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

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 Υ = (F, Ac, K) where F is a set of fluents; Ac is a set of actions as follows A1,A2,...,An where Ai $\in$ Ac and i = 1 to n and K is a set of positive integers representing Cost of each action Ai $\in$ Ac as follows k1,k2,...kn where ki $\in$ K and i = 1 to n.

2.2 Literal :

A literal is either a fluent f or its negation ¬f.
Notation: for a fluent f ϵ F, we write $\overline{f}$ to denote the literal corresponding to f, i.e., either f or ¬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	$\overline{f}$ after A1 ....An where $\overline{f}$ $\in$ F and Ai $\in$ Ac, for i = 1, ... ,n	$\overline{f}$ holds after performing the sequence A1 ....An of actions in the initial state.
Abbreviation	initially $\overline{f}$	in the initial state $\overline{f}$ holds
Effect Statement	A causes $\overline{f}$ if $\overline{g1}$, . . . , $\overline{gk}$	If the action A is performed in any state satisfying $\overline{g1}$, . . . , $\overline{gk}$, then in the resulting state $\overline{f}$ 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 σ : F $\rightarrow$ {0, 1}. For any f $\in$ F, if σ(f) = 1, then we say that f holds in σ and write $\sigma \vDash$ f. If σ(f) = 0, then we write σ $\vDash$ ¬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 Ψ : Ac $\times \sum \rightarrow \sum$. For any $\sigma \in \sum$, for any action A $\in$ Ac, Ψ(A , σ) is the state resulting from performing the action A in the state σ. Also Ψ(A,σ) will results in same state if the effect of action is empty.

- A cost transition function is a mapping Γ : A $\rightarrow$ k, where A is the action performed with a non-empty effect and k is the cost of the action. For any action having an empty effect, the cost is 0.

- A state transition function is generalized to the mapping Ψ^* : Ac* $\times$ $\sum \rightarrow \sum$ as follows:

1. Ψ^* (ε , σ) = σ,
2. Ψ^* ((A1, . . . , An), σ) = Ψ(An, Ψ^* (A1, . . . , An-1)).

- A program cost transition function is generalized to the mapping Γ^*:A* $\rightarrow$k as follow:

  1. $\Gamma^*(\epsilon) = 0$ (empty program)
  2. Γ^*((A1,...,Ai)) = Γ^*((A1,...,Ai-1)) + Γ(Ai,Ψ(A1,...,Ai),σ_0) (non-empty program)

- Let L be an action language of the class A over the signature Υ = (F, Ac, K). A structure for L is a triplet S = (Ψ, σ_0, Γ) where Ψ is a state transition function, Γ is a cost transition function and $\sigma_0 \in \sum$ is the initial state

- Let S = (Ψ, σ_0, Γ) be a structure for L. A statement s is true in S, in symbols S $\models$ s, iff

  1. s is of the form $\overline{f}$ after A1, . . . , An, then Ψ((A1, . . . , An), σ_0)) $\models \overline{f}$;
  2. if s is of the form A causes $\overline{f}$ if $\overline{g1}$, . . . , $\overline{gk}$, then for every $\sigma \in \sum$ such that $\sigma \models \overline{gi}$, i = 1, . . . , k, Ψ(A, σ) $\models \overline{f}$.
  3. if s is of the form: A costs k where A is the action, if performed with a non empty effect, the cost is k, Γ(A) = k

Let D be an action domain in the language L over the signature Υ=(F,Ac,K). A structure S = (Ψ,σ_0, Γ) is a model of D iff
(M1) for every statement s $\in$ D, S $\models$ s;
(M2) for every A $\in$ Ac for every f,g1,...,gn $\in$ F, for every k $\in$ K and for every $\sigma \in \sum$, if one of the following conditions holds:
(i) D contains an effect statement and a cost statement as follows:

- **A causes** $\overline{f}$ **if** $\overline{g1}$,...,$\overline{gk}$, and $\sigma \not\models \overline{gj}$ for some j = 1,...,k
- Γ(A) = k iff Ψ(A, σ) $\not\models \sigma$

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

- **A causes** $\overline{f}$ **if** $\overline{g1}$,...,$\overline{gk}$ then $\sigma \models$ f iff Ψ(A,σ) $\models$ f.
- Γ(A)=0 iff Ψ(A, σ) $\models \sigma$.

4 Query

4.1 Syntax

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

$$\overline{f} \text{ holds after P}$$

  where $\overline{f}$ is the given condition and program is the sequence P=(A1, A2,..., An), n $\geq$ 0, of actions.

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

$$\text{C sufficient for P}$$

  where C is the cost which needs to be checked and program is the sequence P=(A1, A2,..., An), n $\geq$ 0, of actions

Query	**Format**	**Description**
Q1	$\overline{f}$ holds after A	whether $\overline{f}$ holds after executing the program P where P=(A1, A2,..., An), n $\geq$ 0, is a sequence of actions.
Q2	C sufficient for P	Cost C is sufficient to execute the program P=(A1,..., An), n $\geq$ 0, is a sequence of actions

Table 2: Syntax Table

4.2 Semantics

- Query 1 is defined as Q1: (P,$\overline{f}$) $\rightarrow$ True/False, where P is the program P=(A1, A2, ..., An), n $\geq$ 0, of actions and $\overline{f}$ is the condition to check.

  Q1 $\rightarrow$ True, iff Ψ^*((A1,..., An), σ) $\models \sigma$

  Q1 $\rightarrow$ False, iff Ψ^*((A1,..., An), σ) $\not\models \sigma$

- Query 2 is defined as Q2: (P,C) $\rightarrow$ True/False, where P is the program P=(A1, A2,..., An), n $\geq$ 0, of actions and C is the cost to be checked. For every model S of the action domain D and for the mapping Γ^* defined above:

Q2 $\rightarrow$ True, iff $\Gamma^*((A1,..., An)) \leq C$

Q2 $\rightarrow$ False, iff $\Gamma^*((A1,..., An)) \geq C$

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 fuy fuel. Fuel costs him 40$

5.1.2 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;

5.1.3 Calculation

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

$\sigma_0 = \{$ fuel, ¬reserve $\}$ $\quad$ $\sigma_1 = \{$ ¬fuel, ¬reserve $\}$
$\sigma_2 = \{$ ¬fuel, reserve $\}$ $\quad$ $\sigma_3 = \{$ fuel,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

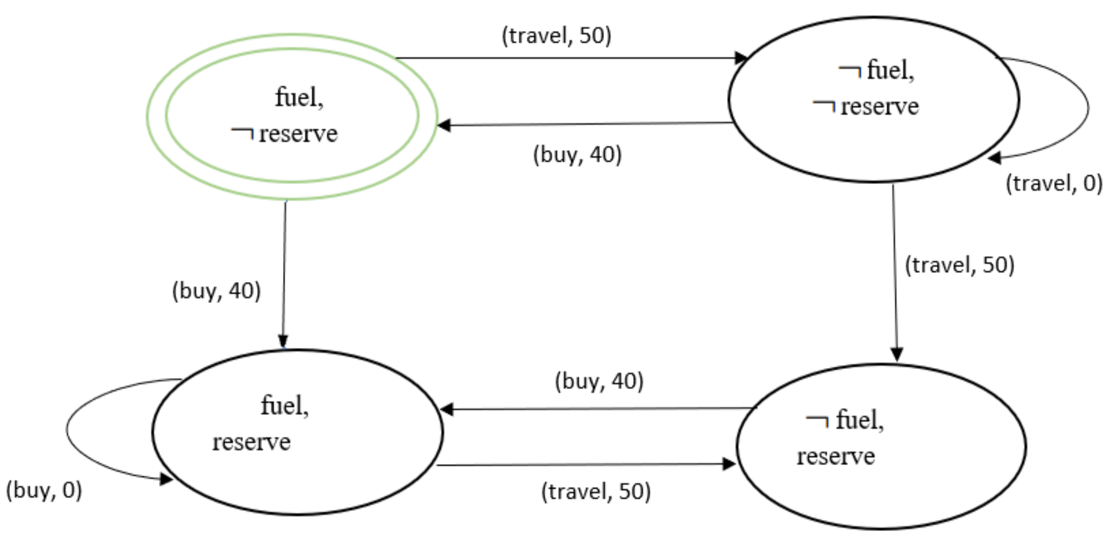

Figure 1: Example 01

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 = {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$;
order causes available if ¬availabl;
order costs 50$;

5.2.3 Calculation:

$\sum = \{\ \sigma_0, \sigma_1, \sigma_2, \sigma_3\}$
$\sigma_0 = \{$ ¬available, ¬sold$\}$
$\sigma_1 = \{$ available, ¬sold$\}$
$\sigma_2 = \{$ ¬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, $\sigma 2$) = $\sigma 1$
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

5.2.4 Graph

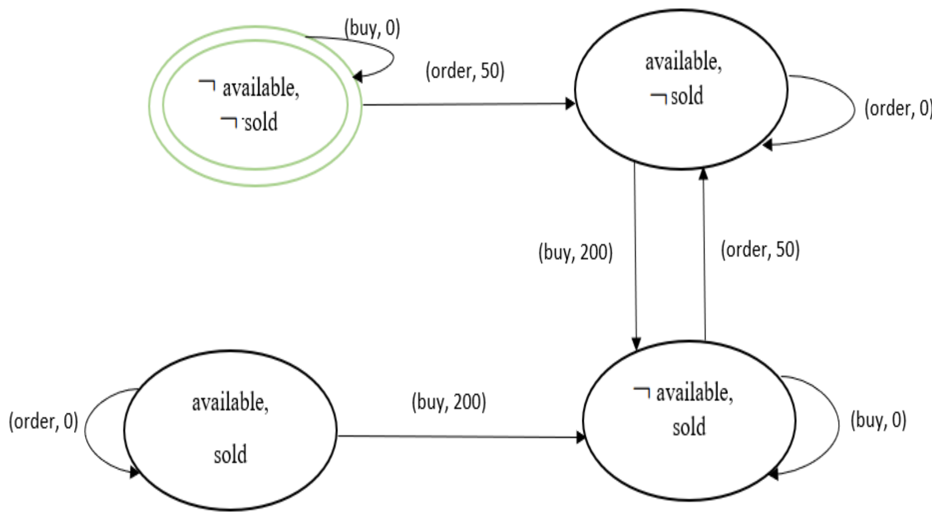

Figure 2: Example 02

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 = {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;

5.3.3 Calculation

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

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

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

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

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

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

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

5.3.4 Graph

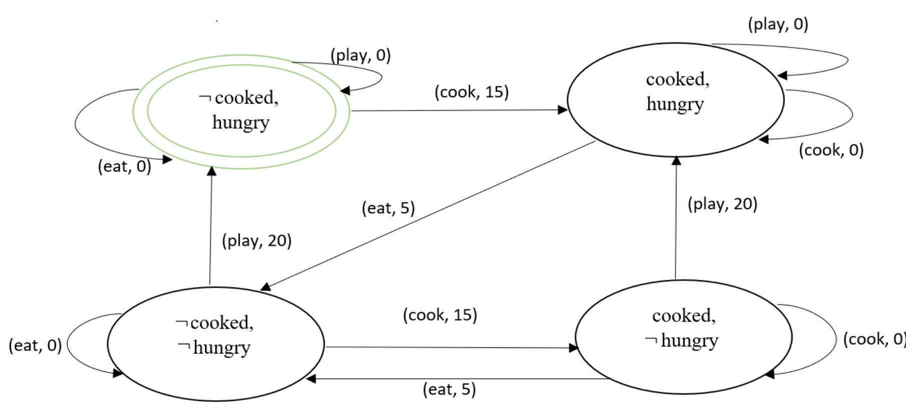

Figure 3: Example 03

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

List of Tables