

1. Introduction

The Finite Automaton (FA), also known as Finite State Machine (FSM), is a broadly applied mathematical model for analysing the behavior of systems that are commonly called discrete event systems or discrete dynamic systems. The FA is perfectly suited for the analysis of the *statics* of a system, i.e. its state space (the set of allowable states) and its transition space (the set of allowable transitions). It falls short, however, if one also wants to investigate the kinematics and the dynamics of a system. By *kinematics* is understood the time dimension in the transition space. Put differently, it is about the *process* that evolves in the course of time. By *dynamics* is understood the mechanism that causes transitions to take place.

The δ -theory (the Greek letter “ δ ” is pronounced as “DELTA”, which stands for “Discrete Event in Linear Time Automaton”) is a theory about the the statics, the kinematics, and the dynamics of discrete event systems. It constitutes a.o. the theoretical foundation for the formalisation of the π -theory and the ψ -theory. The δ -theory is rooted in automata theory (e.g. [Hopcroft, Ullman 1979]).

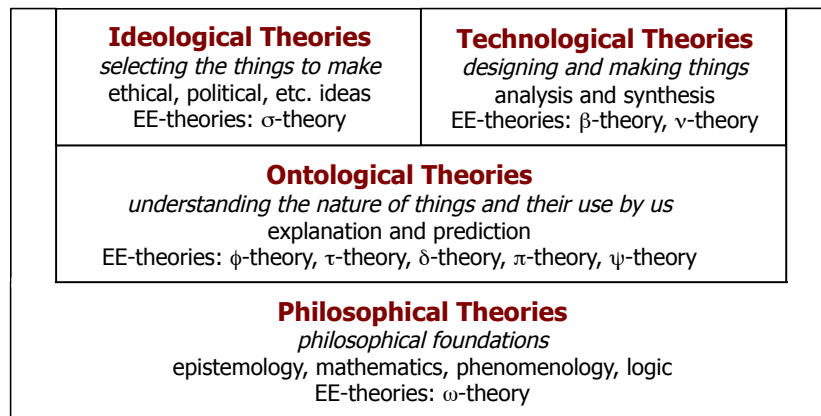


Figure 1.1 Classification scheme for enterprise engineering theories

This extended summary presents and discusses the main subjects and outcomes of the δ -theory, including its relationship with other enterprise engineering theories, as exhibited in the classification scheme in figure 1.1, taken from [Dietz, Hoogervorst et al. 2013] (Note: in this scheme, an arrow from A to B means that A is a basis of B). The scheme exhibits that the δ -theory is a *philosophical* theory, which means that it concerns basic and generic theoretical topics. The next core system related notions are clarified by the δ -theory: state, action, event, process. From other EE-theories, the next notions are taken: system (τ -theory), model (τ -theory).

2. Finite automata

A finite automaton (FA) is a mathematical model of a system with discrete inputs and outputs [Hopcroft, Ullman, 1979]. It consists of a finite set of states (which represent internal configurations of the modelled system), and a finite set of state transitions. Transitions occur on an input from a set of possible inputs. Usually there is an initial state and there are one or more final states. FAs are mostly associated with the way they are commonly represented, namely the state transition diagram (STD). Figure 2.1 exhibits the STD of the FA that models the well known man-wolf-goat-cabbage problem. A man with a wolf, a goat, and a cabbage, is on one bank of a river. They all have to be brought to the other bank. There is a boat that can carry the man and each of the other three. However, if the wolf and the goat are left unattended, the wolf will eat the goat. Similarly, if the goat and the cabbage are left unattended, the goat will eat the cabbage. The states in Figure 2.1 are labeled by hyphenated pairs such as MG-WC, where the symbols to the left of the hyphen denote the entities on the starting bank and the symbols to the right denote the entities on the target bank. The labels M, G, W, and C refer to the man, the wolf, the goat and the cabbage respectively; \emptyset denotes the empty set. The inputs to the FA represent the crossings between the banks: m stands for only the man, w for man plus wolf, g for man plus goat, and c for man plus cabbage. The initial state is MWGC- \emptyset and the final state is \emptyset -MWGC (doubly circled). As one can see from

Figure 2.1 there are two equally short paths through the STD that solve the problem. Note that every transition has a similar reverse transition; this is not the case in general. Besides, these transitions do not advance the search for a solution.

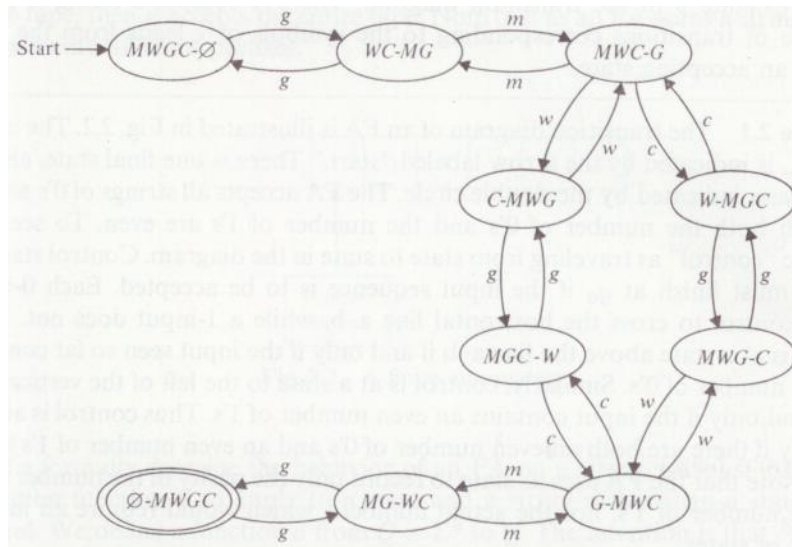


Figure 2.1 STD of the man-wolf-goat-cabbage problem (taken from [Hopcroft, Ullman, 1979])

An FA can formally be defined as a tuple $\langle I, S, T \rangle$ where I is the set of inputs, S the set of states, and T is the transition function. T is a partial mapping from $I * S$ to S . Thus, the extension of T is a set of tuples $(i, s\text{-from}, s\text{-to})$. In the man-wolf-goat-cabbage problem, $(m, WC-MG, MWC-G)$ is an example of such a tuple.

Another example of an FA is the complete transaction pattern from the ψ -theory, exhibited in Figure 2.2. The number of different states is 20 (excluding the non-communicative state of being executed; so, from the state of being promised, a direct transition to the state of being stated can be made).

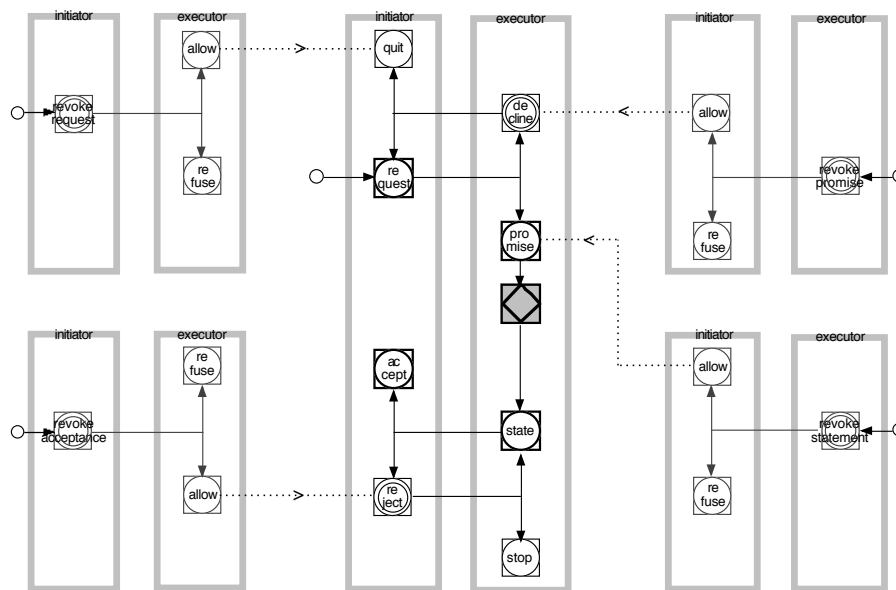


Figure 2.2 Graphical notation of the complete transaction pattern [JDM-5]

As is explained in [Perinforma 2013], the graphical notation in Figure 2.2 is a compressed form of the STD. For example, the symbol labeled “request” consists of both the act of requesting and the resulting state of being requested. Arrows are only precedence relations; they are not elements of the FA. The initial state is represented by the small disk to the left of the request symbol. There are three final states: accepted, quitted, and stopped.

When applying the formal definition of the FA to the complete transaction pattern, the transition function consists of tuples $(i, s\text{-from}, s\text{-to})$ where i is a coordination act, and $s\text{-from}$ and $s\text{-to}$ are coordination facts. Examples of such tuples are:

(request, <initial state>, requested)
 (promise, requested, promised)
 (decline, requested, declined)
 (request, declined, requested)
 (quit, declined, quitted)
(revoke-request, <revoke from state>, request-revoked)
(allow-revoke-request, request-revoked, request-revoked-allowed)
 (<no act>, request-revoked-allowed, quitted)
(refuse-revoke-request, request-revoked, request-revoked-refused)
 (<no act>, request-revoked-refused, <revoke from-state>)

In these example tuples, the states labeled in italics are meta states of the transaction process. The process returns from the meta level to the standard level automatically. This is represented by $i = \text{<no-act>}$. Returning to the standard level state where the revoke of a basic coordination act (request, promise, state, accept) was performed from, is denoted by <revoke from-state>.

3. The DELTA automaton

The core of the δ -theory is the DELTA automaton (Discrete Event in Linear Time Automaton). An instance of the automaton type DELTA is called a *delta*. Every delta can be used as a conceptual model of a concrete system (Cf. τ -theory [JDM-2]). As said, a delta is an FA that allows to study the kinematics and dynamics of a system, in addition to its statics. Delta's operate in a discrete linear time dimension, which means that there are distinct points in time, and that the difference between any two consecutive points in time is the same.

The operation of a delta can be explained as follows. At every point in time a delta is in some state, called its (current) *location*. When an input *event* occurs, a *transition rule* is evaluated, resulting into a state *transition* from the location to a state, called its *destination*. At the same time, an event may be generated. An event is a pair (A, t) where A is an *act* and t is a point in time; t is called the *occurrence* time or the *event* time.

Hereafter, a formal definition of a delta is presented. Points in time are represented by elements of the set \mathbb{T} ; the current point in time is denoted by Now. Time delays are represented by elements of the set \mathbb{D} .

A delta can formally be defined as a tuple $\langle \mathbf{D}, \mathbf{E}, \mathbf{L}, \mathbf{T}, \mathbf{A} \rangle$, where:

D : a set of states, called *destinations*
E : a set of acts, called the *event base*
L : a set of states, called *locations*
T : a partial function, called *transition rules*
A : a set of acts, called the *action base*

$$\mathbf{T} : \mathbf{A} * \mathbf{L} \rightarrow \mathbf{D} * (\mathbf{E} * \mathbb{D})$$

The partial function **T** can conveniently be represented as a set of *transition rules* of the form $\langle A, L, D, R \rangle$ where: A is the current *action*; $A \in \mathbf{A}$;

L is the current state, called *location*; $L \in \mathbf{L}$;

D is the new state, called *destination*; $D \in \mathbf{D}$;

R is a pair (A, d) with $A \in \mathbf{E}$ and $d \in \mathbb{D}$, called the *reaction*; d is the *delay* of A ; it means that action A will become current at time $\text{Now}+d$; Note that \mathbf{E} contains the element NILL (meaning no event).

Informally expressed: if the event (A, Now) occurs, and the current state is L, then a transition is made to the new state D, and the event (E, Now+d) is generated (possibly none). Every transition is a change of state from a location (the current state) to a destination (the new state).

A sequence of events in the course of time is called a *process*. Hereafter, the formal definition of the process of a delta is provided. In order to take into account its history before the start of observing the process of a delta, we define the *initiation conditions* $I = \langle IT, IS, IA \rangle$, with:

- IT : the *initiation time*, i.e. the start time of observing the process; $IT \in \mathbb{T}$;
- IS : the *initial state*, i.e. the state at time IT; $IS \in \mathbf{L}$;
- IE : the *initial event*, i.e. the event (IA, t) with *initial act* $IA \in \mathbf{A}$ and $t > IT$;

In addition, we define the set of *environmental events* EE, with $EE \subseteq \mathbf{A} * \mathbb{T}$;

Next, in order to keep the specifications of the process variables as simple as possible, the transition function \mathbf{T} is split into two functions: TD and TR, where:

- TD : $\mathbf{A} * \mathbf{L} \rightarrow \mathbf{D}$, called the *destination function*;
- TR : $\mathbf{A} * \mathbf{L} \rightarrow (\mathbf{E} * \mathbf{D})$, called the *reaction function*;

The *process* of a delta can now formally be defined by the pair of process variables $\langle \alpha, \sigma \rangle$, where:

- $\alpha : \mathbb{T} \rightarrow \mathbf{A}$; $\alpha(t)$ is the *action* at time t;
- $\sigma : \mathbb{T} \rightarrow \mathbf{L}$; $\sigma(t)$ is the *state* at time t;

Because changes take place at discrete points in time, the next variable is convenient :

- $\tau : \mathbb{N} \rightarrow \mathbb{T}$; the points in time $\tau(n)$ are the only moments at which an event occurs;
 \mathbb{N} denotes the natural numbers; τ_n is a shorthand notation for $\tau(n)$;
- $\tau_0 = IT$;
- $\tau_1 = t \mid \exists x : \langle x, t \rangle = IE$;
- $\tau_{n+1} = \text{minimum} (t_i \mid t_i > \tau_n \wedge \exists x : \langle x, t_i \rangle = TR(\alpha(\tau_n), \sigma(\tau_{n-1})), t_e \mid t_e > \tau_n \wedge \exists x : \langle x, t_e \rangle \in EE)$, for $n > 0$;

In words: the time τ_{n+1} is the next point in time at which an event will occur. It is assumed that internal events (with event time t_i) and environmental events (with event time t_e) do not coincide.

- $\sigma(\tau_0) = IS$;
- $\sigma(\tau_{n+1}) = TD(\alpha(\tau_{n+1}), \sigma(\tau_n))$;

In words: the state at a time τ_{n+1} is the outcome of evaluating the function TD for the action at time τ_{n+1} and the state at time τ_n .

For every t: $\tau_n < t < \tau_{n+1} : \sigma(t) = \sigma(\tau_n)$;

- $\alpha(\tau_1) = IA$;
- $\alpha(\tau_{n+1}) = x \mid \langle x, \tau_{n+1} \rangle = TR(\alpha(\tau_n), \sigma(\tau_{n-1})) \vee \langle x, \tau_{n+1} \rangle \in EE$;

In words: the action α at a time τ_{n+1} is the outcome of evaluating the function TR for the action at time τ_n and the state at time τ_{n-1} or it is an environmental event.

For every t: $\tau_n < t < \tau_{n+1} : \alpha(t) = \emptyset$;

4. Example: Traffic Control System

Let us apply the δ -theory to the traffic control of a simple crossing of two roads. The light signals in the opposite directions of each road are identical. In each direction a recurrent pattern of light signal changes can be observed: green - yellow - red - green - yellow etc. Instead of roads we will talk of cycles. So, there are two cycles, called C1 and C2. At any moment each of the cycles is in a particular phase, denoted by “green”, “yellow” or “red”. The mutual dependencies between C1 and C2 are depicted in Figure 4.1. They are symmetric, so it suffices to explain the case that is exhibited in the figure. Both the red phase and the green phase of a cycle may be arbitrarily long. However, a green phase lasts at least the so-called (standard) green time (GT). It can be prolonged after that time, which is indicated in the figure by the dashed line.

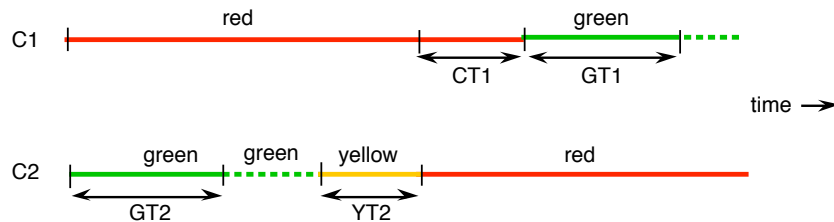


Figure 4.1. The traffic control cycles and their interdependencies

Let us observe what happens if during the green phase of C2 a car in one of the directions of C1 approaches the crossing. At some distance before the crossing, the car passes a traffic sensor that is put in the road. This causes the generation of a signal that is sent to the Traffic Control System. What happens next depends on the time that C2 is already in its green phase. If it is less than the standard green time (GT2) then the control system waits until the end of that period. If the approaching of the car is in the prolonged green phase (the dashed line in Figure 4.1), the controller proceeds immediately. The first thing that happens then is that C2 switches to phase yellow for some amount of time, called the yellow time (YT2). After that, it switches to the red phase. It always takes a particular amount of time to clear a crossing. Therefore, C1 will stay in its red phase for some time after C2 has gone into its red phase. This is called the clear time (CT1). After that time has passed, C1 goes over in its green phase, for at least GT1 time units. It will be prolonged as long as there is no traffic in the cross directions. The phases of C1 and C2 are made known to the traffic participants by means of the traffic lights. The red phase is represented by the red light, the yellow phase by the yellow light and the green phase by the green light. Obviously, at any point in time, only one of the three lights is switched on.

The components **L**, **D**, **A**, and **E** of the traffic control system can now be specified as follows (Cf. Figure 4.1):

L = {R1G2, R1Y2, R1R2, G1R2, Y1R2};

D = {R1G2, R1Y2, R1R2, G1R2, Y1R2};

A = {summon1, setY1, setR1, setG1, summon2, setY2, setR2, setG2};

E = {setY1, setR1, setG1, setY2, setR2, setG2};

The set of locations **L** happens to be equal to the set of destinations **D**. This need not be the case in general. Next, the event base **E** happens to be a subset of the action base **A**. Note that this is not the normal case. The action base **A** contains two external events: summon1 and summon2.

The transition rules **T** are specified in the table below. By the duration of R1G2 is meant the time that has passed since the transition to R1G2 was made. Note: The transition rules should be extended with a similar set of rules, in which cycle 1 and cycle 2 are exchanged. They are omitted for the sake of brevity.

action	location	destination	reaction event	reaction delay
summon1	R1G2		setY2	$\max(0, (GT2 - \text{duration}(R1G2)))$
setY2	R1G2	R1Y2	setR2	YT2
setR2	R1Y2	R1R2	setG1	CT1
setG1	R1R2	G1R2		

Although the Traffic Control System is still a simple system, it could not be studied properly by the classical FA, because the FA lacks the time dimension. One could only draw up the STD as shown in figure 4.2. This STD is a correct model of the state space (the set of allowable states, thus $L \cup D$) and the transition space (the set of allowable transitions) of the system, but it disregards the environmental events (summon1 and summon2). This omission can formally be solved by extending the state space with the states 'C1 summoned' and 'C2 summoned' (Cf. Figure 4.3). However, that is a trick, because the added states are artificial; they cannot be observed.

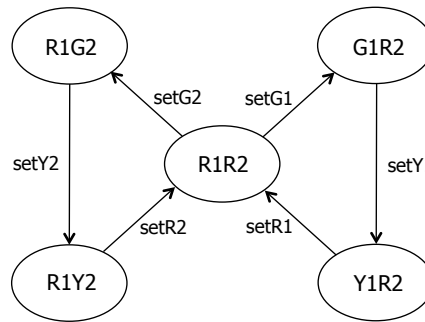


Figure 4.2. STD of the Traffic Control System

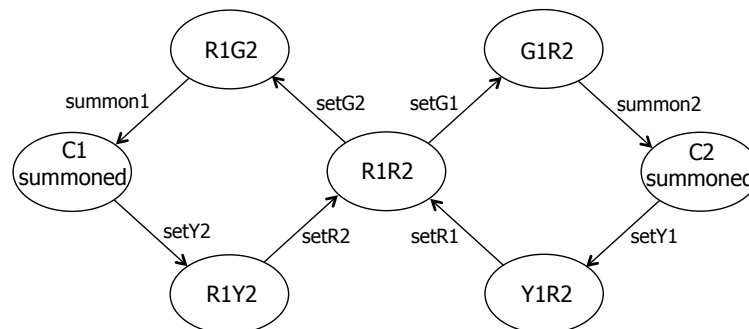


Figure 4.3. STD of the Traffic Control System (extended version)

5. Discussion

The δ -theory offers an automaton type (delta) that is able to capture the kinematics of a discrete event system, in contrast to the classical FA. The process of a delta can be a true model of the process of the modelled system. It is not necessary that the event times are exactly known. Values like "asap" (as soon as possible) or "any time" (the exact time doesn't matter) are valid values.

There is a causal mechanism but it is primitive one: a delta just responds instantly to its current action.

References

Dietz, J.L.G., Hoogervorst, J.A.P. et. al: The Discipline of Enterprise Engineering. In: *International Journal of Organisational Design and Engineering*, Vol. 3, No. 1, 2013, pp 86-114

Hopcroft, J.E., Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979

Perinforma, A.P.C.: *The essence of organisation*. Sapio, 2013

List of TEEMs (Theories in Enterprise Engineering Memorandum)

TEEM-1: the ω -theory (OMEGA: Organisation's Management, Engineering & Governance Appreciation)

TEEM-2: the τ -theory (TAO: Teleology Across Ontology)

TEEM-3: the δ -theory (DELTA: Discrete Event in Linear Time Automaton)

TEEM-4: the ϕ -theory (FI: Fact and Information)

TEEM-5: the ψ -theory (PSI: Performance in Social Interaction)

TEEM-6: the σ -theory (SIGMA: Socially Inspired Governance and Management Advancement)

TEEM-7: the π -theory (PI: Performance in Interaction)

TEEM-8: the β -theory (BETA: Binding Essence to Technology under Architecture)

TEEM-9: the ν -theory (NU: Normalised Unification)