

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	2
2	Examples	2
2.0.1	Description	2
2.0.2	Representation	3
2.0.3	Calculation	3
2.0.4	Graph	4
2.0.5	Queries	4
2.1	Example 02	10
2.1.1	Description	10
2.1.2	Representation:	10
2.1.3	Calculation:	11
2.1.4	Graph	12
2.1.5	Queries	12
2.2	Example 03	18
2.2.1	Description	18
2.2.2	Representation in language	18
2.2.3	Calculation	19
2.2.4	Graph	20
2.2.5	Queries	20
3	Appendix	27

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: $F = \{\text{fuel}, \text{reserve}\}$

Actions: $Ac = \{\text{buy}, \text{travel}\}$

Costs: $K = \{40, 50\}$

initially fuel;

initially $\neg\text{reserve}$;

travel causes $\neg\text{fuel}$ if fuel, reserve;

travel causes $\neg\text{reserve}$ if $\neg\text{fuel}$, reserve;

travel causes $\neg\text{fuel}$ if fuel, $\neg\text{reserve}$

travel costs 50;

buy causes fuel if $\neg\text{fuel}, \text{reserve}$;

buy causes fuel if $\neg\text{fuel}, \neg\text{reserve}$;

buy causes reserve if fuel, $\neg\text{reserve}$;

buy costs 40;

2.0.3 Calculation

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

$$\sigma_0 = \{ \text{fuel}, \neg\text{reserve} \} \quad \sigma_1 = \{ \neg\text{fuel}, \neg\text{reserve} \}$$

$$\sigma_2 = \{ \neg\text{fuel}, \text{reserve} \} \quad \sigma_3 = \{ \text{fuel}, \text{reserve} \}$$

$$\Psi(\text{buy}, \sigma_0) = \sigma_3$$

$$\Psi(\text{travel}, \sigma_0) = \sigma_1$$

$$\Gamma(\text{buy}, \sigma_0) = 40$$

$$\Gamma(\text{travel}, \sigma_0) = 50$$

$$\Psi(\text{buy}, \sigma_1) = \sigma_0$$

$$\Psi(\text{travel}, \sigma_1) = \sigma_1$$

$$\Gamma(\text{buy}, \sigma_1) = 40$$

$$\Gamma(\text{travel}, \sigma_1) = 0$$

$$\Psi(\text{buy}, \sigma_2) = \sigma_3$$

$$\Psi(\text{travel}, \sigma_2) = \sigma_1$$

$$\Gamma(\text{buy}, \sigma_2) = 40$$

$$\Gamma(\text{travel}, \sigma_2) = 50$$

$$\Psi(\text{buy}, \sigma_3) = \sigma_3$$

$\Psi(\text{travel}, \sigma_3) = \sigma_2$
 $\Gamma(\text{buy}, \sigma_3) = 0$
 $\Gamma(\text{travel}, \sigma_3) = 50$

2.0.4 Graph

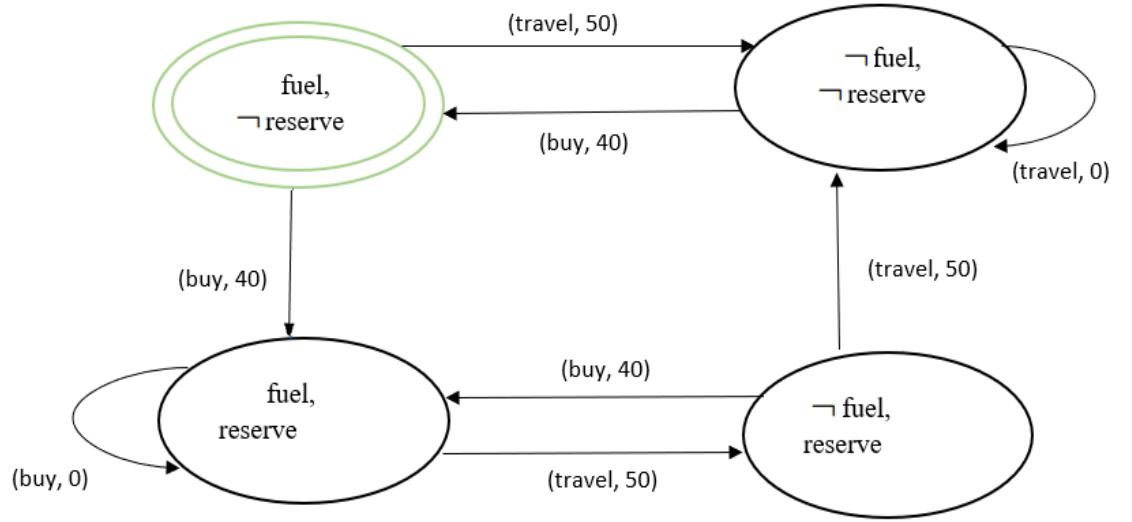


Figure 1: Example 01

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

The screenshot shows a web application window titled "Action Language A with Cost". The interface is divided into a left sidebar and a main content area. The sidebar contains the following links: "Status:", "Status of the model :", "Scenario Build", "Show Transitions", "Queries", and "ClearData". The main content area has a "Fluent Name" label followed by a text input field containing "Type Fluent Name...". To the right of the input field is an "Add Fluent" button. Below the input field is a large text area containing the text "The Fluent fuel was added" and "The Fluent reserve was added". At the bottom right of the main content area is a "Submit" button.

Action Language A with Cost

Status:

Status of the model :

Scenario Build

Show Transitions

Queries

ClearData

Fluent Name

Add Fluent

The Fluent fuel was added
The Fluent reserve was added

Submit

Figure 2: Add Fluents

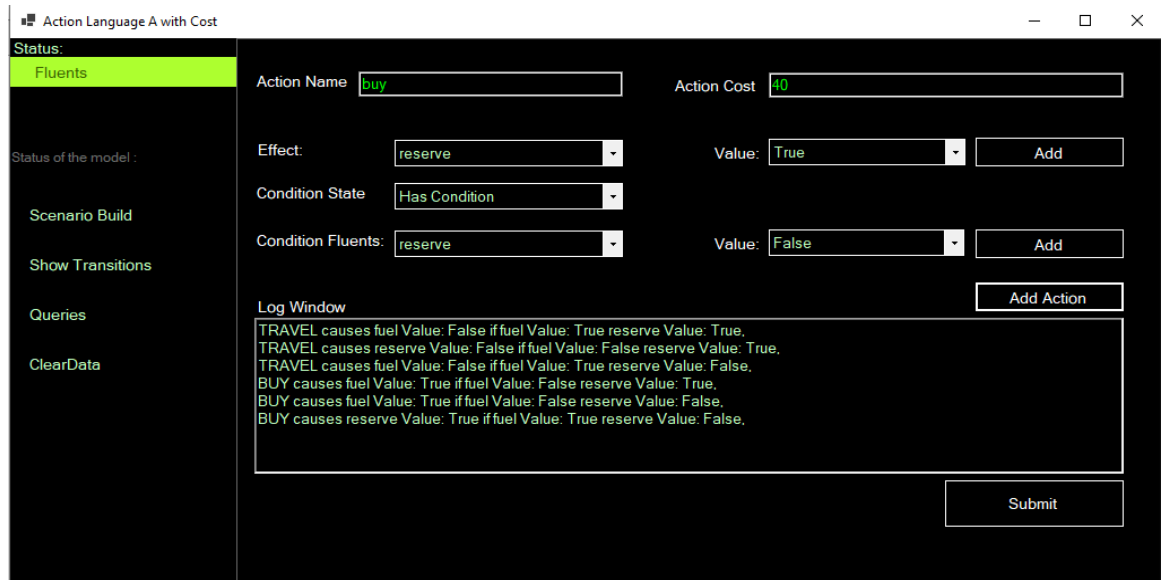


Figure 3: Add Actions



Figure 4: Set Initial State fluents



Figure 5: Set Initial state



Figure 6: Show Transitions



Figure 7: Show Transition contd..

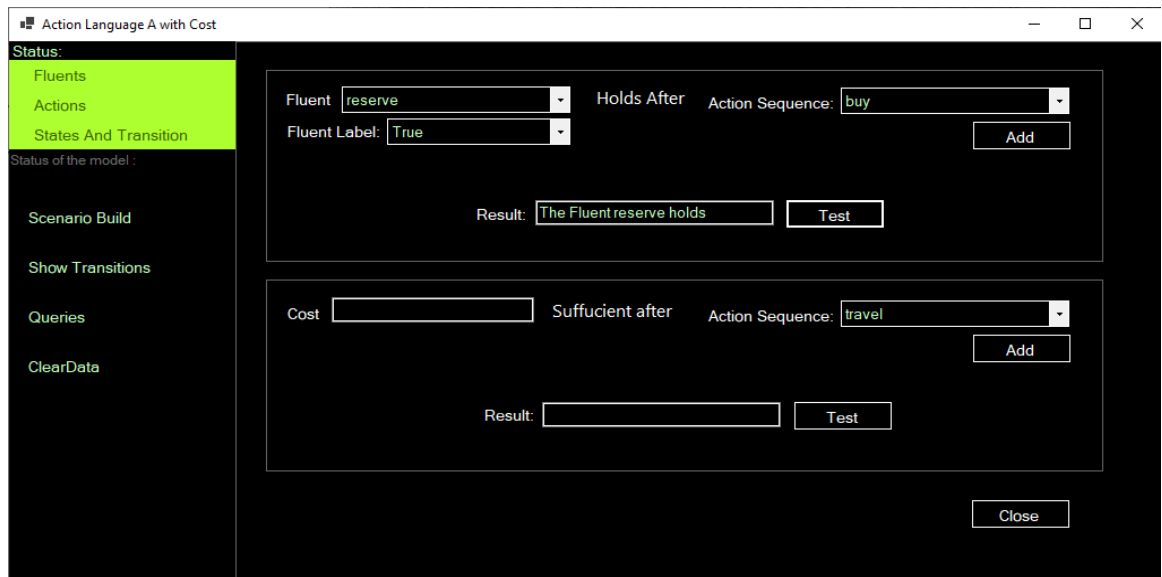


Figure 8: Queries(Fluents)

Action Language A with Cost

Status:

- Fluents
- Actions
- States And Transition

Status of the model :

Scenario Build

Show Transitions

Queries

ClearData

Fluent: Holds After Action Sequence:

Fluent Label:

Result:

Cost: Sufficent after Action Sequence:

Result:

Figure 9: Queries(Fluents)

Action Language A with Cost

Status:

- Fluents
- Actions
- States And Transition

Status of the model :

Scenario Build

Show Transitions

Queries

ClearData

Fluent: Holds After Action Sequence:

Fluent Label:

Result:

Cost: Sufficent after Action Sequence:

Result:

Figure 10: Queries(Cost)

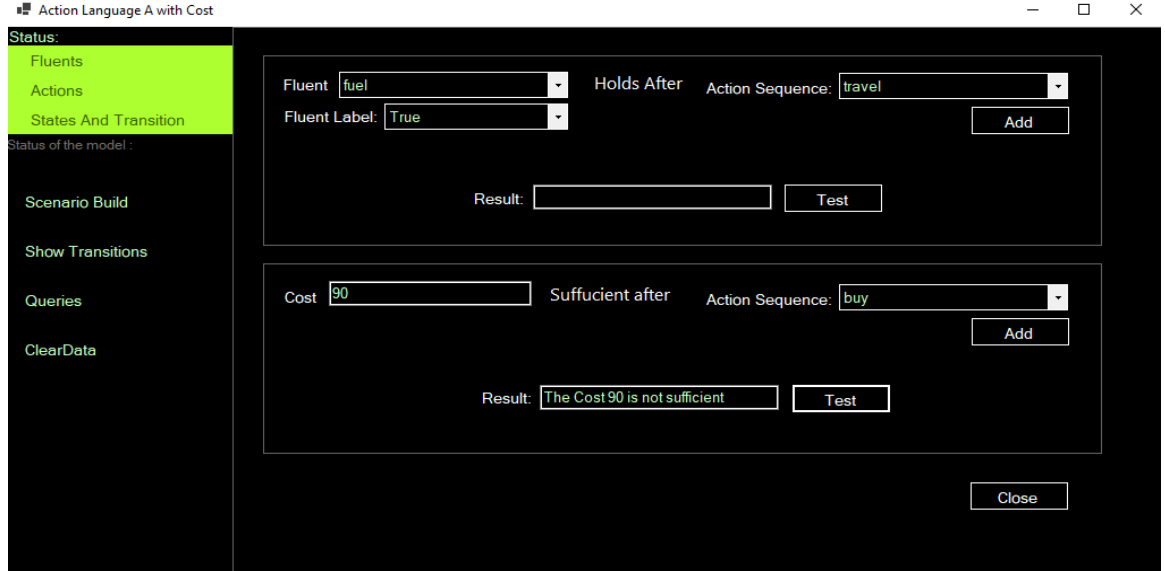


Figure 11: Queries(Cost)

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: $F = \{\text{available}, \text{sold}\}$

Actions: $Ac = \{\text{buy}, \text{order}\}$

Costs: $K = \{200, 50\}$

initially $\neg\text{available}$;

initially $\neg\text{sold}$;

buy causes sold if available;

buy causes $\neg\text{available}$;

buy costs 200\$;

order causes available if $\neg\text{available}$;

order costs 50\$;

2.1.3 Calculation:

$$\begin{aligned}\Sigma &= \{ \sigma_0, \sigma_1, \sigma_2, \sigma_3 \} \\ \sigma_0 &= \{ \neg \text{available}, \neg \text{sold} \} \\ \sigma_1 &= \{ \text{available}, \neg \text{sold} \} \\ \sigma_2 &= \{ \neg \text{available}, \text{sold} \} \\ \sigma_3 &= \{ \text{available}, \text{sold} \}\end{aligned}$$

$$\begin{aligned}\Psi(\text{buy}, \sigma_0) &= \sigma_0 \\ \Psi(\text{order}, \sigma_0) &= \sigma_1 \\ \Gamma(\text{buy}, \sigma_0) &= 0 \\ \Gamma(\text{order}, \sigma_0) &= 50\end{aligned}$$

$$\begin{aligned}\Psi(\text{buy}, \sigma_1) &= \sigma_2 \\ \Psi(\text{order}, \sigma_1) &= \sigma_1 \\ \Gamma(\text{buy}, \sigma_1) &= 200 \\ \Gamma(\text{order}, \sigma_1) &= 0\end{aligned}$$

$$\begin{aligned}\Psi(\text{buy}, \sigma_2) &= \sigma_2 \\ \Psi(\text{order}, \sigma_2) &= \sigma_3 \\ \Gamma(\text{buy}, \sigma_2) &= 0 \\ \Gamma(\text{order}, \sigma_2) &= 50\end{aligned}$$

$$\begin{aligned}\Psi(\text{buy}, \sigma_3) &= \sigma_2 \\ \Psi(\text{order}, \sigma_3) &= \sigma_3 \\ \Gamma(\text{buy}, \sigma_3) &= 200 \\ \Gamma(\text{order}, \sigma_3) &= 0\end{aligned}$$

2.1.4 Graph

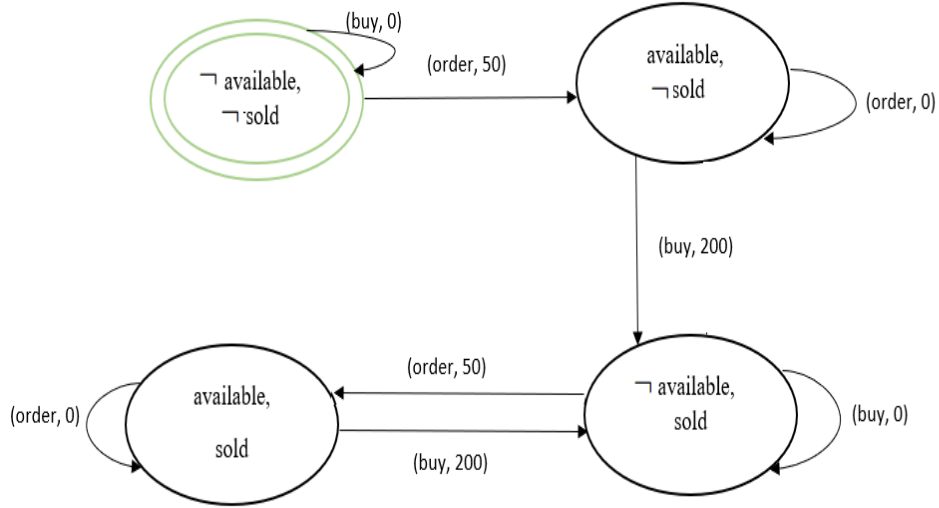


Figure 12: Example 02

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

The screenshot shows a web application window titled "Action Language A with Cost". The interface is divided into a left sidebar and a main content area. The sidebar contains the following links: "Status:", "Status of the model :", "Scenario Build", "Show Transitions", "Queries", and "ClearData". The main content area has a "Fluent Name" label followed by a text input field containing the placeholder "Type Fluent Name...". To the right of the input field is an "Add Fluent" button. Below the input field is a large text area containing the text "The Fluent available was added" and "The Fluent sold was added". At the bottom right of the main content area is a "Submit" button.

■ Action Language A with Cost

Status:

Status of the model :

Scenario Build

Show Transitions

Queries

ClearData

Fluent Name

Add Fluent

The Fluent available was added
The Fluent sold was added

Submit

Figure 13: Add Fluents

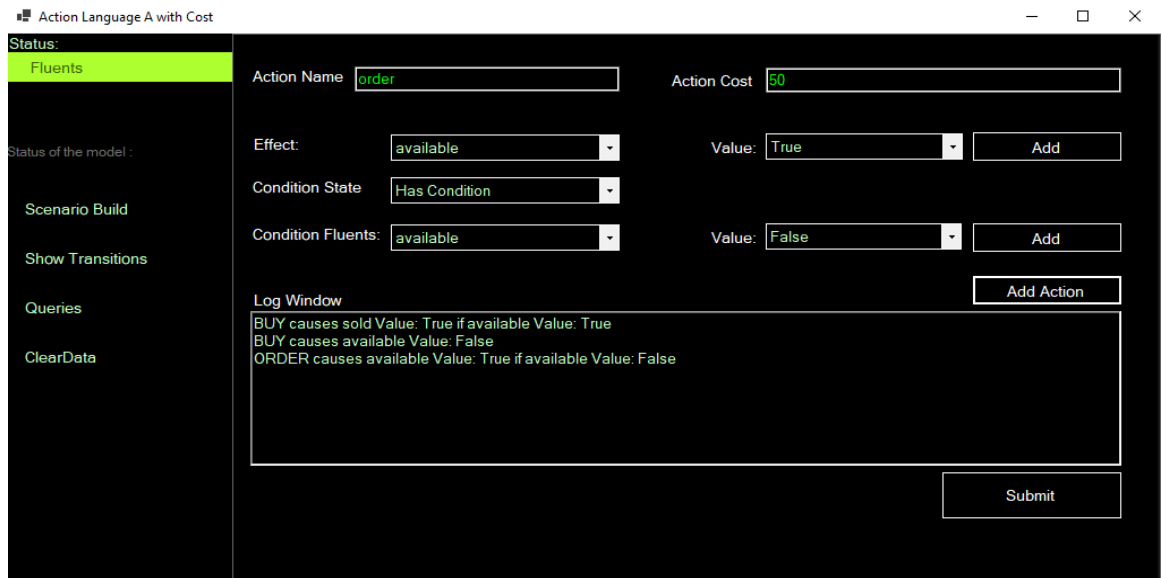


Figure 14: Add Actions

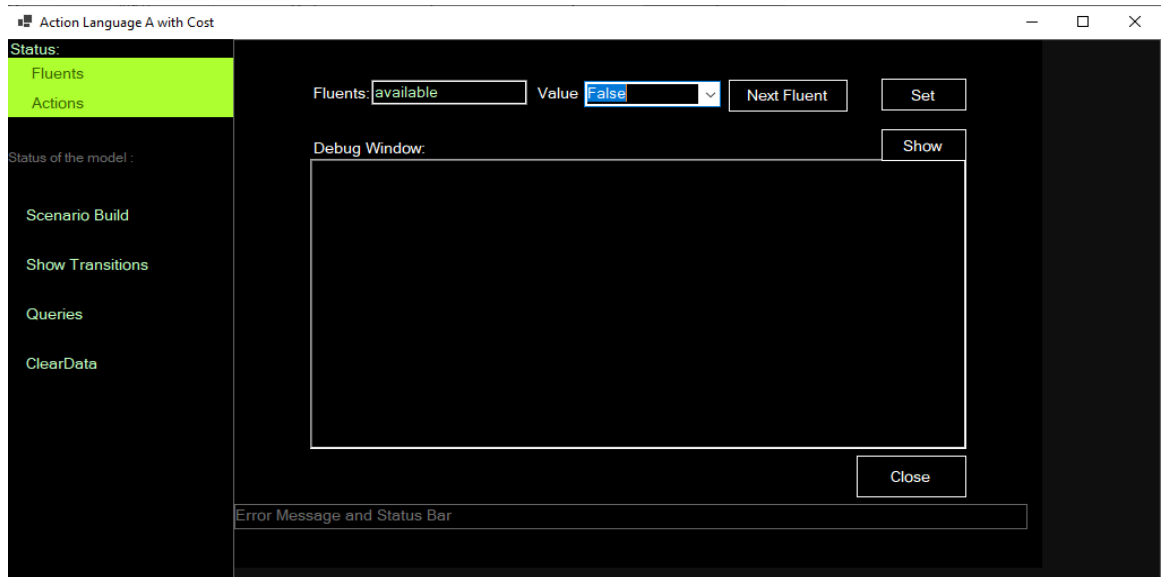


Figure 15: Set Initial State fluents

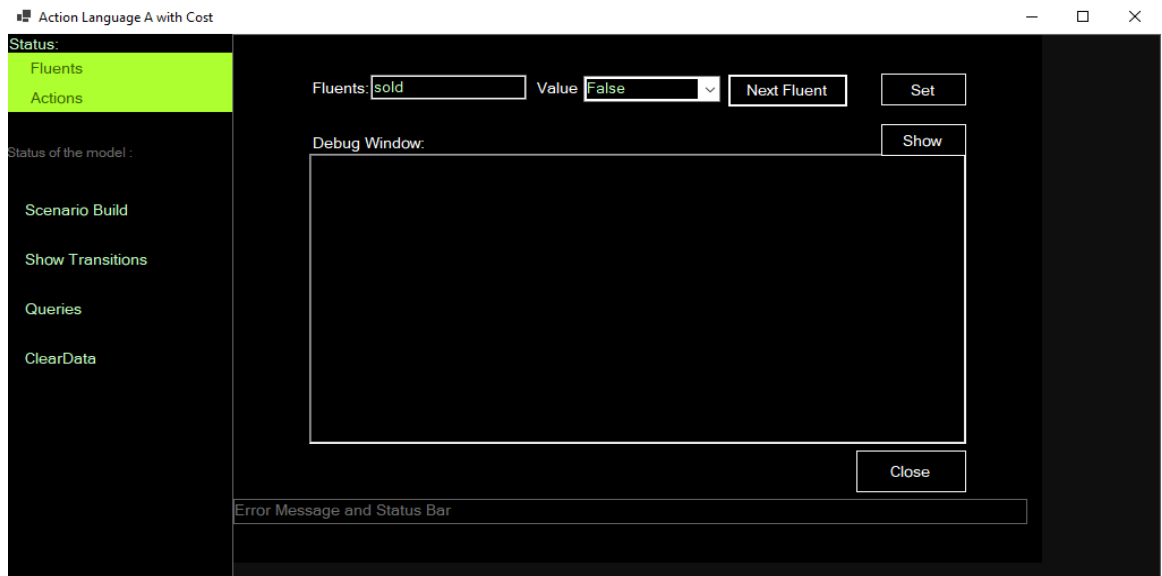


Figure 16: Set Initial state



Figure 17: Show Transitions

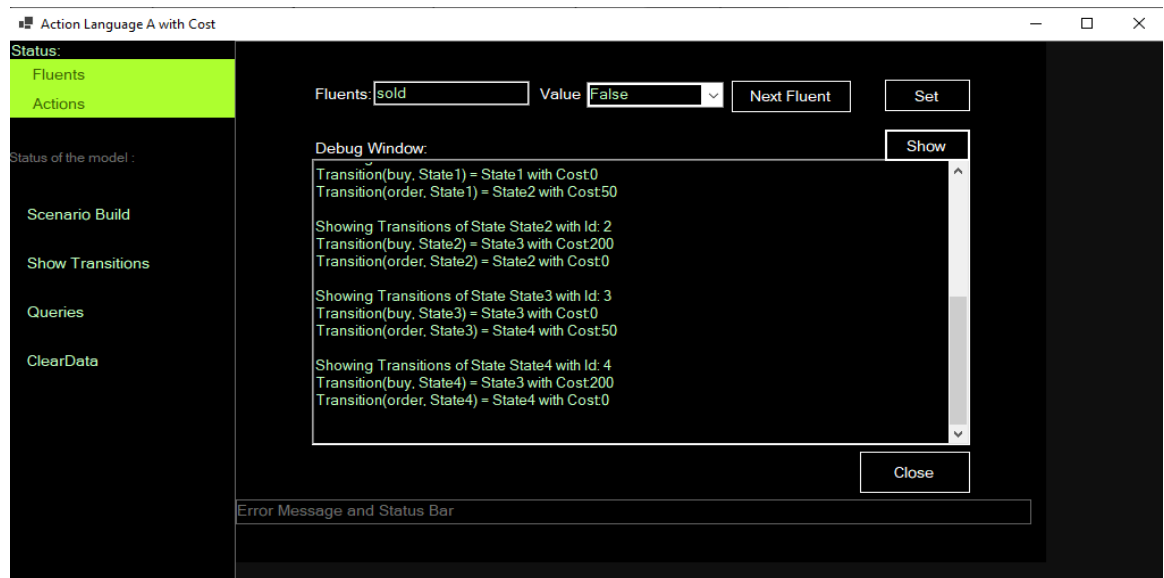


Figure 18: Show Transition contd..

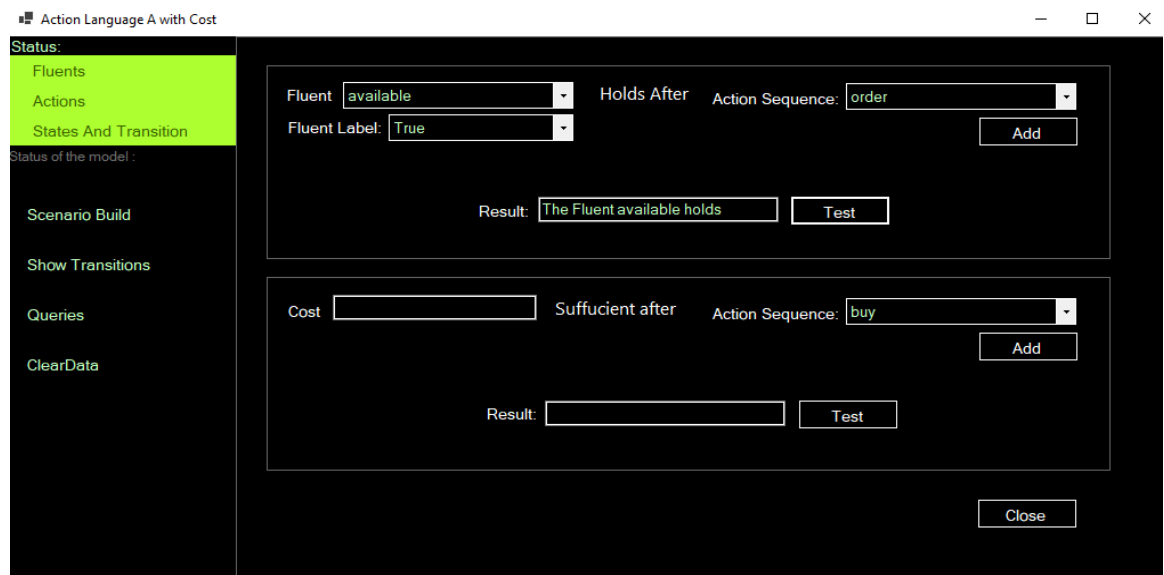


Figure 19: Queries(Fluents)

Action Language A with Cost

Status:

- Fluents
- Actions
- States And Transition

Status of the model :

Scenario Build

Show Transitions

Queries

ClearData

Fluent: Holds After Action Sequence:

Fluent Label:

Result:

Cost: Sufficent after Action Sequence:

Result:

Figure 20: Queries(Fluents)

Action Language A with Cost

Status:

- Fluents
- Actions
- States And Transition

Status of the model :

Scenario Build

Show Transitions

Queries

ClearData

Fluent: Holds After Action Sequence:

Fluent Label:

Result:

Cost: Sufficent after Action Sequence:

Result:

Figure 21: Queries(Cost)

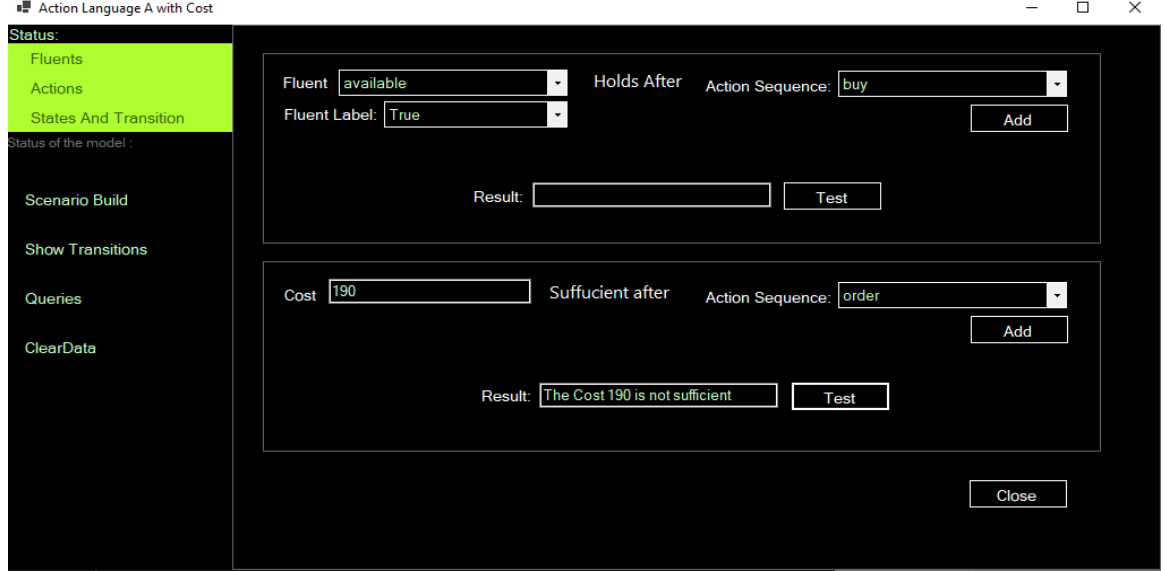


Figure 22: Queries(Cost)

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

initially $\neg\text{cooked};$
 initially hungry;

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

2.2.3 Calculation

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

$$\begin{aligned}
 \sigma_0 &= \{\neg\text{cooked}, \text{hungry}\} \\
 \sigma_1 &= \{\text{cooked}, \text{hungry}\} \\
 \sigma_2 &= \{\neg\text{cooked}, \neg\text{hungry}\} \\
 \sigma_3 &= \{\text{cooked}, \neg\text{hungry}\}
 \end{aligned}$$

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

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

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

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

2.2.4 Graph

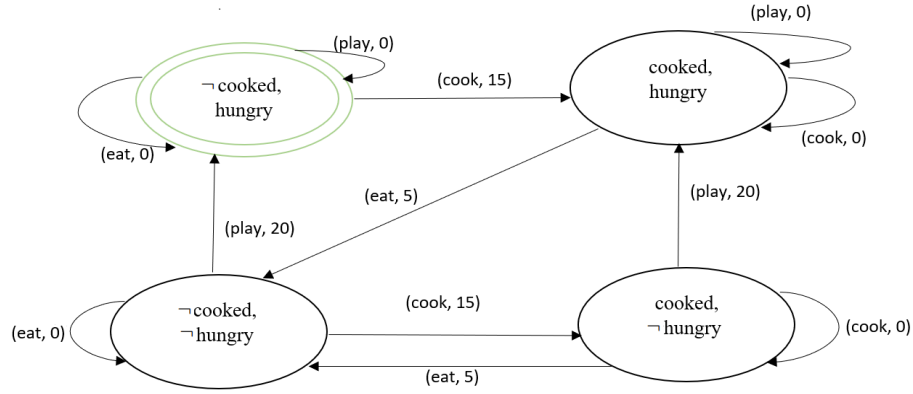


Figure 23: Example 03

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

Action Language A with Cost

Status:

Status of the model :

Scenario Build

Show Transitions

Queries

ClearData

Fluent Name

Add Fluent

The Fluent cooked was added

The Fluent hungry was added

Submit

Figure 24: Add Fluents

Action Name: Action Cost:

Effect: Value:

Condition State:

Condition Fluents: Value:

Log Window
 COOK causes cooked Value: True if cooked Value: False
 EAT causes cooked Value: False if cooked Value: True
 EAT causes hungry Value: False if cooked Value: True
 PLAY causes hungry Value: True if hungry Value: False

Figure 25: Add Actions

Fluents: Value:

Debug Window:

Error Message and Status Bar

Figure 26: Set Initial State fluents

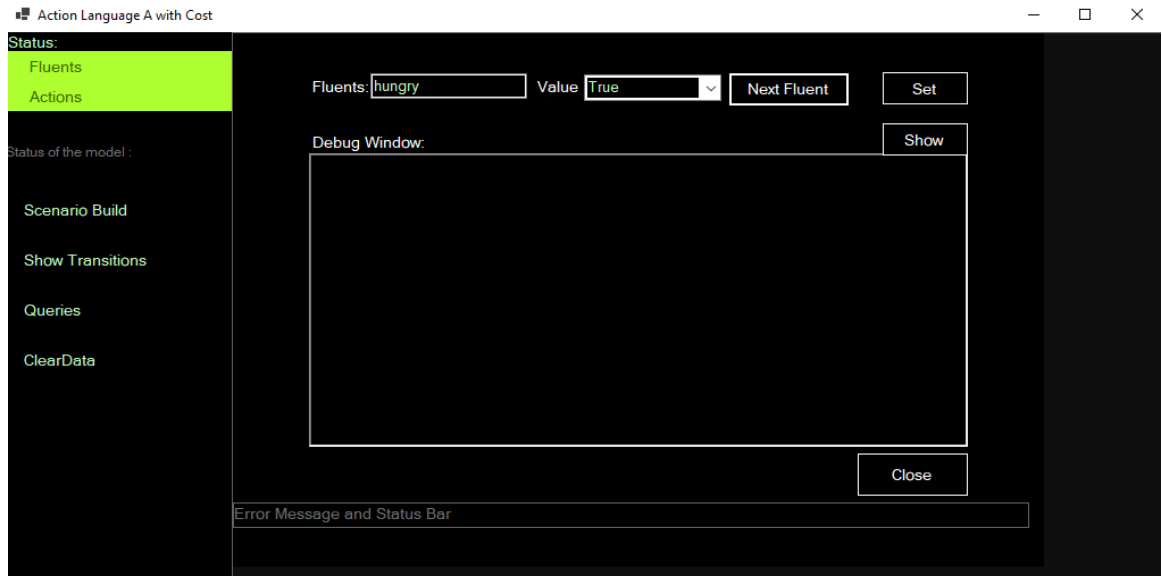


Figure 27: Set Initial state

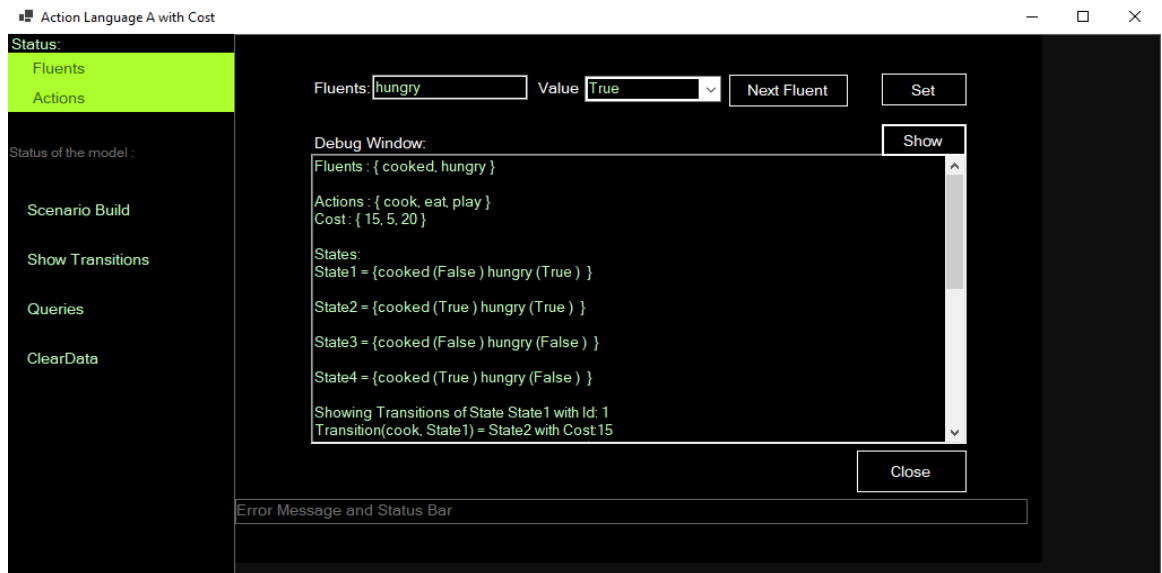


Figure 28: Show Transitions

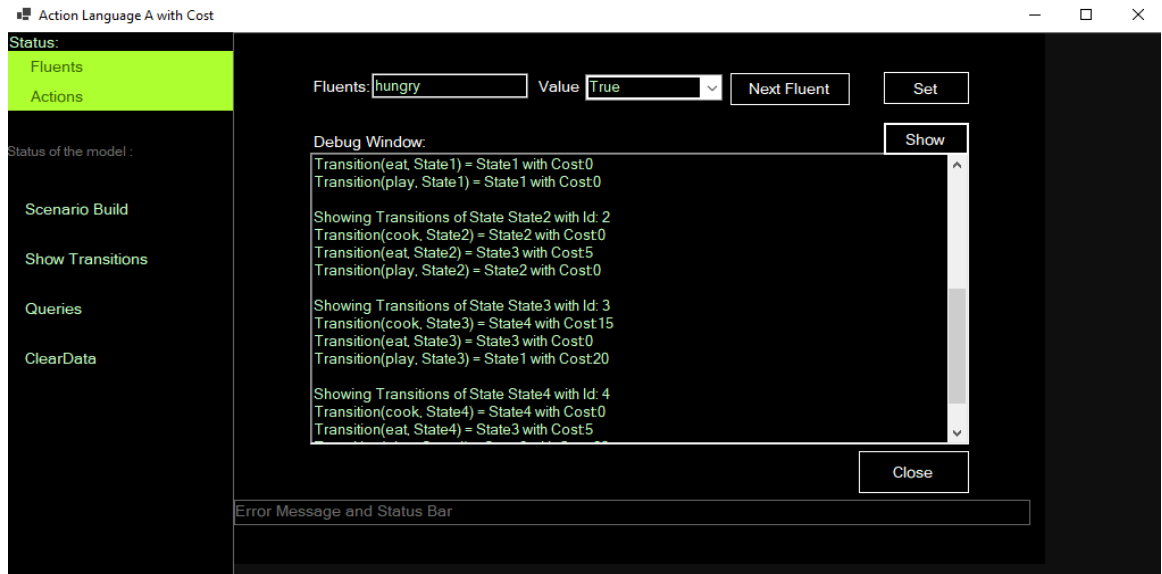


Figure 29: Show Transitions contd..

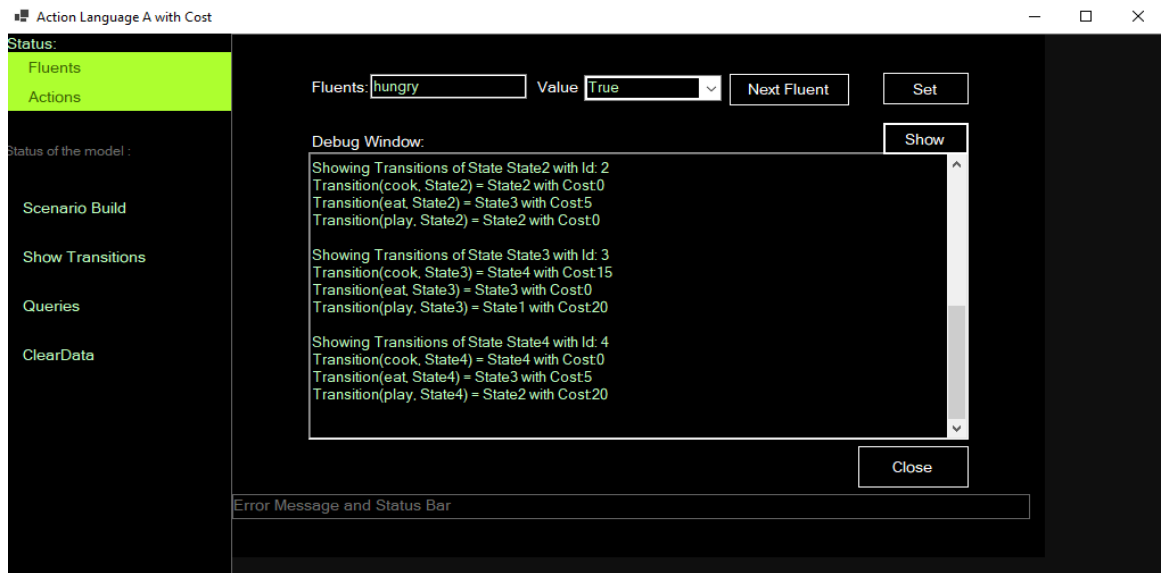


Figure 30: Show Transitions contd..)

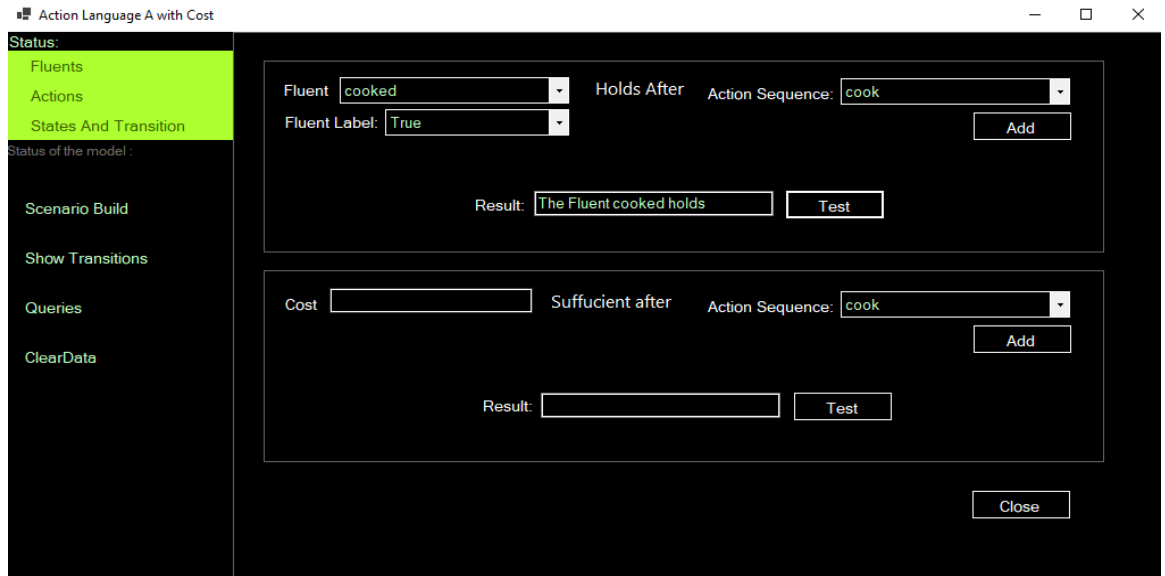


Figure 31: Queries(Fluents)

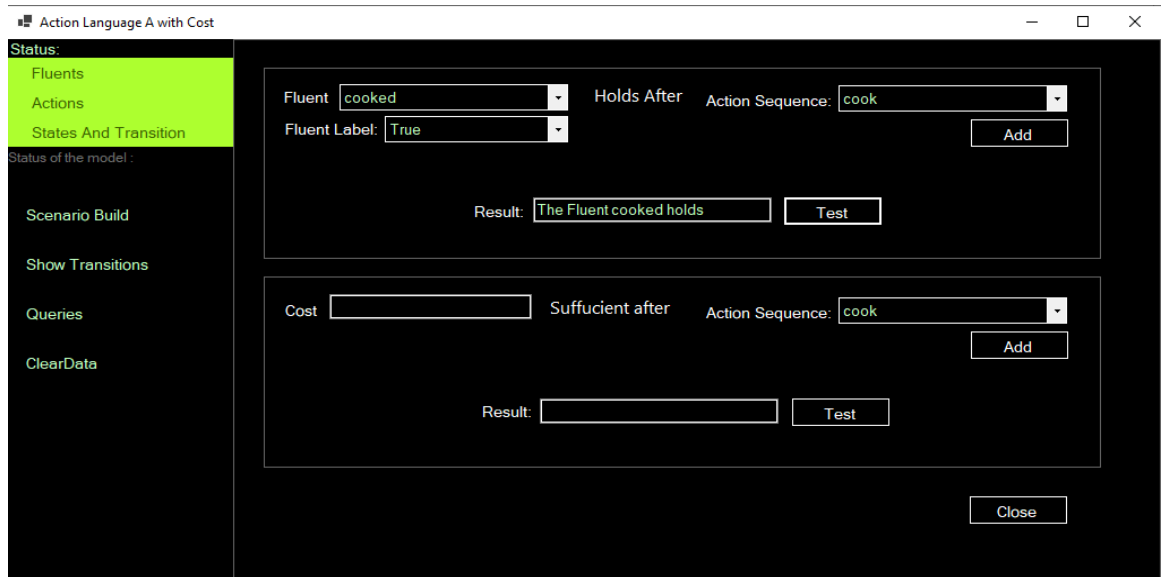


Figure 32: Queries(Fluents)

Action Language A with Cost

Status:

Fluents

Actions

States And Transition

Status of the model :

Scenario Build

Show Transitions

Queries

ClearData

Fluent:

Holds After

Action Sequence:

Fluent Label:

Add

Result:

Test

Cost:

Sufficient after

Action Sequence:

Add

Result:

Test

Close

Figure 33: Queries(Cost)

Action Language A with Cost

Status:

Fluents

Actions

States And Transition

Status of the model :

Scenario Build

Show Transitions

Queries

ClearData

Fluent:

Holds After

Action Sequence:

Fluent Label:

Add

Result:

Test

Cost:

Sufficient after

Action Sequence:

Add

Result:

Test

Close

Figure 34: Queries(Cost)

3 Appendix

List of Figures

1	Example 01	4
2	Add Fluents	5
3	Add Actions	6
4	Set Initial State fluents	6
5	Set Initial state	7
6	Show Transitions	7
7	Show Transition contd..	8
8	Queries(Fluents)	8
9	Queries(Fluents)	9
10	Queries(Cost)	9
11	Queries(Cost)	10
12	Example 02	12
13	Add Fluents	13
14	Add Actions	14
15	Set Initial State fluents	14
16	Set Initial state	15
17	Show Transitions	15
18	Show Transition contd..	16
19	Queries(Fluents)	16
20	Queries(Fluents)	17
21	Queries(Cost)	17
22	Queries(Cost)	18
23	Example 03	20
24	Add Fluents	21
25	Add Actions	22
26	Set Initial State fluents	22
27	Set Initial state	23
28	Show Transitions	23
29	Show Transition contd..	24
30	Queries(Fluents)	24
31	Queries(Fluents)	25
32	Queries(Cost)	25
33	Queries(Cost)	26
34	Queries(Cost)	26

List of Tables