

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	Query	5
4.1	Syntax	5
4.2	Semantics	5
5	Examples	6
5.1	Example 01	6
5.1.1	Description	6
5.1.2	Representation	6
5.1.3	Calculation	6
5.1.4	Graph	7
5.1.5	Queries	8
5.2	Example 02	8
5.2.1	Description	8
5.2.2	Representation:	8
5.2.3	Calculation:	8
5.2.4	Graph	10
5.2.5	Queries	10
5.3	Example 03	10
5.3.1	Description	10
5.3.2	Representation in language	11
5.3.3	Calculation	11
5.3.4	Graph	12
5.3.5	Queries	13
6	Appendix	14

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 triplet $\Upsilon = (F, Ac, K)$ where F is a set of fluents; Ac is a set of actions as follows A_1, A_2, \dots, A_n where $A_i \in Ac$ and $i = 1$ to n and K is a set of positive integers representing Cost of each action $A_i \in Ac$ as follows k_1, k_2, \dots, k_n where $k_i \in K$ and $i = 1$ to n .

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
Value Statement	\bar{f} after $A1 \dots An$ where $\bar{f} \in F$ and $Ai \in Ac$, for $i = 1, \dots, n$	\bar{f} holds after performing the sequence $A1 \dots An$ of actions in the initial state.
Abbreviation	initially \bar{f}	in the initial state \bar{f} holds
Effect Statement	Ai causes \bar{f} if $\overline{g1}, \dots, \overline{gk}$	If the action Ai is performed in any state satisfying $\overline{g1}, \dots, \overline{gk}$, then in the resulting state \bar{f} holds.
Cost Statement	A costs k	The action A , if performed with a non-empty effect, costs k

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 \sum stand for the set of all states.
- A state transition function is a mapping $\Psi : Ac \times \sum \rightarrow \sum$. For any $\sigma \in \sum$, for any action $Ai \in Ac$ where $i = 1, \dots, n$, $\Psi(Ai, \sigma)$ is the state resulting from performing the action Ai in the state σ . Also $\Psi(Ai, \sigma)$ will results in same state if the effect of action is empty.
- A program cost transition function is a mapping $\Gamma : A \rightarrow k$, where A is the sequence of actions $A1, A2, \dots, Ai$ performed on a state and k is the sum of corresponding costs of action and is defined as $k = k1 + k2 + \dots + ki$. For any action having an empty effect, the cost is 0.
- A transition function is generalized to the mapping $\Psi^* : Ac^* \times \sum \rightarrow \sum$ as follows:

1. $\Psi^* (\varepsilon, \sigma) = \sigma$,
 2. $\Psi^* ((A1, \dots, An), \sigma) = \Psi(An, \Psi^* (A1, \dots, An-1))$.
- A program cost transition function is generalized to the mapping $\Gamma^*: A^* \rightarrow k$ as follow:
 1. $\Gamma^*(\epsilon) = 0$ (empty program)
 2. $\Gamma^*((A1, \dots, Ai)) = \Gamma^*((A1, \dots, Ai-1)) + \Gamma(Ai, \Psi(A1, \dots, Ai), \sigma_0)$ (non-empty program)
 - Let L be an action language of the class A over the signature $\Upsilon = (F, Ac, K)$. A structure for L is a triplet $S = (\Psi, \sigma_0, \Gamma)$ where Ψ is a state transition function, Γ is a program cost transition function and $\sigma_0 \in \Sigma$ is the initial state
 - Let $S = (\Psi, \sigma_0, \Gamma)$ be a structure for L. A statement s is true in S, in symbols $S \models s$, iff
 1. s is of the form \bar{f} after $A1, \dots, An$, then $\Psi((A1, \dots, An), \sigma_0) \models \bar{f}$;
 2. if s is of the form Ai causes \bar{f} if $\bar{g1}, \dots, \bar{gk}$, then for every $\sigma \in \Sigma$ such that $\sigma \models \bar{gj}$, $j = 1, \dots, k$, $\Psi(Ai, \sigma) \models \bar{f}$.
 3. if s is of the form: A costs k where A is a sequence of actions $(A1, \dots, Ai)$ and k is sum of cost, $k = K1 + K2 + \dots + Ki$ respectively. $\Gamma(A) = k$

Let D be an action domain in the language L over the signature $\Upsilon = (F, Ac, K)$. A structure $S = (\Psi, \sigma_0, \Gamma)$ is a model of D iff

- (M1) for every statement $s \in D$, $S \models s$;
 (M2) for every $Ai \in Ac$ for every $f, g1, \dots, gn \in F$, for every $ki \in K$ and for every $\sigma \in \Sigma$, if one of the following conditions holds:

(i) D contains an effect statement and a cost statement as follows:

- **Ai causes \bar{f} if $\bar{g1}, \dots, \bar{gk}$, $\sigma \not\models \bar{gj}$ for some $j = 1, \dots, k$**
- A costs k if $\Psi(A, \sigma) \not\models \sigma$

(ii) D does not contain an effect statement but contains a 0 cost statement, as follows:

- **Ai causes \bar{f} if $\bar{g1}, \dots, \bar{gk}$ then $\sigma \models f$ iff $\Psi(Ai, \sigma) \models f$.**
- $\Gamma(A) = 0$ if $\Psi(A, \sigma) \models \sigma$.

4 Query

4.1 Syntax

- Query 1: Q1 is a query defined to determine whether a given condition holds after executing a program or not:

$$\bar{f} \text{ holds after } A$$

where \bar{f} is the given condition and A is the sequence of actions performed.

- Query 2: Q2 is a query determined from model S of the action domain D to check whether a given cost k is sufficient to execute a program or not.

$$k \text{ sufficient for } A$$

where A is the sequence of actions performed and k is the sum of cost of respective actions.

Query	Format	Description
Q1	\bar{f} holds after A	whether \bar{f} holds after A where A is the sequence of actions performed
Q2	k sufficient for A	Whether k is sufficient to execute the program A, where A is a sequence of actions (A1,...,Ai) and k is sum of costs, $k = K1 + K2 + \dots + Ki$ respectively

Table 2: Syntax Table

4.2 Semantics

- Query 1 is defined as Q1: $(A, \bar{f}) \rightarrow \text{True/False}$, where A is the sequence of actions performed and \bar{f} is the condition to check. If the given condition holds after performing A, then the result of Q1 is True. If the given condition does not hold after performing A, the result of Q1 is False.
- Query 2 is defined as Q2: $(A, k) \rightarrow \text{True/False}$, where A is the sequence of actions performed and k is the sum of cost of respective actions. For every model S of the action domain D and for the mapping Γ^* defined above, iff $\Gamma^*((A1, \dots, An)) \geq k$, then result of Q2 is True, else is False

5 Examples

5.1 Example 01

5.1.1 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 buy fuel. Fuel costs him 40\$

5.1.2 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;

5.1.3 Calculation

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

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

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

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

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

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

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

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

$\Psi(\text{travel}, \sigma_1) = \sigma_1$
 $\Gamma(\text{buy}, \sigma_1) = 40$
 $\Gamma(\text{travel}, \sigma_1) = 0$

$\Psi(\text{buy}, \sigma_2) = \sigma_3$
 $\Psi(\text{travel}, \sigma_2) = \sigma_1$
 $\Gamma(\text{buy}, \sigma_2) = 40$
 $\Gamma(\text{travel}, \sigma_2) = 50$

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

5.1.4 Graph

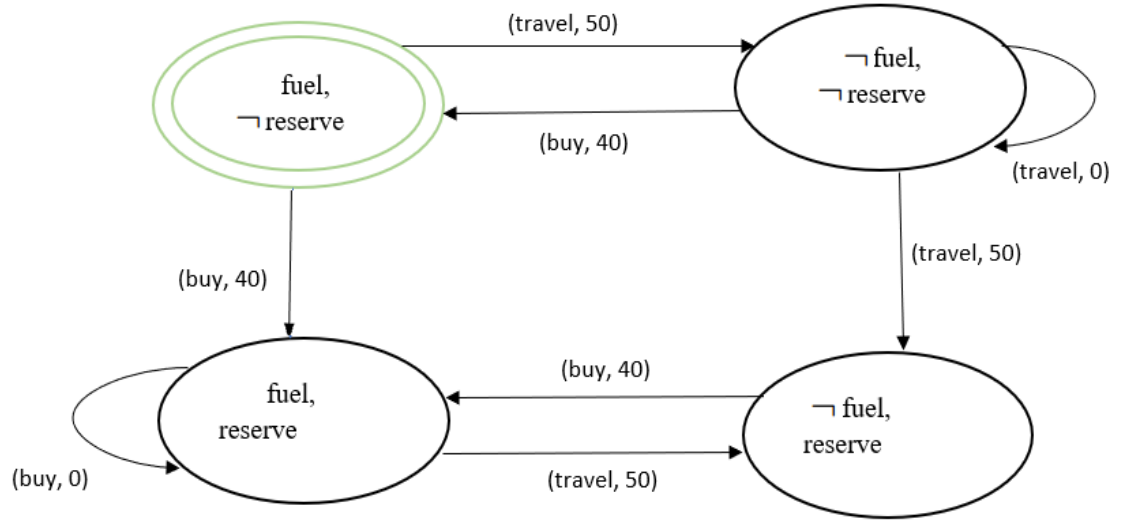


Figure 1: Example 01

5.1.5 Queries

σ_1 holds after (travel, travel, buy, travel): True

σ_2 holds after (buy, travel, buy): False

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

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

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 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.

5.2.2 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\$;

5.2.3 Calculation:

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

$\sigma_0 = \{\neg\text{available}, \neg\text{sold}\}$

$\sigma_1 = \{\text{available}, \neg\text{sold}\}$

$\sigma_2 = \{\neg\text{available}, \text{sold}\}$

$\sigma_3 = \{\text{available}, \text{sold}\}$

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

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

$$\begin{aligned}\Psi(\text{buy}, \sigma_2) &= \sigma_2 \\ \Psi(\text{order}, \sigma_2) &= \sigma_1 \\ \Gamma(\text{buy}, \sigma_2) &= 0 \\ \Gamma(\text{order}, \sigma_2) &= 50\end{aligned}$$

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

5.2.4 Graph

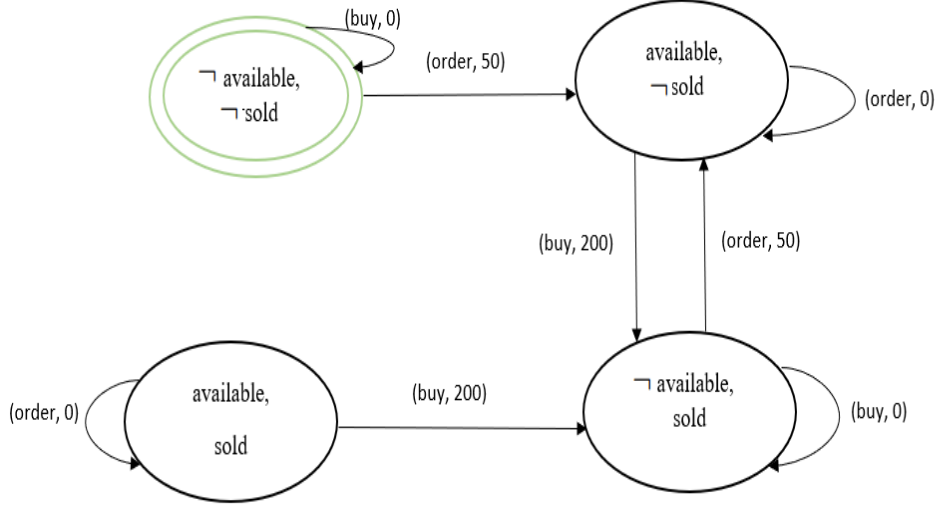


Figure 2: Example 02

5.2.5 Queries

σ_1 holds after (buy, order, buy, order): True

σ_3 holds after (buy, order, buy): False

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

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

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, he 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: $F = \{\text{cooked}, \text{hungry}\}$

Actions: $Ac = \{\text{cook}, \text{eat}, \text{play}\}$

Costs: $K = \{15, 5, 20\}$

initially $\neg\text{cooked}$;

initially hungry;

cook causes cooked if $\neg\text{cooked}$;

cook costs 15;

eat causes $\neg\text{cooked}$ if cooked;

eat causes $\neg\text{hungry}$ if cooked;

eat costs 5;

play causes hungry if $\neg\text{hungry}$;

play costs 20;

5.3.3 Calculation

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

$\sigma_0 = \{\neg\text{cooked}, \text{hungry}\}$

$\sigma_1 = \{\text{cooked}, \text{hungry}\}$

$\sigma_2 = \{\neg\text{cooked}, \neg\text{hungry}\}$

$\sigma_3 = \{\text{cooked}, \neg\text{hungry}\}$

$\Psi(\text{eat}, \sigma_0) = \sigma_0$

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

$\Psi(\text{play}, \sigma_0) = \sigma_0$

$\Gamma(\text{eat}, \sigma_0) = 0$

$\Gamma(\text{cook}, \sigma_0) = 15$

$\Gamma(\text{play}, \sigma_0) = 0$

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

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

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

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

$$\begin{aligned}\Psi(\text{eat}, \sigma_2) &= \sigma_2 \\ \Psi(\text{cook}, \sigma_2) &= \sigma_3 \\ \Psi(\text{play}, \sigma_2) &= \sigma_1 \\ \Gamma(\text{eat}, \sigma_2) &= 0 \\ \Gamma(\text{cook}, \sigma_2) &= 15 \\ \Gamma(\text{play}, \sigma_2) &= 20\end{aligned}$$

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

5.3.4 Graph

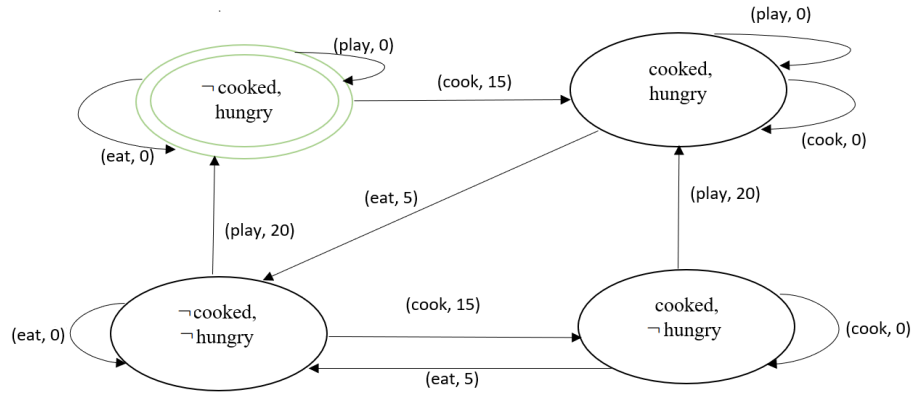


Figure 3: Example 03

5.3.5 Queries

σ_1 holds after (play, play, eat, cook): True

σ_0 holds after (cook, eat play, cook): False

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

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

6 Appendix

List of Figures

1	Example 01	7
2	Example 02	10
3	Example 03	12

List of Tables

1	Syntax Table	3
2	Syntax Table	5