Deterministic Action With Cost

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

Contents

1 Introduction

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 Scenario

A. Can a given program:

- always
- ever

be executed during at most cost units?
B. Does a given condition α hold

- always
- ever

after performing a given program in an initial state?
C. Is the Cost k enough to realize the given program?

3 Syntax

A system is defined by a set of fluent **F**, actions **Ac** and Cost **k** $\in$ **N** and characterized by signature **(F, Ac, k)**
A formula is any propositional combination of fluent:

$\alpha :: \neg\alpha \ |\alpha \wedge \beta \ |\alpha \vee \beta \ |\alpha \rightarrow \beta$

The system and changes occurring within can be described through a sequence of statements defined in the table:

Statement	**Format**	**Description**
Initial Statement	initially α	Initial condition α of the fluent
Effect Statement	Ac costing k causes α	Perform action Ac for cost k in any state leads to the effect α.
Release Statement	Ac costing k causes α	Perform action Ac for cost k in any state might, but need not, change the value of α.
Constraint Statement	Always α	Every state satisfies condition α.
Value Statement	α after A1....An	The condition α always (must) hold after performing the sequence A1 ....An of actions.
Observation Statement	observable α after A1 ....An	The condition α sometimes (may) holds after performing the sequence A1 ....An of actions.

Table 1: Sequence of Statements

4 Semantics

Let D be an action domain in the language L over the signature Υ=(F,A_c).
A structure S = (Ψ,σ0,k) is a model of D iff
(M1) for every s $\in$ D, S $|=$ s;
(M2) for every A $\in$ Ac, for every f,g1,...,gn $\in$ F, and for every $\sigma \in \sum$, if one of the following conditions holds:

(i) D contains an effect statement

A causes $\overline{f}$ for the cost k if $\overline{g}$,...,$\overline{gn}$,

where k $\neq$ 0 and $\sigma|\neq$ gi for some i = 1,...,n

(ii) D does not contain an effect statement

A causes $\overline{f}$ if g1,...,gn

then σ $|=$ f iff Ψ(A,σ,k) $|=$ f. where k = 0.

5 Examples

5.1 Example 01

5.1.1 Description

Andrew wants to travel by his car to a place.Travelling costs him 100$ if he uses fuel from the fuel tank of the car.If in case of emergency, Andrew is carrying a bottle of fuel as reserve, which can cost him 150$ for travelling because its a low quality fuel. Buying fuel costs him 200$ and reserve costs him 250$.

5.1.2 Representation

Initially we have:

1. Fuel
2. Reserve

Travel causes $\neg$fuel if fuel $\vee$ reserve
Travel causes $\neg$reserve if $\neg$fuel $\vee$ reserve
BuyF causes fuel if $\neg$fuel
BuyS causes reserve if $\neg$reserve

5.1.3 Calculation

$\sum = \{ \sigma_0, \sigma_1, \sigma_2, \sigma_3 \}$
$\sigma_0 = \{$ fuel, reserve $\}$ $\sigma_1 = \{ \neg$fuel, reserve $\}$
$\sigma_2 = \{ \neg$fuel, $\neg$reserve $\}$ $\sigma_3 = \{$ fuel, $\neg$reserve $\}$

Ψ(BuyF,σ_0)=σ_0 Ψ(BuyS,σ_0)=σ_0
Ψ(BuyF,σ_1)=σ_0 Ψ(BuyS,σ_1)=σ_1
Ψ(BuyF,σ_2)=σ_3 Ψ(BuyS,σ_2)=σ_1
Ψ(BuyF,σ_3)=σ_3 Ψ(BuyS,σ_3)=σ_0

Ψ(Travel,σ_0)=σ_1 Ψ(Travel,σ_1)=σ_2
Ψ(Travel,σ_2)=σ_2 Ψ(Travel,σ_3)=σ_2

5.1.4 Graph

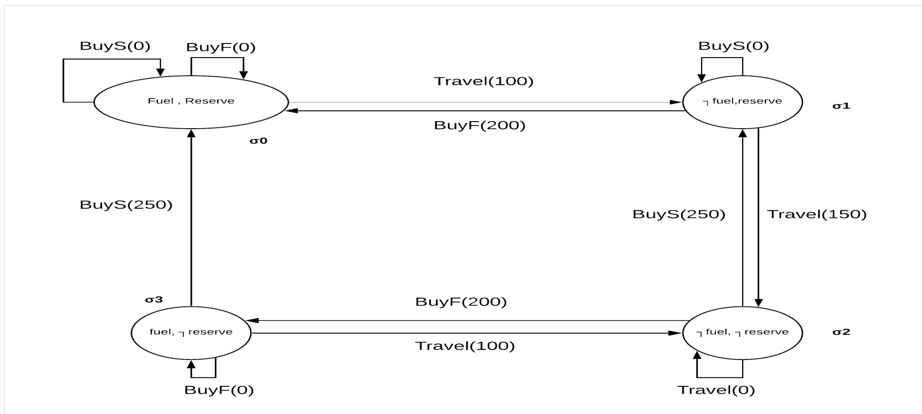

Figure 1: Example 01

5.2 Example 02

5.2.1 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 that will cost him 100$ extra. At any time only one copy of painting is available and another one to be ordered once sold.

5.2.2 Representation:

Fluents: available, sold.
Actions: BUY, ORDER.
BUY costs 200$ **if** available
BUY costs 300$ **after** ORDER
MIN COST:0$
MAX COST: 300$
Always ORDER $\rightarrow$ available
Always BUY $\rightarrow$ sold
initially: $\neg$available $\vee$ $\neg$sold
BUY causes sold if available
ORDER causes available if $\neg$available
$\neg$available after BUY

5.2.3 Calculation:

$\sum = \{ \sigma\ 0, \sigma\ 1, \sigma\ 2, \sigma\ 3\}$
$\sigma\ 0 = \{ \neg\text{available}, \neg\text{sold}\}$
$\sigma\ 1 = \{ \neg\text{available}, \text{sold}\}$
$\sigma\ 2 = \{ \text{available}, \neg\text{sold}\}$
$\sigma\ 3 = \{ \text{available}, \text{sold}\}$
Ψ (BUY, σ 0) = σ 0
Ψ (ORDER, σ 0) = σ 1
Ψ (BUY, σ 1) = σ 2
Ψ (ORDER, σ 1) = σ 1
Ψ (BUY, σ 2) = σ 2
Ψ (ORDER, σ 2) = σ 1
Ψ (BUY, σ 3) = σ 2
Ψ (ORDER, σ 3) = σ 3

5.2.4 Graph

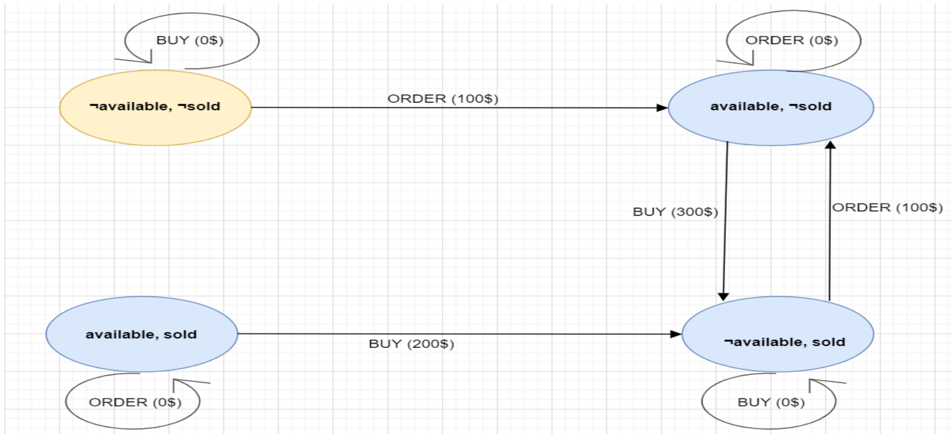

Figure 2: Example 02

5.3 Example 03

5.3.1 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, the is hungry, and no food is cooked. In terms of energy, eating costs 5, cooking costs 15, playing costs 20.

5.3.2 Representation in language

Fluents: cooked, hungry.
Actions: cook, eat, play.

eat costs 5
cooking cost 15
play cost 20
initially ¬cooked ∧ hungry
cook causes cook if ¬cooked

eat causes ($\neg$cooked $\wedge$ $\neg$hungry) if cooked
play causes hungry if $\neg$hungry

5.3.3 Calculation

$\sum = \{\sigma 0\ ,\sigma 1,\ \sigma 2,\ \sigma 3\}$

$\sigma 0 = \{\neg\text{cooked, hungry}\}$
$\sigma 1 = \{\text{cooked, hungry}\}$
$\sigma 2 = \{\neg\text{cooked}, \neg\text{hungry}\}$
$\sigma 3 = \{\text{cooked}, \neg\text{hungry}\}$

$\psi(\text{eat}, \sigma 0) = \sigma 0$
$\psi(\text{cook}, \sigma 0) = \sigma 1$
$\psi(\text{play}, \sigma 0) = \sigma 0$

$\psi(\text{eat}, \sigma 1) = \sigma 2$
$\psi(\text{cook}, \sigma 1) = \sigma 1$
$\psi(\text{play}, \sigma 1) = \sigma 1$

$\psi(\text{eat}, \sigma 2) = \sigma 2$
$\psi(\text{cook}, \sigma 2) = \sigma 3$
$\psi(\text{play}, \sigma 2) = \sigma 1$

$\psi(\text{eat}, \sigma 3) = \sigma 2$
$\psi(\text{cook}, \sigma 3) = \sigma 3$
$\psi(\text{play}, \sigma 3) = \sigma 1$

5.3.4 Graph

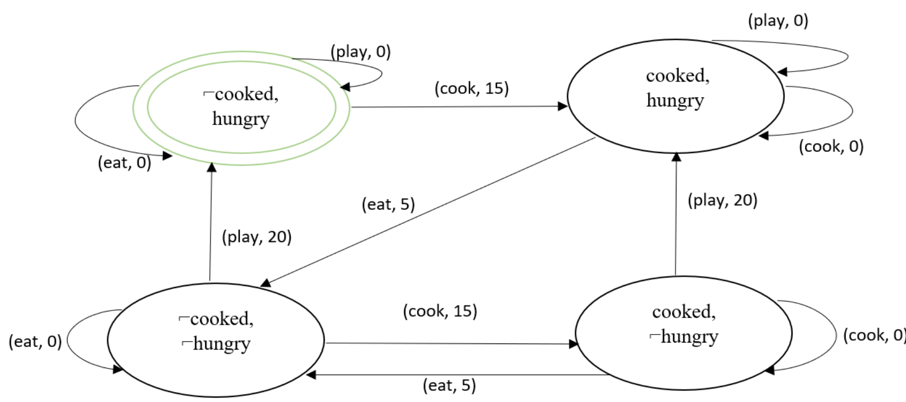

Figure 3: Example 03

6 Appendix

List of Figures

List of Tables