REACHABILITY IN DISCRETE EVENT SYSTEMS MODELED AS HIERARCHICAL STATE MACHINES

Y. Brave
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000 , Israel.

M. Heymann
Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000 , Israel.

Abstract

Discrete event systems (DESs) are systems in which state changes take place in response to events that occur discretely, asynchronously and often nondeterministically. In this paper we consider a class of DESs modeled as hierarchical state machines (HSMs), a special case of the statecharts formalism introduced recently [2]. We provide an efficient algorithm for solving reachability problems in the HSM framework that utilizes the hierarchical structure of HSMs. This efficient solution is used extensively in control applications, where controllers achieving a desired behavior are synthesized on-line.

1. Introduction

Hierarchical state machines (HSMs), modeled after the StateChart formalism of Harel [3], are an extension of ordinary (sequential) state machines that are endowed with natural constructs of hierarchy (depth) and orthogonality (parallelism). Two basic structural features distinguish HSMs from ordinary state machines:

1. States are organized in a hierarchy of superstates and substates thereby achieving depth, and states are composed orthogonally (in parallel), thereby achieving concurrency.
2. Transitions are allowed to take place at all levels of the hierarchical structure, thereby achieving succinctness or descriptive economy (i.e., exponential reduction in descriptive complexity as compared to ordinary state machines).

HSMs are well suited for modeling and specification of complex processes such as manufacturing systems, communication networks, resource distribution systems, air traffic control systems, etc,

The paper focuses on HSM computations, with emphasis on computations of asynchronous HSMs (AHSMs). There is assumed to be no interaction between the parallel components of AHSMs (all interaction is assumed to be modeled in the control constraints; see [1] for more details). It is shown that such pivotal issues as computation of reachability can be executed in the AHSM framework in exponential reduction of complexity as compared with the equivalent ordinary state machine representation of the process. We develop an efficient algorithm for testing reachability that makes fundamental use of the hierarchical structure of the process thereby demonstrating the inherent advantage of the AHSM representation. In the remainder of this section we shall give an informal description of HSMs. The formal structure of HSMs is presented in section 2, whereas section 3 deals with reachability computations in the HSM framework.

States are represented by boxes. Hierarchy is represented by the insideness of boxes, as illustrated in Figure 1, where 1a may replace 1b. The symbols α–η stand for events associated with the various edges (transition-paths). The state a is called an *OR-state* which means that being in a is equivalent to being in either b or c or d (but not in more than one state at a time). The edge γ, which leaves the contour of a, applies to b, c and d, as in 1b. *Default-arrows* indicate default states. In Figure 1b, state e is selected as the initial state, and not a (a fact represented in 1a by the default arrow attached to e). The arrow attached to state c is the default among b, c and d if we are already in a, and alleviates the need for continuing the β-arrow beyond a's boundary.

Orthogonality or concurrency is the dual of the *OR* decomposition of states. In Figure 2a, state h consists of two *orthogonal components*, f and g, related by *AND;* to be in h is equivalent to being in both f and g, and hence the two default arrows. Edges *internal* to f, such as the transition labeled α, do not affect the g component. Thus, if α occurs at $<a,d>$, it affects only the f-component, resulting in $<b,d>$. An event λ at j causes entrance to combination $<a,d>$, and μ at $<b,e>$ causes transfer to i. The event θ from d states that the *AND*-state $h(=f\times g)$ is left and j entered, depending only on the fact that the g-component is actually at d. The η-arrow, on the other hand, leaves h unconditionally. By default arrows, event λ at i means entering $<b,e>$, whereas ρ means entering $<b,d>$. The intended semantic of Figure 2a is given by its equivalent 'flat' version Figure 2b.

2. Formal Structure of HSMs

An HMS is a structure $H=(A, \vdash, \Sigma, T, \rho)$ where: A is a set of states, $\vdash$ is the hierarchy relation on A, Σ is a set of event symbols, T is a set of transition-paths (or edges) and ρ is a default function. Next we give a detailed description of these elements, as well as some related notions (part of the terminology is adopted from [2, 4]). First, we shall need the following notations. Let x, y and z be three tuples. We shall write $x \subseteq y$ **iff** every element of x is an element of y, e.g., $<a,b> \subseteq <a,c,b,d>$. We shall write $z = x - y$ **iff** z consists of all elements of x that do not appear in y, e.g., $<a,c,b,d> - <a,b> = <c,d>$.

2.1 states

A is the (finite) set of states of H, consisting of A^+, the subset of *OR*-states, $A^\perp$, the subset of *AND*-states and A^{basic}, the subset of *basic* states (see below). The hierarchical structure of H's states is represented by the binary relation $\vdash$ on A, called the *hierarchy relation* and satisfies the following conditions:

1. There exists a unique state, called the *root state of* H and denoted $r=(r(H))$, such that for no state $a \in A$, $a \vdash r$.
2. For every state $a \in A$, $a \neq r$, there exists a unique state $b \in A$ such that $b \vdash a$. The state b is called the *immediate superstate* of a, whereas a is an *immediate substate* of a.
3. A state $a \in A$ is basic if and only if a has no immediate substates.
4. If $a \vdash b$ then either $a \in A^+ \wedge b \notin A^+$, or $a \in A^\perp \wedge b \in A^+$.

Assumption 4, *the alternating structure assumption,* means that the immediate substates of *OR*-states are either *AND*-states of basic states, whereas the immediate substates of *AND*-states are *OR*-states. The reflexive and transitive closure of $\vdash$ is denoted $\vdash^*$. Thus, $a \vdash^* b$ means that b is a (not necessarily immediate) substate of a. It is clear that the pair $(A, \vdash)$ defines a tree, called the *hierarchy tree* of H (see

Figure 3 for the hierarchy tree of the HSM of Figure 2a).

2.2. Configurations

Let q be a tuple of (disjoint) states. (The examples throughout this subsection relate to Figure 2a).

- The *restriction* of q to a state a, denoted $q\restriction_a$, is obtained from q by deleting all elements that are not substates of a. E.g., $<b,e>\restriction_f = <b>$.
- A state a is a *Superstate* of q if every element of q is a substate of a. E.g., h and k are superstates of $<b,e>$.
- The *Lowest Superstate* of q denoted $LS(q)$, is the superstate a of q that satisfies the condition that for each superstate b of q, $b \vdash^* a$. E.g., $LS(<b,e>) = h$.
- Two states q_1 and q_2 are *orthogonal*, denoted $q_1 \perp q_2$, if either $q_1 = q_2$ or, alternatively, if neither is a superstate of the other and $LS(<q_1,q_2>) \in A^{\perp}$. A tuple of states q is *orthogonal* if every pair of states in q is orthogonal. E.g., $<b,e>$ is orthogonal, whereas $<b,h>$ and $<b,i>$ are not orthogonal. An orthogonal tuple is also called a *configuration*. Intuitively, a configuration is a tuple of states all of whose elements can be occupied simultaneously when running H.
- Let q be a configuration and let a be a superstate of q. Then q is a *full configuration of* a if it cannot be extended with the augmentation of further orthogonal substates of a, i.e., if q satisfies the condition that

$$\forall\, b \in A\,, \quad a \vdash^* b \;\Rightarrow\; <q,b> \text{ is not orthogonal}\,. \qquad (2.1)$$

  If q does not satisfy (2.1) then it is a *partial* configuration of a. E.g., $<b>$ and $<b,g>$ are, respectively, partial and full configurations of h. The configuration q is *basic* if all its elements are basic states. The set of all basic full configurations of a is denoted Q_a.
- For a basic (partial or full) configuration of state a, the *a-span* of q, denoted $C_a(q)$ is defined as the set of all basic full configurations p of a such that $q \subseteq p$. E.g., $C_k(<b>) = \{\, <b,e>, <b,d> \,\}$.

2.3 Transition-paths

Associated with each *OR*-state a is a set T^a of transition-paths. A *transition-path* of a is formally represented by a triple $t = (u,\sigma,v)_a$, where u is a (possibly partial) configuration of a and v is a full configuration of a, called, respectively, the source and destination configurations of t, and $\sigma \in \Sigma$ is an event symbol that labels t. The association of t with T^a implies that in the associated HSM graph, a is the lowest *OR*-state containing t's source and destination configurations, as well as its entire arc. Thus, in Figure 2a, the transition-path labeled α belongs to state f, whereas the θ-transition-path belongs to state k. The set of transition-paths T of the entire HSM is defined as $T = \bigcup_{a \in A^*} T^a$. For each *OR*-state a, let S_a (D_a) denote the set of all source (respectively, destination) configurations of transition-paths of a. It was shown in [2] that it is possible to transform an HSM to one in which every source configuration of a transition-path is basic. Henceforth, we shall assume that all HSMs have been thus transformed.

2.4 Default configurations

- The *default function* $\rho : A^+ \to A$ specifies for each *OR*-state an immediate substate, called its *default*.
- The *default configuration function* $\hat{\rho}$ specifies inductively for each state a a unique basic full configuration, called its *default configuration*, as follows:
  1. For an *AND*-state a with immediate substates $a_1, \ldots, a_k$,
  $$\hat{\rho}(a) = <\hat{\rho}(a_1), \ldots, \hat{\rho}(a_k)>\,.$$
  2. For an *OR*-state a with immediate substates $a_1, \ldots, a_k$,
  $$\hat{\rho}(a) = \hat{\rho}(a_i) \quad \textbf{iff} \quad a_i = \rho(a)\,.$$
  3. For a basic state a, $\hat{\rho}(a) = <a>$.

  In Figure 2a, $\hat{\rho}(k) = \hat{\rho}(h) = <\hat{\rho}(f), \hat{\rho}(g)> = <b,e>$.
- Let $q = <q_1, \ldots, q_l>$ be a full configuration of a state a. The *default* of q, denoted $d(q)$, is then defined as the basic full configuration $<\hat{\rho}(q_1), \ldots, \hat{\rho}(q_l)>$ of a (where $\hat{\rho}(q_i)$ is the default configuration of q_i as defined above).
- A transition-path $(u,\sigma,v)_a$ is *canonical* if its destination configuration is a basic full configuration of a, that is, if $d(p) = p$.

We shall assume throughout that transition-paths are given in their canonical form. In view of the assumption that source configurations are basic and that transition-paths are in canonical form, we shall assume henceforth that all configurations are basic.

2.5 Transition functions

In the remainder of the paper we shall consider only asynchronous HSMs (AHSMs), that is, HSMs in which no two distinct states have transition-paths labeled by identical event symbols. That is, for every pair of distinct states $a, b \in A^+$

$$(u,\sigma,v) \in T^a \;\wedge\; (u',\sigma',v') \in T^b \;\Rightarrow\; \sigma \neq \sigma'\,.$$

We interpret the transition-paths of an AHSM H as follows. Suppose H is at configuration $q \in Q$ ($=Q_r$). Then a transition labeled $\sigma \in \Sigma$ is *defined* at q **iff** there exists a transition-path $t = (u,\sigma,v)$ such that $u \subseteq q$. Furthermore, the 'next' configuration of H will be p, where p is the configuration obtained from q by replacing (in q) the restriction of q to a with the destination configuration v. Thus, in Figure 2a, the transition-path $t = (<d>, \theta, <j>) \in T^k$ is defined at configuration $q = <b,d>$, and if the AHSM H executes θ at q it enters configuration $p = <q - q\restriction_k, j> = <j>$ (since $q\restriction_k = q$). Formally, we associate with each state $a \in A$ the *transition function* $\delta_a : Q_a \times \Sigma \to 2^{Q_a}$ satisfying the condition that for all $q,p \in Q_a$ and $\sigma \in \Sigma$, $p \in \delta_a(q,\sigma)$ **iff** there exists a transition-path (u,σ,v) of a substate of a such that

$$u \subseteq q \quad \wedge \quad p = <q - q\restriction_a, v>\,.$$

The transition function of H is defined as $\delta = \delta_r$. We interpret an AHSM $H = (A, \vdash, \Sigma, \tau, \rho)$ as a device that starts at configuration $q_o = \hat{\rho}(r)$ and executes configuration transitions according to its transition function δ. That is, H can be represented by its equivalent (ordinary) state machine $M(H) = (Q, \Sigma, \delta, q_o)$ whose states consist of all full configurations of H and whose transition function is the transition function δ of (the root of) H.

3. Reachability

In this section we discuss the problem of testing reachability of a set of (full or partial) configurations from a given full configuration. Let $a \in A$. A *path* of a is a finite sequence $s = q_o, \sigma_1, q_1, \ldots, \sigma_n, q_n$,

where the q_i are full configurations of a and the σ_i are symbols in Σ, such that $q_i \in \delta_a(q_{i-1}, \sigma_i)$ for all $i = 1, 2, \ldots, n$. In this case we say that q_n is *a-reachable* from q_o. For a subset P of full configurations of a, define $R_a^{-1}(P)$ to be the set of all full configurations of a from which P can be a-reached.

Given a full configuration q of H and a subset P of configurations of H, our objective is to verify whether there exists a full configuration w of H such that for some $p \in P$, $p \subseteq w$, and w is r-reachable from q, i.e., to verify whether

$$q \in R_r^{-1}(C_r(P)) \ , \tag{3.1}$$

where $C_r(P)$ is the r-span of P. In the sequel, we present an algorithm for testing (3.1) that does not require the construction of the equivalent state machine $M(H)$ whose state set is exponential in the number of orthogonal components in H. Let us begin with a simple example. Consider the AHSM H depicted in Figure 4. Let $q = \langle a_1, b_1 \rangle$ and $p = \langle a_2, b_2 \rangle$, and suppose we wish to verify whether p is c-reachable from q. That is, whether there exists a path s of c that starts at q and ends at p, such that s consists only of transition-paths that belong to substates of c; in this case ψ, ϕ, θ and ρ (for clarity, we identify transition-paths with their event symbols). By the asynchrony assumption, this question can be resolved by independent reachability tests in states a and b. Thus we check whether $\langle a_2 \rangle$ is a-reachable from from $\langle a_1 \rangle$, and whether $\langle b_2 \rangle$ is b-reachable from from $\langle b_1 \rangle$. Since the answer to the latter is negative, we conclude that $\langle a_2, b_2 \rangle$ is not c-reachable from $\langle a_1, b_1 \rangle$.

Next we discuss the effect of the transition-paths α, β, γ and δ of state f. Specifically, we wish to verify whether $p = \langle a_3, b_3 \rangle$ is f-reachable from $q = \langle a_1, b_1 \rangle$. Since p is *not* c-reachable from q, we search for a configuration $s \in S_f$ (i.e., s is a source of a transition-path of f) such that s is c-reachable from q. Since the source $\langle b_3 \rangle$ of γ is not c-reachable from $q = \langle a_1, b_1 \rangle$, we proceed with β whose source $\langle a_2 \rangle$ is c-reachable from q. Our final destination is $p = \langle a_3, b_3 \rangle$ (which is a configuration of c), and thus we continue with α, thereby entering configuration $\langle a_4, b_4 \rangle$. Now we check whether $p = \langle a_3, b_3 \rangle$ is c-reachable from $\langle a_4, b_4 \rangle$. Since this is not the case, we continue with δ, the only transition-path of f whose source $\langle b_3 \rangle$ is c-reachable from $\langle a_4, b_4 \rangle$, and return to state c through γ. This search terminates successfully since $p = \langle a_3, b_3 \rangle$ is c-reachable from the destination $\langle a_3, b_2 \rangle$ of γ. In summary, $p = \langle a_3, b_3 \rangle$ is f-reachable from $q = \langle a_1, b_1 \rangle$ via the following path (see Figure 5): $\langle a_1, b_1 \rangle$, ψ, $\langle a_2, b_1 \rangle$, β, $\langle d \rangle$, α, $\langle a_4, b_4 \rangle$, ρ, $\langle a_4, b_3 \rangle$, δ, $\langle e \rangle$, γ, $\langle a_3, b_2 \rangle$, ρ, $\langle a_3, b_3 \rangle$.

Notice that during the search performed in the previous paragraph we checked whether p is c-reachable from q, and from $\langle a_4, b_4 \rangle$ and $\langle a_3, b_2 \rangle$, where the latter are configurations of c that are in D_f, the set of all destinations of transition-paths of f. In fact, a reachability test from $q, \langle a_4, b_4 \rangle$ and $\langle a_3, b_2 \rangle$ has been carried out also w.r.t. $\langle a_2 \rangle$ and $\langle b_3 \rangle$ that are configurations of c and belong to S_f, the set of all sources of transition-paths of f. Thus we conclude that the information about reachability within state c, that is relevant for state f, is the c-reachability of configurations in $S_f \cup \{ q \}$ from configurations in $D_f \cup \{ p \}$. This observation is a key point in the development of the algorithm below for reachability computations associated with (3.1).

Fix a full configuration q of H, and a set P of configurations of H. For each state $a \in A$ we define a set $X_a(q)$ (called the *input* set of a) of full configurations of a, and a set $Y_a(P)$ (called the *output* set of a) of configurations of a, as follows. A configuration w of a is an element in $X_a(q)$ **iff** either $x = q \mid_a$, or $x = d \mid_a$ where d is a destination configuration in D_b for some strict superstate b of a. That is, in Figure 4, $X_c(\langle a_1, b_1 \rangle) = \{ \langle a_4, b_4 \rangle, \langle a_3, b_2 \rangle \} \cup \{ \langle a_1, b_1 \rangle \}$. Analogously, a configuration y of state a is an element in $Y_a(P)$ **iff** either $y = p \mid_a$ for some $p \in P$, or $y = s \mid_a$ where s is a source configuration in S_b for some strict superstate b of a. That is in Figure 4, $Y_c(\{ \langle a_3, b_3 \rangle \}) = \{ \langle a_2 \rangle, \langle b_3 \rangle \} \cup \{ \langle a_3, b_3 \rangle \}$.

It should be clear from the examples above that for each state $a \in A$ at a given level in the hierarchy, all the information about reachability that may be required for higher level computations concerns only a-reachability tests between input configurations in $X_a(q)$ and output configurations in $Y_a(P)$. If a is an *AND*-state, these a-reachability tests are carried out separately and independently in each substate of a. If, however, a is an *OR*-state, we test a-reachability in the digraph $G_a(q, P)$ whose edge set consists of all transition-paths of a, and edges representing reachability within the substates of a. Figure 6 shows the digraph $G_f(q, p)$, where f is the root state of the AHSM in Figure 4, $q = \langle a_1, b_1 \rangle$ and $p = \langle a_3, b_3 \rangle$. The results of these tests are represented by a subset $W_a(q, P) \subseteq X_a(q) \times Y_a(P)$, where for a pair $(x, y) \in X_a(q) \times Y_a(P)$, $(x, y) \in W_a(q, P)$ means that y is a-reachable from x. The computation proceeds inductively (up the hierarchy), and since for the root state r, $X_r(q) = \{ q \}$ and $Y_r(P) = P$, the verification of (3.1) can be accomplished by testing whether $W_r(q, P) = \varnothing$. Formally, we have the following algorithm for testing reachability.

Algorithm: Given q and P as above, compute $W_a(q, P) \subseteq X_a(q) \times Y_a(P)$ inductively (up the hierarchy) as follows:

(1) For a basic state a, $W_a(q, P) = \varnothing$.

(2) For an *OR*-state a with immediate substates $a_1, \ldots, a_k$, and for all $(x, y) \in X_a(q) \times Y_a(P)$:
$(x, y) \in W_a(q, P)$ **iff** y is reachable from x in the digraph $G_a(q, P)$, whose node set is

$$V_a(q, P) = X_a(q) \cup Y_a(P) \cup D_a \cup S_a \ ,$$

and whose edge set is

$$E_a(q, P) = \left[\bigcup_{i=1}^{k} W_{a_i}(q, P) \right] \cup \{ (u, v) \mid \exists \ \sigma, \text{ s.t. } (u, \sigma, v) \in T^a \} \ .$$

(3) For an *AND*-state a with immediate substates $a_1, \ldots, a_k$, and for all $(x, y) \in X_a(q) \times Y_a(P)$:
$(x, y) \in W_a(q, P)$ **iff** for each substate a_i **either** $(x \mid_{a_i}, y \mid_{a_i}) \in W_{a_i}(q, P)$ **or** $y \mid_{a_i} = \langle \ \rangle$.

(4) Upon termination (at the root state r),

$$q \in R_r^{-1}(C(P)) \quad \textbf{iff} \quad W_r(q, P) \neq \varnothing \ .$$

■

Example 3.1: Consider the AHSM H of Figure 4, and suppose we wish to test whether $p = \langle a_3, b_3 \rangle$ is reachable from $q = \langle a_1, b_1 \rangle$. For applying the Algorithm above, we first compute the input/output sets: $X_f(q) = \{ q \}$, $Y_f(p) = \{ p \}$, $X_c(q) = \{ q, \langle a_4, b_4 \rangle, \langle a_3, b_2 \rangle \}$, $Y_c(p) = \{ p, \langle a_2 \rangle, \langle b_3 \rangle \}$, $X_a(q) = \{ \langle a_1 \rangle, \langle a_3 \rangle, \langle a_4 \rangle \}$, $Y_a(p) = \{ \langle a_3 \rangle, \langle a_2 \rangle \}$, $X_b(q) = \{ \langle b_1 \rangle, \langle b_2 \rangle, \langle b_4 \rangle \}$ and

$Y_b(p) = (<b_1>, <b_3>)$ The digraphs $G_a(q,p)$ and $G_b(q,p)$ (see step (2) in the algorithm) consist of the transition-paths of states a and b, respectively. Thus

$$W_a(q,p) = \{ (<a_1>, <a_2>), (<a_3>, <a_3>), (<a_3>, <a_2>) \} ,$$

and

$$W_b(q,p) = \{ (<b_1>, <b_1>), (<b_2>, <b_1>), (<b_2>, <b_3>), (<b_4>, <b_3>) \}$$

The digraph G_f is given in Figure 6, where the set $W_c(q,p)$ is the set of all dashed arrows. Since p is reachable from q in G_f, and therefore, $W_f(q,p) \neq \emptyset$, we conclude that p is reachable from q in H. ■

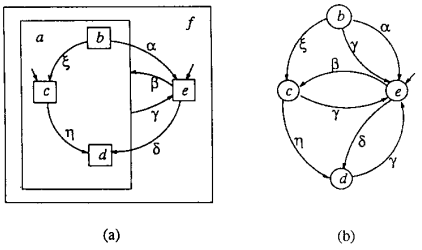

Figure 1: (a) An HSM H consisting of OR-states. (b) The equivalent state machine of H.

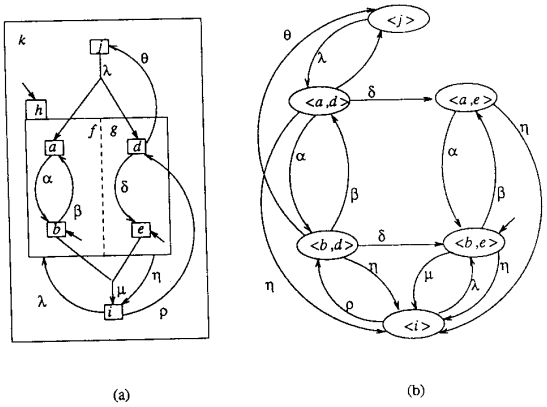

Figure 2: (a) An HSM H consisting of AND- and OR-states. (b) The equivalent state machine of H.

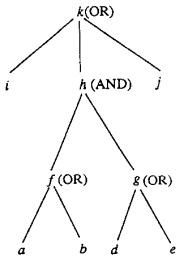

Figure 3: The hierarchical tree of the HSM of Figure 2(a).

References

[1] Brave, Y. and Heymann, M., " Control of discrete event systems modeled as hierarchical state machines", CIS Report #9103, Technion - IIT, 1991.

[2] Drusinsky, D., " On synchronized statecharts " , Ph. D. Dissertation, Weizmann Institute of Science, Rehovot, April 1988.

[3] Harel, D., " Statecharts: A visual formalism for complex systems", *Science of Computer Programming*, **8** , pp. 231-274,1987.

[4] Harel, D., Pnueli, A., Schmidt, J. P., and Sherman, R., " On the formal semantic of statecharts" , *IEEE Symposium on Logic in Computer Science*, 1987.

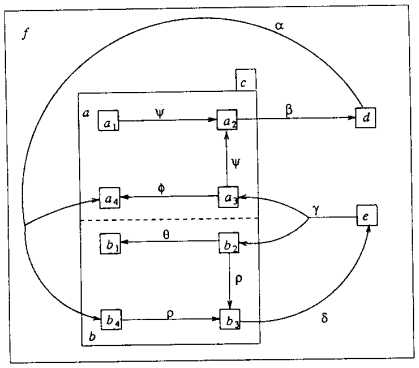

Figure 4: The HSM of Example 3.1.

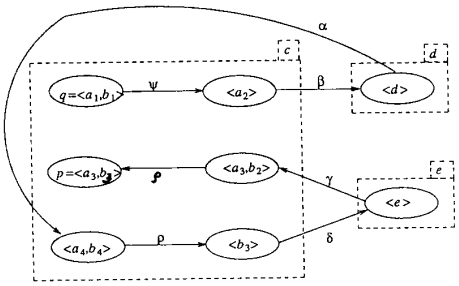

Figure 5: A path of state f (in Figure 10) that connects q to p.

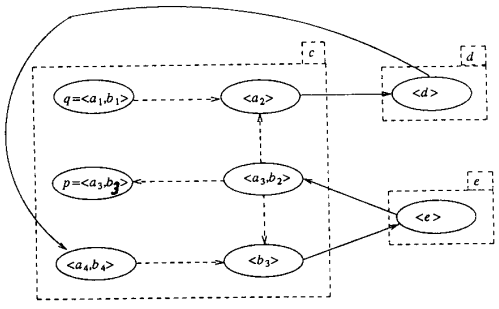

Figure 6: The digraph $G_f(q,p)$ of state f in Figure 10.