

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

CREATED BY
RISHABH JAIN, RAHUL TOMER, KULDEEP SHANKAR,
ALAA ABBOUSHI, HARAN DEV MURUGAN,
BUI TUAN ANH.

Contents

1	Introduction	2
2	Syntax	2
2.1	Signature :	2
2.2	Literal :	3
2.3	Statements :	3
3	Semantics	3
4	Examples	4
4.1	Example 01	4
4.1.1	Description	4
4.1.2	Representation	5
4.1.3	Calculation	5
4.1.4	Graph	6
4.2	Example 02	6
4.2.1	Description	6
4.2.2	Representation:	6
4.2.3	Calculation:	7
4.2.4	Graph	8
4.3	Example 03	8
4.3.1	Description	8
4.3.2	Representation in language	8
4.3.3	Calculation	9
4.3.4	Graph	10
5	Appendix	11

1 Introduction

A dynamic system (DS) is viewed as

- a collection of objects, together with their properties, and
- a collection of actions which, while performed, change properties of objects (in consequence, the state of the world).

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 Syntax

2.1 Signature :

A signature is a pair $\Upsilon = (F, Ac, k)$ where F is a set of fluents; Ac is a set of actions and Cost k .

2.2 Literal :

A literal is either a fluent f or its negation $\neg f$.

Notation: for a fluent $f \in F$, we write \bar{f} to denote the literal corresponding to f , i.e., either f or $\neg f$.

2.3 Statements :

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	A causes α if g_1, \dots, g_k	If the action A is performed in any state satisfying g_1, \dots, g_k , then in the resulting state α holds.
Value Statement	α after $A_1 \dots A_n$	The condition α always (must) hold after performing the sequence $A_1 \dots A_n$ of actions.
Cost Statement	A cost n	Perform the cost of action A

Table 1: Syntax Table

3 Semantics

- A state is a mapping $\sigma : F \rightarrow 0, 1$. For any $f \in F$, if $\sigma(f) = 1$, then we say that f holds in σ and write $\sigma \models f$. If $\sigma(f) = 0$, then we write $\sigma \models \neg f$ and say that f does not hold in σ . Let σ stand for the set of all states.
- A state is a mapping $\sigma : F \rightarrow 0, 1$. For any $f \in F$, if $\sigma(f) = 1$, then we say that f holds in σ and write $\sigma \models f$. If $\sigma(f) = 0$, then we write $\sigma \models \neg f$ and say that f does not hold in σ . Let σ stand for the set of all states.
- A transition function is a mapping $\Upsilon : Ac \times \sigma \rightarrow \sigma$. For any $\sigma \in \sigma$ and for any $A \in Ac$, $\Upsilon(A, \sigma)$ is the state resulting from performing the action A in the state σ .
- A transition function is generalized to the mapping $\Upsilon^* : Ac^* \times \sigma \rightarrow \sigma$ as follows: $\Upsilon^*(\varepsilon, \sigma) = \sigma$, $\Upsilon^*((A_1, \dots, A_n), \sigma) = \Upsilon(A_n, \Upsilon^*(A_1, \dots, A_{n-1}))$.

- A transition function is generalized to the mapping $\Upsilon^* : A_c * \times \sigma \rightarrow \sigma$ as follows: $\Upsilon^* (\varepsilon, \sigma) = \sigma$, $\Upsilon^* ((A1, \dots, An), \sigma) = \Upsilon(An, \Upsilon^*(A1, \dots, An-1))$.
- Let L be an action language of the class A over the signature $Y = (F, A_c)$. A structure for L is a pair $S = (\Upsilon, \sigma_0)$ where Υ is a transition function and $\sigma_0 \in \sigma$ is the initial state
- Let $s = (\Upsilon, \sigma_0)$ be a structure for L. A statement s is true in S, in symbols $S \models s$, iff if s is of the form f after $A1, \dots, An$, then $\Upsilon((A1, \dots, An), \sigma_0) \models f$;
- Let $s = (\Upsilon, \sigma_0)$ be a structure for L. A statement s is true in S, in symbols $S \models s$, iff if s is of the form f after $A1, \dots, An$, then $\Upsilon((A1, \dots, An), \sigma_0) \models f$; if s is of the form A causes f if $g1, \dots, gk$, then for every $\sigma \in \sigma$ such that $\sigma \models gi$, $i = 1, \dots, k$, $\Upsilon(A, \sigma) \models f$.

Let D be an action domain in the language L over the signature $\Upsilon = (F, A_c)$.

A structure $S = (\Psi, \sigma_0, k)$ is a model of D iff

(M1) for every $s \in D$, $S \models s$;

(M2) for every $A \in A_c$, for every $f, g1, \dots, gn \in F$, and for every $\sigma \in \Sigma$, if one of the following conditions holds:

- (i) D contains an effect statement
A causes \bar{f} for the cost k if $\bar{g}, \dots, \bar{g}n$,
 where $k \neq 0$ and $\sigma \not\models gi$ for some $i = 1, \dots, n$
- (ii) D does not contain an effect statement
A causes \bar{f} if $g1, \dots, gn$
 then $\sigma \models f$ iff $\Psi(A, \sigma, k) \models f$. where $k = 0$.

4 Examples

4.1 Example 01

4.1.1 Description

Andrew wants to travel by his car to a place. Travelling costs him 50\$ 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 50\$ for travelling. buying Fuel costs him 100\$ if both fuel and reserve are empty. and buying causes both his Fuel and Reserve are filled.

4.1.2 Representation

Initially we have:

1. Fuel
2. Reserve

Travel causes \neg fuel if fuel Travel causes \neg reserve if \neg fuel \vee reserve
Buy causes fuel, reserve if \neg fuel \vee reserve

4.1.3 Calculation

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

$$\begin{aligned} \sigma_0 &= \{ \text{fuel, reserve} \} & \sigma_1 &= \{ \neg\text{fuel, reserve} \} \\ \sigma_2 &= \{ \neg\text{fuel, } \neg\text{reserve} \} \end{aligned}$$

$$\begin{aligned} \Psi(\text{Buy}, \sigma_0) &= \sigma_0 & \Psi(\text{Travel}, \sigma_0) &= \sigma_1 \\ \Psi(\text{Buy}, \sigma_1) &= \sigma_1 & \Psi(\text{Travel}, \sigma_1) &= \sigma_2 \\ \Psi(\text{Buy}, \sigma_2) &= \sigma_1 & \Psi(\text{Travel}, \sigma_2) &= \sigma_2 \end{aligned}$$

4.1.4 Graph

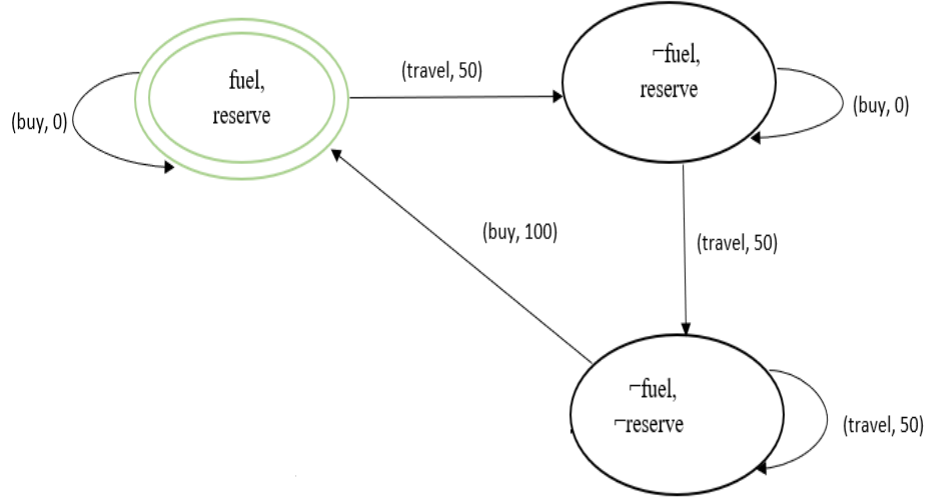


Figure 1: Example 01

4.2 Example 02

4.2.1 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 and will buy once its available. At any time only one copy of painting is available and another one to be ordered once sold.

4.2.2 Representation:

Fluents: available, sold.
Actions: BUY, ORDER.
BUY costs 200\$ **if** available
BUY costs 0\$ **if** \neg available
Order Cost: 0 \$
initially: \neg available \wedge \neg sold
BUY causes sold if available

ORDER causes available if \neg available
BUY causes \neg available

4.2.3 Calculation:

$$\begin{aligned}\Sigma &= \{ \sigma0, \sigma1, \sigma2, \sigma3 \} \\ \sigma0 &= \{ \neg\text{available}, \neg\text{sold} \} \\ \sigma1 &= \{ \neg\text{available}, \text{sold} \} \\ \sigma2 &= \{ \text{available}, \neg\text{sold} \} \\ \sigma3 &= \{ \text{available}, \text{sold} \} \\ \Psi(\text{BUY}, \sigma0) &= \sigma0 \\ \Psi(\text{ORDER}, \sigma0) &= \sigma1 \\ \Psi(\text{BUY}, \sigma1) &= \sigma2 \\ \Psi(\text{ORDER}, \sigma1) &= \sigma1 \\ \Psi(\text{BUY}, \sigma2) &= \sigma2 \\ \Psi(\text{ORDER}, \sigma2) &= \sigma1 \\ \Psi(\text{BUY}, \sigma3) &= \sigma2 \\ \Psi(\text{ORDER}, \sigma3) &= \sigma3\end{aligned}$$

4.2.4 Graph

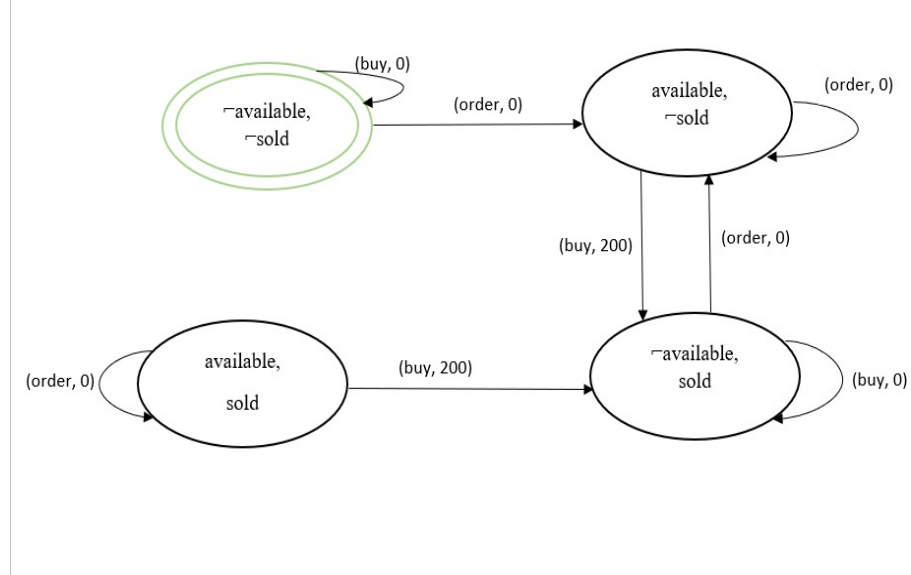


Figure 2: Example 02

4.3 Example 03

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

4.3.2 Representation in language

Fluents: cooked, hungry.

Actions: cook, eat, play.

eat cost 5

cooking cost 15
 play cost 20
 initially $\neg\text{cooked} \wedge \text{hungry}$
 cook causes cook if $\neg\text{cooked}$
 eat causes $(\neg\text{cooked} \wedge \neg\text{hungry})$ if cooked
 play causes hungry if $\neg\text{hungry}$

4.3.3 Calculation

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

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

$$\begin{aligned}\psi(\text{eat}, \sigma_0) &= \sigma_0 \\ \psi(\text{cook}, \sigma_0) &= \sigma_1 \\ \psi(\text{play}, \sigma_0) &= \sigma_0\end{aligned}$$

$$\begin{aligned}\psi(\text{eat}, \sigma_1) &= \sigma_2 \\ \psi(\text{cook}, \sigma_1) &= \sigma_1 \\ \psi(\text{play}, \sigma_1) &= \sigma_1\end{aligned}$$

$$\begin{aligned}\psi(\text{eat}, \sigma_2) &= \sigma_2 \\ \psi(\text{cook}, \sigma_2) &= \sigma_3 \\ \psi(\text{play}, \sigma_2) &= \sigma_1\end{aligned}$$

$$\begin{aligned}\psi(\text{eat}, \sigma_3) &= \sigma_2 \\ \psi(\text{cook}, \sigma_3) &= \sigma_3 \\ \psi(\text{play}, \sigma_3) &= \sigma_1\end{aligned}$$

4.3.4 Graph

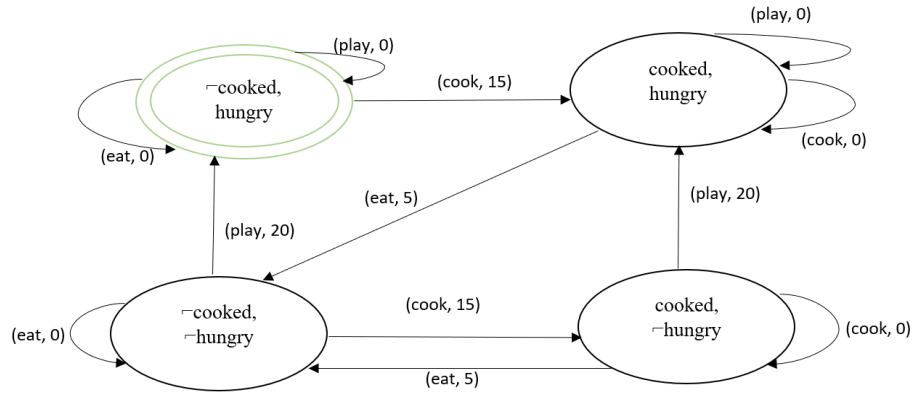


Figure 3: Example 03

5 Appendix

List of Figures

1	Example 01	6
2	Example 02	8
3	Example 03	10

List of Tables

1	Syntax Table	3
---	------------------------	---