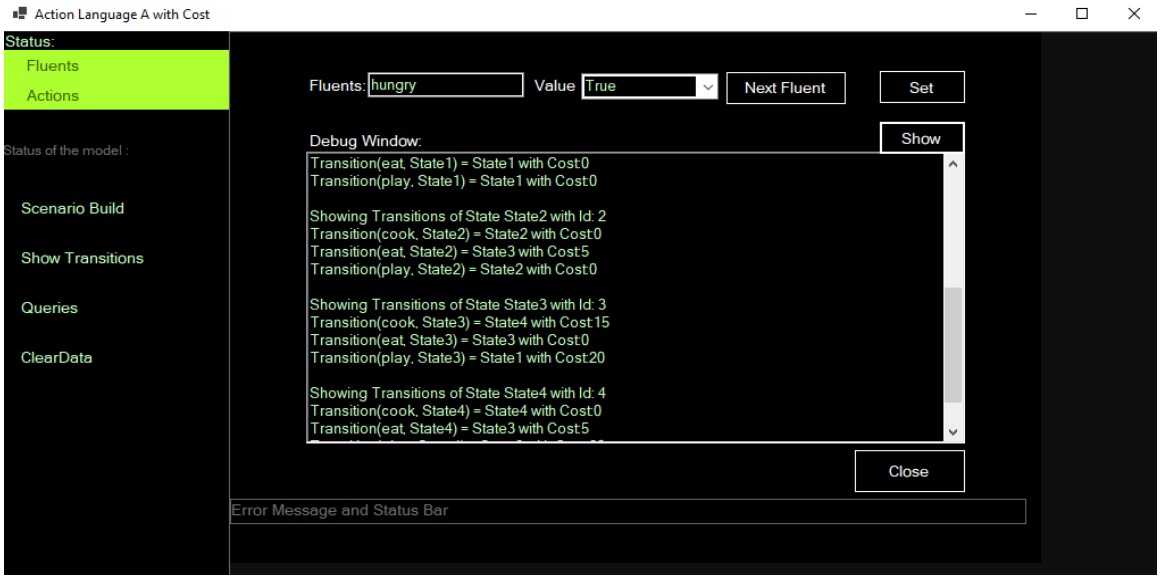

Figure 29: Show Transitions contd..

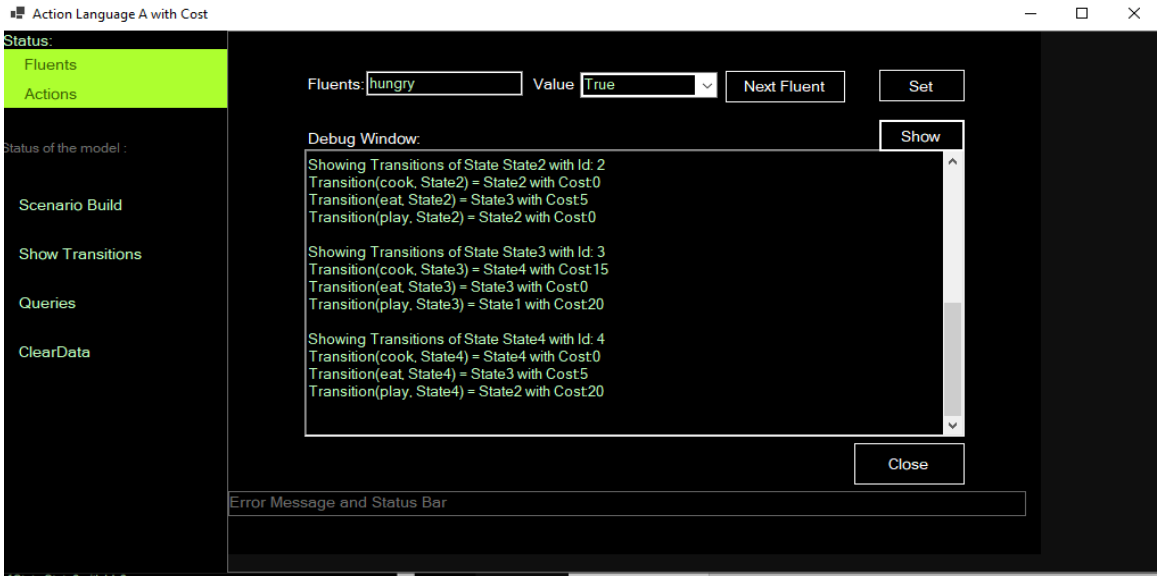

Figure 30: Show Transitions contd..)

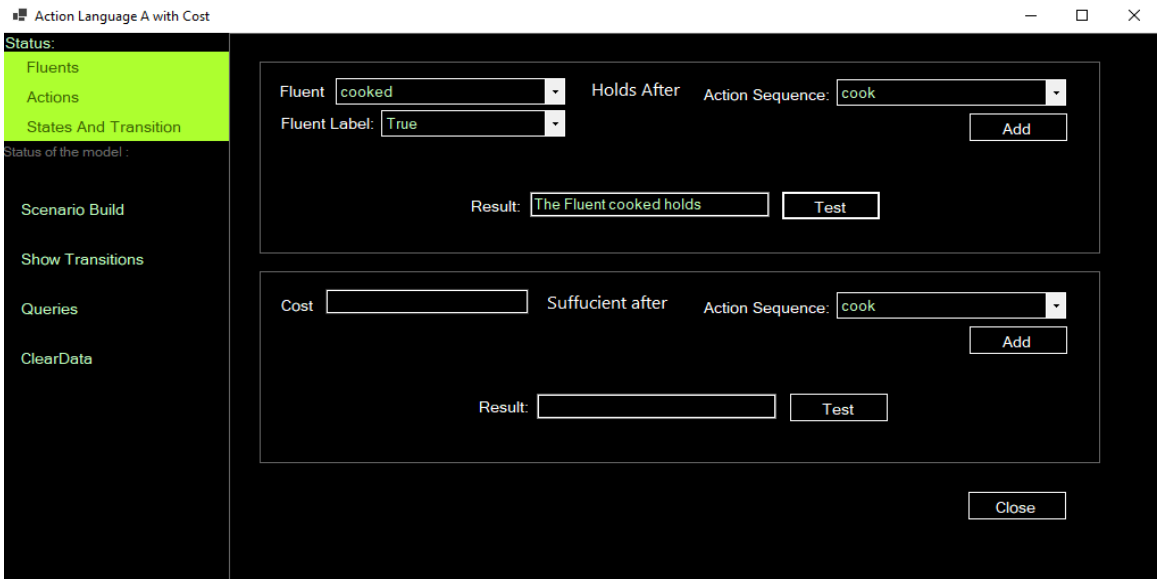

Figure 31: Queries(Fluents)

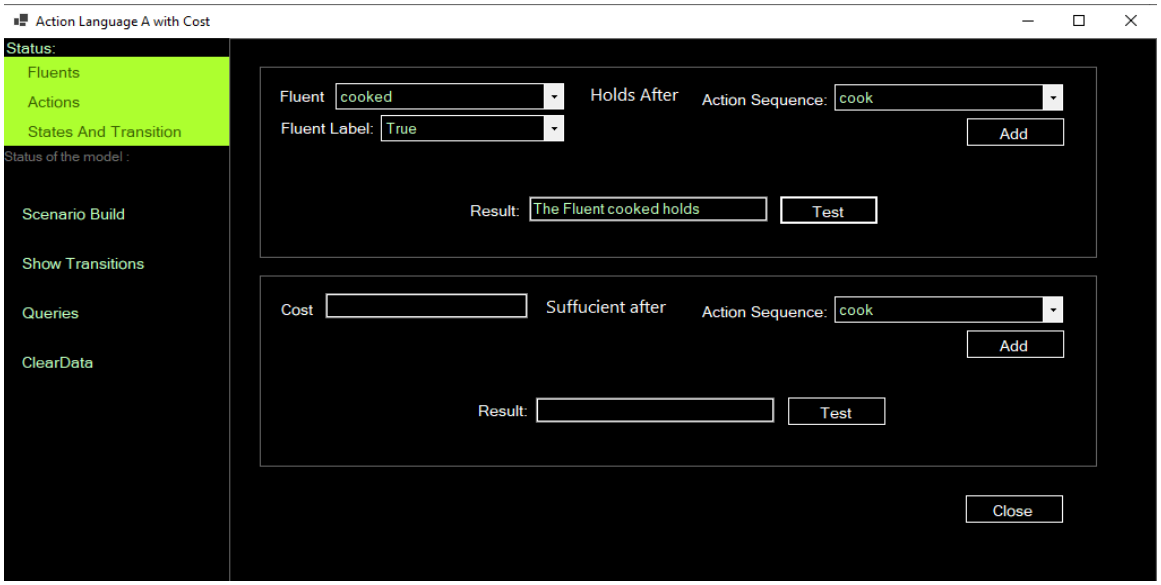

Figure 32: Queries(Fluents)

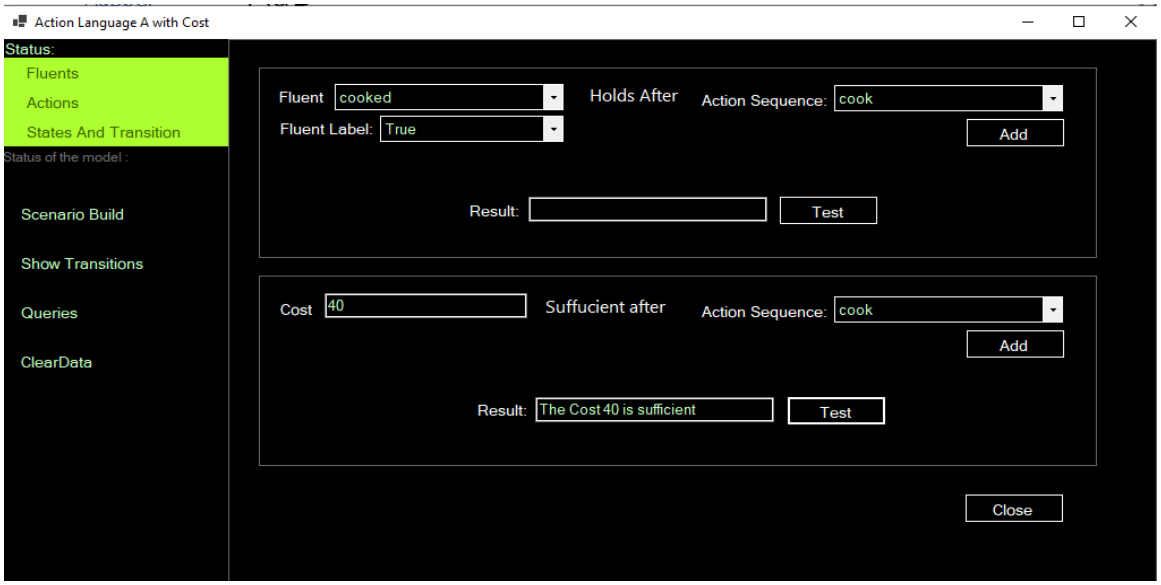

Figure 33: Queries(Cost)

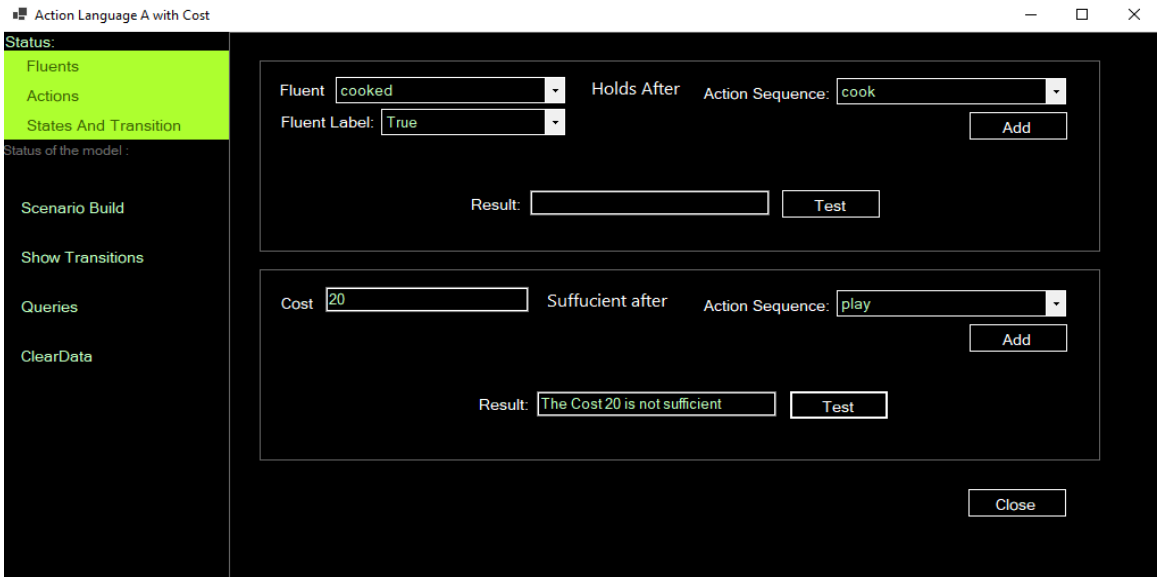

Figure 34: Queries(Cost)

3 Appendix

List of Figures

List of Tables