A Discrete-Event Systems Approach to Modeling Dextrous Manipulation

S. L. Ricker[⋆] N. Sarkar[⋆†] K. Rudie[‡⋆]

November 1995

External Technical Report

ISSN-0836-0227-

95-390

[⋆]Department of Computing and Information Science

[†]Department of Mechanical Engineering

[‡]Department of Electrical and Computer Engineering

Queen's University

Kingston, Ontario K7L 3N6

Document prepared November 8, 1995

Abstract

To perform dextrous manipulation efficiently, it is necessary to coordinate the interactions of many component processes. This paper investigates one approach to coordination: discrete-event systems. The applicability of discrete-event systems to the modeling of dextrous manipulation tasks is studied. Discrete-event control theory offers formal methods for determining whether a coordinator of the components can be generated. A representative dextrous manipulation task, the planar Grasp-Lift-Replace task of Howe and Cutkosky, is presented as a discrete-event process. The task is extended to include two-fingered exploratory procedures. The effectiveness of the discrete-event system approach is illustrated through simulations of several test cases.

1 Introduction

In many robotics problems, robot hands must perform delicate and precise operations that include grasping and smoothly lifting an object. Robotics research has addressed many components of this dextrous manipulation problem. These issues include multi-arm or finger coordination for manipulating objects [17],[26],[27], optimally distributing the load among different arms [28], decomposing the grasp force into equilibrating and interacting force fields [13], and allowing contact motion while manipulating an object [4],[22]. While efficient dextrous manipulation requires designing a controller for each component process (e.g., controlling contact force), it is also necessary to design a mechanism by which the component processes can be enabled and disabled in a preferred sequence.

Dextrous manipulation is a process in which tactile events mark transitions between phases of the manipulation task, resulting in control discontinuities. The need for techniques to facilitate a smooth progression of control through the discrete phases of the manipulation task is presented in [5]. Our work represents an initial effort to use a theoretical framework for the coordination of these control discontinuities. Describing a task as a series of distinct segments is not novel to dextrous manipulation; however, there is a need for a sound theoretical approach to provide not only a high-level coordinator, but also to provide some insight into the task organisation and help determine when, where and if sensors should be included as part of the manipulation operation. We apply the discrete-event control theory of Ramadge and Wonham [18] to the high-level description and coordination of a dextrous manipulation task. This leads to the design of a controller that permits only a set of desired actions and forbids undesirable actions. The discrete-event systems approach provides a systematic procedure to generate strategies guaranteed to achieve a feasible goal for the task.

Discrete-event control theory has been used to control a variety of robotic applications including manufacturing and assembly tasks [2], [16], the coordination of mobile robotic agents [12], and a grasping task performed under the supervision of a vision system [23]. We present here a more complex working example of both the supervisor and the plant using the control-theoretic approach of Ramadge and Wonham, thereby explicitly synthesising a supervisor. There are different ways to represent a discrete-event system, including Petri nets, finite-state machines, and formal logic. We use finite-state machines to model a dextrous manipulation task because the finite-state machine formalism is sufficiently expressive for our purposes and we would like to exploit the existing body of work in discrete-event control of finite-state machines.

First, we present a description of the task. We state the assumptions required for the discrete-event systems model and discuss possible strategies for using two fingers to measure several object properties during the manipulation task. Next, a summary of discrete-event control theory is provided. Finally, the model and results of simulations are discussed.

An earlier summary version of this paper appeared in [20].

2 Background

The purpose of this paper is to investigate a discrete-event systems approach to designing a high-level supervisory controller for a dextrous manipulation task. We select a task (described in Section 2.1) that has received considerable attention in the study of both biological and robotic tactile systems and that has not been modeled by a discrete-event system formalism. In addition, we propose a modeling strategy that allows exploratory procedures (EPs) to be carried out in the course of the task. There are some tasks, like the Grasp-Lift-Replace task, where it is possible to incorporate into the manipulation task itself the determination of certain physical properties of an object. To that end we indicate how two existing exploratory procedures, a friction EP and a stiffness EP, could occur with minimal intrusion during the Grasp-Lift-Replace task. To our knowledge, incorporating EPs into the task itself has not been examined in existing work.

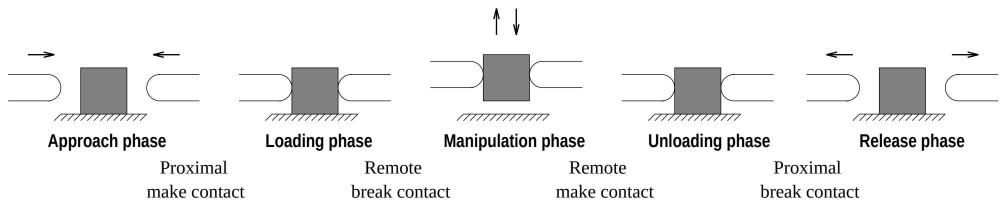

Figure 1: Phases of a Grasp-Lift-Replace task and the tactile events that distinguish phases as adapted from studies by Johansson and Westling(1987) by Howe *et al.* (1990); after Howe *et al.* (1990).

2.1 The Grasp-Lift-Replace Task

Johansson and Westling [11] addressed the role of tactile signals during manipulations performed by humans — in particular, manipulating objects using a precision grip. The task selected for their series of studies is deceptively simple: an object is grasped between the thumb and index finger, lifted vertically off the table, and subsequently replaced on the table. The Grasp-Lift-Replace task is divided into a series of distinct phases where a transition from one phase to the next is triggered by one or more of the four types of tactile units in the fingertips.

Of particular interest is the relationship between the **grip force**, which is the force required to secure the object between the fingers, and the **load force**, which is the vertical lifting force required to overcome gravity. In a typical lifting task, the ratio between the grip and load force remains constant after the initial contact with the object. To perform the manipulation task smoothly, it is necessary to sense the coefficient of friction at the point of contact.

The series of experiments conducted by Johansson and Westling has contributed much towards an understanding of how tactile sensors in the hands control fine manipulation tasks. To determine whether similar mechanisms would aid in the control of robot manipulation tasks, Howe and Cutkosky [8],[10] apply the hypotheses of the human studies to a robotic system. The robotic Grasp-Lift-Replace task involves five phases which are linked by four contact events (see Figure 1): the **approach** phase — the two fingers move towards the object in anticipation of the grasp; the **loading** phase — when both fingers make contact with the object, the grip (horizontal) force is increased simultaneously with the load (vertical) force until the object is both secure in the grasp and lifted off the table, signaling the beginning of the next phase; the **manipulation** phase — contact between the object and the table is broken, the object is lifted to a pre-determined height and then returned to the table ; the **unloading** phase — the object is on the table and the grip force is decreased; and the **release** phase — the fingers break contact with the object and move away to a neutral position. A change in contact marks the transition from one phase to another. In fact, the contact events also indicate a change from one low-level control mode to another. For instance, the end of the approach phase is indicated by contact between the fingers and the block (proximal make contact) and signifies a change from position to force control.

Robotic tactile sensors, described in [8], [9] and [24], detect the contact events and trigger the transitions through the phases of the Grasp-Lift-Replace task. The specialised sensors detect slip during finger/object contact, as well as vibrational information to identify the remote contact events.

2.2 Exploratory Procedures

Exploratory procedures (EPs), first described by Lederman and Klatzky [14], elucidate a set of hand configurations and actions that humans consistently use to determine specific properties of objects. The structured nature of the EPs provides a framework for the development of similar procedures for robotic hands. Robotic EPs have been implemented for dextrous manipulators [1],[25] and for robot fingers[3],[6]. Some of the sampled object properties include: gross object size (volume), surface texture, thermal properties, weight, gross object shape, hardness/softness, and the shape of the contact area.

3 Task Description

We have selected the Grasp-Lift-Replace manipulation task of Howe and Cutkosky [10], with an additional step to determine stiffness, to be modeled as a discrete-event system. This provides a simple and concrete example against which we test the modeling framework. Information available from the manipulation task includes three important properties: coefficient of friction between the finger and the object, weight and stiffness of the object. These properties play a significant role in grasping and manipulation and sometimes need to be determined through tactile exploration.

To conform to the experimental set-up of Howe and Cutkosky, the following restrictions apply:

- The objects are block-shaped.
- The location of an object in space is known.
- Each finger is equipped with a two-axis force sensor, to measure the grasp and load forces.
- Only one finger has a multi-element stress-rate sensor [24], to detect incipient slip.
- The dynamic tactile sensors are able to distinguish incipient slip from other vibrations.
- Fingertip position information is available.
- Once the coefficient of friction is estimated, the subsequent adjustment to the grasp force is sufficient to ensure no further slip occurs.

To impose some reasonable constraints on the discrete-event system, the following assumption is made: there is a maximum number of attempts allowed for aligning the fingers on either side of the object and for achieving stable contact — exceeding this threshold results in starting the task again or, in the latter case, failure of the task with no re-initiation permitted.

The task is divided into seven phases and includes two EPs, as illustrated in Figure 2. The approach, loading, manipulation, unloading and release phases are adopted directly from Howe *et al.* [10]. The **squeezing** phase incorporates the Stiffness EP. The Friction EP spans the loading and initial manipulation phases.

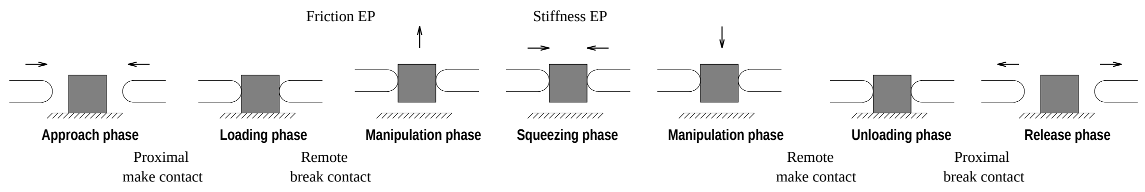

Figure 2: Phases of a Grasp-Lift-Replace task as modified for the discrete-event model.

After successful completion of the approach phase, the Friction EP occurs. The coefficient of friction is calculated from normal and tangential force signals immediately preceding the detection of incipient slip [24]. Starting with a small grip force (loading phase), the load force is applied to lift the object (manipulation phase). When slip is detected (by the multi-element stress rate sensor), the grip force is increased, and another attempt is made to lift the object. This procedure repeats until the maximum number of increments is exceeded or the object is successfully lifted. The lifted object is held in static equilibrium at the end of the first manipulation phase. From the Friction EP we measure the minimum grip force, F_g, and the load force, F_l. Once the Friction EP is determined, the weight of the object, W, can be calculated. With an estimate of the coefficient of friction, μ, for the contact between the finger and the block, and the grip force the weight W is given by $W = 2\mu F_g = F_l$.

The Stiffness EP is patterned after the one-finger "Hardness" EP of Dario *et al.* [6] and constitutes the squeezing phase of the task. With the object held stationary at the end of the first manipulation phase, the gripping force is increased by ΔF_g which we call the squeezing force. This produces a deformation of the object by an amount Δx, which can be determined by the position of the fingertips. The stiffness is the ratio of the squeezing force to the deformation produced, $K = \Delta F_g/\Delta x$. It is possible that the deformation produced at each finger is different, though the squeezing force is the same, because the object is not homogeneous. In that case, the stiffness is taken as the average of the two.

4 Discrete-Event System Background

This work adopts the framework for discrete-event systems as described by Ramadge and Wonham in [19]. A brief review of essential concepts, collected from [18],[19],[21], is provided in this section.

A discrete-event system is a process characterised by sequences of events. In particular, a change in a system state of a process is precipitated by the occurrence of an action or event, not merely by the passage of time. Ideally we would like to control the undesirable behaviour of a discrete-event system by either preventing some events from taking place or allowing — but not forcing — others to occur.

The uncontrolled discrete-event system is modeled by an automaton, called the *plant*, of the following form:

$$G = (Q, \Sigma, \delta, q_o, Q_m)$$

where Q is a set of *states*; Σ is a non-empty set of event labels called an *alphabet*; δ is the *transition function*, a partial function $\delta : \Sigma \times Q \to Q$; $q_o \in Q$ is the *initial state*; and $Q_m \subseteq Q$ is the set of *marker* (terminal) *states*. When Q is finite, G can be described as a finite-state automaton and can be represented by a directed graph where the nodes of the graph are the states in Q, the arcs of the graph are the transitions defined by the function δ, and the set of labels for the arcs are the events in Σ. Thus for any event $\sigma \in \Sigma$ and an initial state $q_o \in Q$, $\delta(\sigma, q_o)$ is defined (written $\delta(\sigma, q)!$) if there is an arc from q_o to some other state labeled by σ.

The set Σ^* contains all possible finite sequences, or *strings*, over Σ plus the null string ε. The definition for δ can be extended to Σ^*

$$\delta(\varepsilon, q) := q,$$

$$(\forall \sigma \in \Sigma)(\forall s \in \Sigma^*)\delta(s\sigma, q) := \delta(\sigma, \delta(s, q)).$$

The language generated by G, also called the *closed behaviour* of G, describes all possible event sequences that the discrete-event system can undergo

$$L(G) := \{s : s \in \Sigma^* \text{ and } \delta(s, q_o)!\}.$$

The language $L_m(G)$, or the *marked behaviour* of G, describes all possible event sequences that represent completed tasks

$$L_m(G) := \{s : s \in \Sigma^* \text{ and } \delta(s, q_o) \in Q_m\}.$$

By definition, $L_m(G) \subseteq L(G)$.

For any string $s \in \Sigma^*$, we say that $t \in \Sigma^*$ is a *prefix* of s if s=tu for some $u \in \Sigma^*$. Thus every string $s \in \Sigma^*$ has at least two prefixes, ε and s.

If $L \subseteq \Sigma^*$, the *prefix closure* of L is a language, denoted by $\overline{L}$, consisting of all prefixes of strings of L

$$\overline{L} := \{t \in \Sigma^* | t \text{ is a prefix of } s, \text{ for some } s \in L\}.$$

Because every string is a prefix of itself, $L \subseteq \overline{L}$. A language is said to be prefix-closed if $L = \overline{L}$.

To establish supervision on G, we partition the set of events Σ into the disjoint sets Σ_c, *controllable* events, and Σ_{uc}, *uncontrollable* events. Controllable events are those events whose occurrence is either preventable (i.e., may be disabled) or allowable (i.e., are said to be 'enabled'). Uncontrollable events are those events which cannot be prevented and are deemed permanently enabled. A *supervisor* (or controller) may enable or disable controllable events at any time during its observation of a sequence of events generated by G. Thus the supervisor allows only a subset of $L(G)$ to be generated.

Formally, a supervisor $\mathcal{S}$ is a pair (S, ψ) in which S is an automaton

$$S = (X, \Sigma, \xi, x_o, X_m)$$

where X is a set of *states* for the supervisor; Σ is the alphabet used by G; ξ is the *transition function*, a partial function $\xi : \Sigma \times X \to X$; x_o is the *initial state* for the supervisor; X_m is the set of *marker* states; and ψ, called a *feedback map*, is given by $\psi : \Sigma \times X \to \{0, 1\}$ satisfying

$$\psi(\sigma, x) = 1 \text{ if } \sigma \in \Sigma_{uc}, x \in X,$$

$$\psi(\sigma, x) \in \{0, 1\} \text{ if } \sigma \in \Sigma_c, x \in X.$$

The number 0 is interpreted as the command "disable" and the number 1 as "enable". That is, ψ is interpreted as a rule for disablement and ensures that uncontrollable events are never disabled. The automaton S monitors the behaviour of G and changes state according to the events generated by G. The control rule $\psi(\sigma, x)$ dictates whether σ should be enabled or disabled at the corresponding state in G.

The behaviour of G when it is constrained by $\mathcal{S}$ is described by the automaton $\mathcal{S}/G$, called the *supervised discrete-event system*:

$$\mathcal{S}/G = (\Sigma, Q \times X, (\delta \times \xi)^{\psi}, (q_o, x_o), Q_m \times X_m).$$

The behaviour of $\mathcal{S}/G$ is described by $L(\mathcal{S}/G)$ and $L_m(\mathcal{S}/G)$. The modified transition function $(\delta \times \xi)^{\psi}$ is defined as a mapping $\Sigma \times Q \times X \to Q \times X$:

$$(\delta \times \xi)^{\psi}(\sigma, (q, x)) := \begin{cases} (\delta(\sigma, q), \xi(\sigma, x)) \text{ if } \delta(\sigma, q)!, \\ \quad \xi(\sigma, x)!, \text{and } \psi(\sigma, x) = 1; \\ \text{undefined otherwise.} \end{cases}$$

A supervisor $\mathcal{S} = (S, \psi)$ is *nonblocking* for G if

$$L(\mathcal{S}/G) = \overline{L_m(\mathcal{S}/G)}.$$

That is, a nonblocking supervisor ensures that, in closed-loop, any sequence s that is started (i.e., $s \in L(\mathcal{S}/G)$) can be completed to a marked sequence (i.e., $s \in L_m(\mathcal{S}/G)$).

The centralised control problem we consider is introduced in [18]:

> *Given a plant G over an alphabet Σ (with controllable events Σ_c) and given some non-empty languages A and E where $A \subseteq E \subseteq L(G)$ find a nonblocking supervisor $\mathcal{S}$ such that*
>
> $$A \subseteq L(\mathcal{S}/G) \subseteq E.$$

What this formalism captures is problems where some process that can be described as a finite state machine is given (in this case, G), and some set of desirable (or "legal") sequences is given (in this case, E) and a controller is sought to inhibit process behaviour so that only desirable sequences are generated. The language A describes the minimally acceptable set of sequences that any closed-loop solution must contain.

To describe a solution to the above problem, it is convenient to use the notion of *controllability*. Given G over an alphabet Σ, for a language $K \subseteq L(G)$, K is *controllable* with respect to G if

$$\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K} \tag{1}$$

where $\overline{K}\Sigma_{uc} := \{st \,|s \in \overline{K} \text{ and } t \in \Sigma_{uc}\}$. If we think of E as a set of "legal" sequences, then we want to know when it will be impossible to stop an illegal sequence from happening. It must be that the introduction of an uncontrollable event into a legal sequence results in another legal sequence. If E is not controllable, a largest (or supremal) controllable sublanguage of E (possibly $\emptyset$), denoted $\sup\underline{C}(E, G)$, can always be found [18]. The standard solution to the control problem produces a supervisor that acts on G to generate $\sup\underline{C}(E, G)$. The important point to note is that such a solution is said to be "minimally restrictive" in that the supervisor disables events in G only when absolutely necessary to prevent an illegal sequence from occurring. That is, the largest possible subset of legal sequences is generated.

Software developed in the Systems Control Group in the Department of Electrical Engineering at the University of Toronto (TCT931124), under the supervision of W. M. Wonham, was used to compute all results reported here. This software tool performs operations on discrete-event systems such as finding the supremal controllable sublanguage of a given legal language.

5 Modeling the Dextrous Manipulation Task

For the Grasp-Lift-Replace (GLR) task, we can model the capabilities of the two robotic fingers as finite-state machines, which together play the role of the "plant", and we can characterise desirable or "legal" sequences by a set of finite-state machines (captured in one composite automaton). Our goal is to find a supervisor to monitor the behaviour of the plant. This amounts to determining when the fingers must be prevented from taking certain actions. To solve this problem, we use the TCT software and input our finite-state descriptions of the plant G and the legal language E. The software produces $sup\underline{C}(E, G)$, the largest achievable legal behaviour. If a sequence can be generated by G but is not in $sup\underline{C}(E, G)$, it must be prevented from occuring. Thus, our problem solution is a listing of commands that indicates when the fingers must be prohibited from performing certain actions.

The dextrous manipulator has two fingers, and therefore we model the actions of each finger separately. However, there are some aspects of the GLR task (e.g., remote make contact) in which the behaviour of both fingers must be synchronised. To accommodate these situations, our plant is the *synchronous product* of the finite-state automata for Finger 1 and Finger 2: $G = FINGER1 /\!/ FINGER2$. An event occurs in the synchronous product only if it occurs in all automata in which the event appears. Shared events lead to coupled transitions which model the actions where the fingers operate concurrently. In general, a synchronous product models the behaviour of two finite-state machines operating concurrently. For example, consider G_1 and G_2 in Figure 3. The synchronous product $G_1 /\!/ G_2$ admits sequences that can be generated by G_1 and G_2 where events common to both machines (in the example, only β) can occur only when each machine is in a state where such an event is defined. In other words, initially either α or γ can occur; if α occurs, then β may occur next since G_1 will be at state 2 and G_2 at state 1 and β is defined at each of these states. Events that are not common to both machines may occur as long as they occur in the appropriate order defined by the transition function of the finite-state machine in which they appear. The behaviour of a synchronous product may be characterised by an automaton whose state set is given by the Cartesian product of the composite automata. When the event sets of G_1 and G_2 are disjoint, the resulting behaviour of $G_1 /\!/ G_2$ is called the *shuffle product* of G_1 and G_2. For a more formal definition, see [7].

The finite-state diagram for Finger 1 is given in Figure 4. Finger 2 is analogous. Nodes represent states of the plant, while the arc labels are the events in Σ. An arc

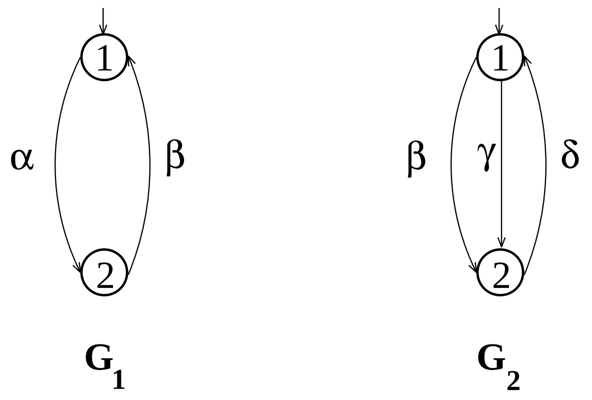

Figure 3: Example for synchronous product.

label superscripted with an asterisk(*) indicates an event in which both fingers must be synchronised. The remaining events are defined independently for each finger. The initial state is identified by a small entry arrow and marker states are identified by a small exit arrow. The plant has only two marker (terminal) states: the initial state prior to the start of the task, and the final state resulting from both fingers breaking contact with the object at the conclusion of the task. Note that because the initial state is marked, the task of "doing nothing" is considered a completed task. The resulting automaton for our dextrous manipulator ($G = \mathit{FINGER1} /\!/ \mathit{FINGER2}$), which defines all possible coordinated and independent behaviour of the two fingers, has a total of 90 states and 242 transitions.

There are many ways in which the GLR task can be decomposed into a set of discrete events. We chose to describe each phase of the GLR task as a sequence of subtasks. While these subtasks are not necessarily comprehensive, the set of events we describe for our discrete-event system captures sufficient detail to illustrate the applicability of our approach. For example, planning a finger trajectory to the object is not specifically enumerated and is subsumed by the $\mathit{approach}_1$ and $\mathit{approach}_2$ events. Similarly, there is an assumption that underlying control exists to coordinate the continuous-time dynamics of the system. The set of events in the GLR task is given by the set Σ below. A subscript of 1 or 2 indicates the finger to which the event relates. A superscript of * indicates an event shared by both fingers.

$$\begin{aligned}\Sigma \; = \; \{ & approach_1, approach_2, retract\_finger_1, retract\_finger_2, \\ & detect\_misalignment^*, re\text{-}align\_with\_finger1^*, align\_finger2^*, \\ & prox\_make\_C_1, prox\_make\_C_2, prox\_breakC_1, prox\_breakC_2, \\ & unstable\_contact_1, unstable\_contact_2, suppress\_vib_1, \\ & suppress\_vib_2, orient\_normal_1, orient\_normal_2, \\ & fing_1\_obj\_misalign, fing_2\_obj\_misalign, apply\_gripF^*, \\ & apply\_loadF^*, incip\_slip^*, inc\_gripF^*, remote\_breakC^*, \\ & dec\_gripF^*, move\_to\_Z^*, stiffness\_EP^*, remote\_makeC^* \}\end{aligned}$$

where, for i=1,2,

- *approach*$_i$ is the approach of Finger i towards the object;
- *retract_finger*$_i$ is the removal of Finger i from the proposed contact location; unlike *prox_break_C*$_i$, no contact has been made and fingers are simply withdrawn;
- *detect_misalignment** occurs when Finger 2 is not lined up in space with Finger 1; to check for this error state we assume position information available to determine that the grip force is approximately collinear;
- *re-align_with_finger1** occurs when the two fingers are not lined up in space and Finger 2 must be moved to a position opposite to the reference point Finger 1;
- *align_finger2** is the pre-contact alignment of Fingers 1 and 2 : Finger 1 is used as a point of reference to determine a contact location for Finger 2;
- *prox_make_C*$_i$ indicates that contact is made between Finger i and the object;
- *prox_breakC*$_i$ is the retraction of the fingers and signals the end of contact between the fingers and the object;
- *unstable_contact*$_i$ is the result of an inability to suppress contact vibrations and local contact is not sufficiently stable to continue the task;

- *suppress_vib*$_i$ means it is necessary to suppress any contact vibration between the finger and the object;
- *orient_normal*$_i$ is the action of rotating the finger at the contact location until the finger is oriented normally to the object;
- *fing*$_i$*_obj_misalign* indicates that Finger i cannot be oriented normally to the object;
- *apply_gripF** is the application of a horizontal force to the object to secure it between the fingers;
- *apply_loadF** is the application of a vertical force to overcome gravity;
- *incip_slip** is the tactile sensing of incipient slip at a contact point;
- *inc_gripF** indicates an incremental increase in the grip force in response to the tactile sensing of slip at a contact point;
- *remote_breakC** is the tactile sensor signal that the object has broken contact with the table;
- *dec_gripF** is the incremental decrease of grip force to initiate incipient slip for purposes of determining the coefficient of friction at the contact point;
- *move_to_Z** is the event requiring the object to be lifted to a pre-specified height above the table;
- *stiffness_EP** indicates that the Stiffness EP occurs;
- *remote_makeC** is the event signaled by a tactile sensor which indicates that the object is once again in contact with the table.

The set of uncontrollable events for the two fingers is

$$\begin{aligned}\Sigma_{uc} &= \{detect\_misalignment^*, unstable\_contact_1, unstable\_contact_2,\\ &\qquad fing_1\_obj\_misalign, fing_2\_obj\_misalign, incip\_slip^*\}.\end{aligned}$$

To construct a supervisor for G, we first detail the desired or legal behaviour of G. We present four specifications, each of which can be expressed as a finite-state automaton, that define the legal language E:

1. Finger 1 approaches the object before Finger 2.
2. Only two consecutive attempts (per approach) are allowed for aligning the two fingers prior to contact.
3. A total of two attempts to make contact will be permitted. After the third unsuccessful attempt, the task terminates with both fingers returning to a starting position.
4. If incipient slip is detected just after the load force is applied, only two attempts to adjust the grip force are allowed. If slip is sensed a third time, the task terminates and both fingers return to a starting position.

The automaton for each specification is shown in Figure 5. It might not be obvious that these finite-state machines do, indeed, capture the verbal specifications. The modeling of the constraints for the GLR task was an iterative process where successive attempts were incomplete due to sequences that would be missing from $\sup\underline{C}(E,G)$. This led to new insights on how to alter the finite-state machines that would generate the system behaviour intended by the verbal specifications. It was also this stage of the modeling that proved to be the most difficult. Whenever verbal specifications must be translated into a language (such as finite-state machines) used by a formalism, the same modeling problem arises: how do we know that this translation accurately reflects the original specifications? This uncertainty is *not* indigenous to discrete-event control theory. Without a systematic approach, one would still need to produce an algorithm or a computer program from a verbal description of the problem. In discrete-event control theory, we can at least pinpoint the place where such modeling accountability is required.

We calculate E by taking the intersection of the four specifications: $E = \mathrm{Spec}_1 \cap \mathrm{Spec}_2 \cap \mathrm{Spec}_3 \cap \mathrm{Spec}_4$. The legal language has 1412 states and 13724 transitions. For the third specification, only the first finger is shown; an isomorphic automaton is defined for Finger 2, and Spec$_3$ is the shuffle product of the two automata. It is important to observe that the legal language presented here is one of many possible combinations of constraints that could be prescribed for the GLR task.

The control problem we are interested in requires that the supervisor $\mathcal{S}$ must impose the legal behaviour or the largest controllable subset of legal behaviour on G. Thus we need an automaton that recognises only the desirable sequences as described by E. In addition we need a set of control rules that will indicate whether or not a given event at the current state of the plant is enabled or disabled. A supervisor that will solve our problem can be constructed as follows: first, we compute $sup\underline{C}(E, G)$, the largest controllable sublanguage of E; then we define a supervisor that ensures that only those strings and all those strings of G that are in this controllable sublanguage are permitted to occur. The minimally adequate language A will capture error-free executions of the task. Formally, A is

$$\begin{aligned}&\{approach_1, approach_2, align\_finger2^*, prox\_make\_C_1, prox\_make\_C_2,\\ &suppress\_vib_1, suppress\_vib_2, orient\_normal_1, orient\_normal_2,\\ &apply\_gripF^*, apply\_loadF^*, remote\_breakC^*, dec\_gripF^*, incip\_slip^*,\\ &inc\_gripF^*, move\_to\_Z^*, stiffness\_EP^*, remote\_makeC^*,\\ &prox\_break\_C_1, prox\_break\_C_2\}.\end{aligned}$$

It can be checked that $A \subseteq \mathrm{sup}\underline{C}(E, G)$. That is, under control, the system is certainly able to execute the GLR task if no error occurs. A supervisor does exist for the GLR task and has 1318 states and 2684 transitions.

It can be seen that for our system, $\mathrm{sup}\underline{C}(E, G) \neq E$. Thus there are some sequences that are legal but lead to a bad state if followed by an uncontrollable event. For example, suppose a string $s = approach_1\ approach_2\ align\_finger2^*\ prox\_make\_C_1$. According to the third specification the following is a valid sequence:

$$\begin{aligned} s' \;:=\; & s \;\; \textit{unstable\_contact}_1 \;\; \textit{prox\_break\_C}_1 \\ & s \;\; \textit{unstable\_contact}_1 \;\; \textit{prox\_break\_C}_1 \\ & s \;\; \textit{unstable\_contact}_1 \;\; \textit{prox\_break\_C}_1 \;\; s \end{aligned}$$

but if we let $\sigma =$*unstable_contact*$_1$ then $s'\sigma \in L(G)$ and $\sigma \in \Sigma_{uc}$ so it cannot be prevented from occurring. However, $s'\sigma$ would be illegal (i.e., $s'\sigma \notin E$). Therefore, our control solution ensures that even though s' is legal, it is prevented from occuring since $s'\sigma$ is illegal and if s' happened, $s'\sigma$ could not be stopped.

We present representative results of our supervised discrete-event system $\mathcal{S}/G$ in the next section.

6 Simulation Results

Our solution, by construction, ensures that the largest possible subset of the legal language is met. We have conducted a series of simulations to illustrate that the supervisor takes appropriate actions for various sequences of plant behaviour. The simulation takes a desired sequence of events as input, performs a simple breadth-first search of the plant and the supervisor, and, using the feedback map, ascertains the status of each event (disable, enable). To illustrate that our controller correctly disables events, we selected two cases: a scenario where the fingers were not aligned on either either side of the block after three consecutive attempts to re-align the fingers, and an attempt at lifting the block where incipient slip occurs three times in a row after subsequent increases in the grip force.

6.1 Problems aligning the fingers

The first sequence tests specification two: the task is successfully started, but more than three consecutive efforts are attempted to align the fingers in space.

approach$_1$, *approach*$_2$,

detect_misalignment*, *re-align_with_finger1**, *detect_misalignment**, *re-align_with_finger1**,

detect_misalignment*, *re-align_with_finger1**.

When we use this sequence of events as input to our supervisor $\mathcal{S}$, the plant G, and feedback map ψ, our simulation produces the following output:

```
Event approach_1 is enabled at state 0

Event approach_2 is disabled at state 0

Event approach_2 is enabled at state 1

Event detect_misalignment* is enabled at state 2

Event re-align_with_finger1* is enabled at state 6

Event detect_misalignment* is enabled at state 12

Event re-align_with_finger1* is enabled at state 24

Event detect_misalignment* is enabled at state 42

Event re-align_with_finger1* is disabled at state 69
```

The controller correctly disables the third attempt to re-align the fingers. Additionally, at state 0, the first finger is required to approach the object before the second finger.

6.2 Problems lifting the block

The second sequence demonstrates that the controller correctly handles the fourth specification: the block is grasped between the two fingers but after two adjustments to the grip force, remote contact is not yet broken and slip has occurred for a third time (perhaps the block is too heavy or slippery).

$approach_1$, $approach_2$, *align_finger2**,

$prox\_make\_C_1$, $prox\_make\_C_2$, $suppress\_vib_1$, $suppress\_vib_2$,

$orient\_normal_1$, $orient\_normal_2$, *apply_gripF**, *apply_loadF**,

incip_slip*, *inc_gripF**,

incip_slip*, *inc_gripF**,

incip_slip*, *inc_gripF**.

The output from the simulation produced the following control sequence:

```
Event approach_1 is enabled at state 0

Event approach_2 is disabled at state 0

Event approach_2 is enabled at state 1

Event align_finger2* is enabled at state 2

Event prox_make_C1 is enabled at state 3

Event prox_make_C2 is enabled at state 8

Event suppress_vib1 is enabled at state 17

Event suppress_vib2 is enabled at state 29

Event orient_normal1 is enabled at state 49

Event orient_normal2 is enabled at state 73

Event apply_gripF* is enabled at state 106

Event apply_loadF* is enabled at state 147

Event incip_slip* is enabled at state 194

Event inc_gripF* is enabled at state 248

Event incip_slip* is enabled at state 313

Event inc_gripF* is enabled at state 390

Event incip_slip* is enabled at state 472

Event inc_gripF* is disabled at state 564
```

The controller cannot disable an uncontrollable event (*incip_slip**) so to meet the fourth specification (i.e., allow only two chances to increase the grip force and avoid further incipient slip) the third occurrence of *inc_gripF** is disabled.

7 Discussion

Previous discrete-event system models of robotic applications have presented relatively coarse decompositions of the task under consideration. We believe that models of dextrous manipulation tasks require a finer breakdown to more accurately represent the integration of sensors into the task. While the planar GLR task has well-defined roles for sensors, such delegation may not be as straightforward for a new task. A finer granularity of the model will help in understanding how the dextrous manipulation task should be performed.

The complexity of the model is a function of the level of detail for the task description as well as the number of fingers on the dextrous robot hand. We chose to model the behaviour of a planar two-fingered manipulator; however, many dextrous manipulation tasks can be accomplished with only three fingers and thus a different manipulator with more fingers would not necessitate exponential growth in the size of the plant and supervisor.

We have presented the design of a discrete controller for a dextrous manipulator and have illustrated some of its functionality with a computer simulation. Incorporating the controller into part of the overall control of a robot hand raises many issues. At the implementation level, in the event that there is more than one way to accomplish the task, some events or transitions may have to be prioritised. For instance, if at a particular state there are several transitions that will eventually lead to the final state some criteria must determine which one to chose. Additionally, Leduc and Wonham [15] recently described the complications of implementing a discrete controller. For example, very large systems are best served by a modular control strategy, thereby reducing the size of the supervisor by taking advantage of obvious parallelism in the application. There is no parallelism inherent to the GLR task and thus this would not be a useful strategy for reducing the size of the supervisor.

One of the difficulties we encountered in the modeling of the GLR task was capturing the uncontrollable event of slip. Ideally, in keeping with the Ramadge-Wonham model, the event of incipient slip should be self-looped; however, since an arbitrary number of slips without a corresponding increase in the grip force would be illegal, controllability would ensure that the task would end before ever getting to a state where incipient slip could occur. Thus we had to impose the control of slip into the plant by removing the self-loop and replacing it with the *incip_slip***/inc_gripF** loop. This is because the interpretation of transitions in the Ramadge-Wonham model is that an event *can* occur, not that it *must* occur. So if there is a slip event that leads out of some state **Y** and the *inc_gripF** event also leads out of **Y**, the Ramadge-Wonham model does not provide a way to guarantee that *inc_gripF** must occur.

8 Conclusions

This initial study suggests that the discrete-event control theory of Ramadge and Wonham provides a useful formalism for capturing the high-level structure of a robotic dextrous manipulation task. Discrete-event control theory offers a simple methodology for describing the characteristics of a plant and for determining the existence of a supervisor. If a supervisor exists, then we can guarantee that there exist sequences of events that will result in a completed task. Describing the desired behaviour of the plant requires only the specification of a finite set of constraints. Similarly, changing any aspect of the plant's "good" behaviour is a matter of adding or removing a constraint.

We have considered the centralised control problem where one supervisor tracks the behaviour of the plant, and all events are observable. Our study revealed that more complicated models capture the richness of a representative sensory-driven dextrous manipulation task. Future work would include an examination of timed or even hybrid models to address the issue of providing a more accurate description of sensor behaviour as well as handling situations where events can be forced to occur and not merely enabled.

Acknowledgements

The authors would also like to thank R. E. Ellis and P. E. G. Gogolek for their helpful comments on the manuscript, and W. M. Wonham for providing the TCT software.

References

[1] P.K. Allen and P. Michelman. Acquisition and interpretation of 3-D sensor data from touch. *IEEE Transactions on Robotics and Automation*, 6(4):397–404, 1990.

[2] B.A. Brandin, W.M. Wonham, and B. Benhabib. Manufacturing cell supervisory control - a modular timed discrete-event system approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 846–851, 1993.

[3] D. G. Caldwell, A. Buysse, and Z. Weizhan. Multi-sensor tactile perception for object manipulation/identification. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1904–1911, 1992.

[4] A. Cole, J. Hauser, and S. Sastry. Kinematics and control of multifingered hands with rolling contact. In *Proceedings of the 1988 International Conference on Robotics and Automation*, pages 228–233, Philadelphia, Pennsylvania, April 1988.

[5] M. R. Cutkosky and J. M. Hyde. Manipulation control with dynamic tactile sensing. In *6th ISRR, Hidden Valley, Pennsylvania*, 1993.

[6] P. Dario, P. Ferrante, G. Giacalone, L. Livaldi, B. Allotta, G. Buttazzo, and A. Sabatini. Planning and executing tactile exploratory procedures. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1896–1903, 1992.

[7] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1985.

[8] R.D. Howe and M. R. Cutkosky. Sensing skin acceleration for slip and texture perception. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 145–150, 1989.

[9] R.D. Howe and M. R. Cutkosky. Dynamic tactile sensing: Perception of fine surface features with stress rate sensing. *IEEE Transactions on Robotics and Automation*, 9(2):140–151, 1993.

[10] R.D. Howe, N. Popp, P. Akella, I. Kao, and M. R. Cutkosky. Grasping, manipulation, and control with tactile sensing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1258–1263, 1990.

[11] R. S. Johansson and G. Westling. Signals in tactile afferents from the fingers eliciting adaptive motor responses during precision grip. *Experimental Brain Research*, 66:141–154, 1987.

[12] J. Košecká and L. Bogoni. Application of discrete event systems for modeling and controlling robotic agents. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2557–2562, 1994.

[13] V. Kumar and K. J. Waldron. Force distribution in walking vehicles on uneven terrain. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 112(1):90–99, 1990.

[14] S. J. Lederman and R. L. Klatzky. Hand movements: A window into haptic object recognition. *Cognitive Psychology*, 19:342–368, 1987.

[15] R. J. Leduc and W. M. Wonham. PLC implementation of a DES supervisor for a manufacturing testbed. To appear in *Proceedings of the 33rd Annual Allerton Conference on Communication, Control, and Computing*, 1995.

[16] B.J. McCarragher and H. Asada. A discrete event approach to the control of robotic assembly tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1993.

[17] Y. Nakamura, K. Nagai, and T. Yoshikawa. Dynamics and stability in coordination of multiple robotic mechanisms. *International Journal of Robotics Research*, 8(2):44–61, 1989.

[18] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[19] P.J. Ramadge and W.M. Wonham. The control of discrete-event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[20] S. L. Ricker, N. Sarkar, and K. Rudie. A discrete-event systems approach to dextrous manipulation. To appear in *Proceedings of the 33rd Annual Allerton Conference on Communications, Control, and Computing*, 1995.

[21] K. Rudie and W.M. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, 1992.

[22] N. Sarkar, X. Yun, and V. Kumar. Dynamic control of 3-D rolling in multi-arm manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 978–983, Atlanta, Georgia, May 1993.

[23] T.M. Sobh and R. Bajcsy. Autonomous observation under uncertainty. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1792–1798, 1992.

[24] J. S. Son, A. Monteverde, and R. D. Howe. A tactile sensor for localizing transient events in manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994.

[25] S. A. Stansfield. Haptic perception with an articulated, sensate robot hand. Technical report, Sandia National Laboratories, March 1990.

[26] T. J. Tarn, A. K. Bejcsy, and X. Yun. Design of dynamic control of two cooperating robot arms: Closed chain formulation. In *Proceedings of the 1987 International Conference on Robotics and Automation*, pages 7–13, Raleigh, North Carolina, March 1987.

[27] M. A. Unseren and A. J. Koivo. Reduced order model and decoupled control architecture for two manipulators holding an object. In *Proceedings of the 1989 International Conference on Robotics and Automation*, pages 1240–1245, Scottsdale, Arizona, May 1989.

[28] Y. Zheng and J. Y. S. Luh. Optimal load distribution for two industrial robots handling a single object. In *Proceedings of the 1988 International Conference on Robotics and Automation*, pages 344–349, Philadelphia, Pennsylvania, April 1988.

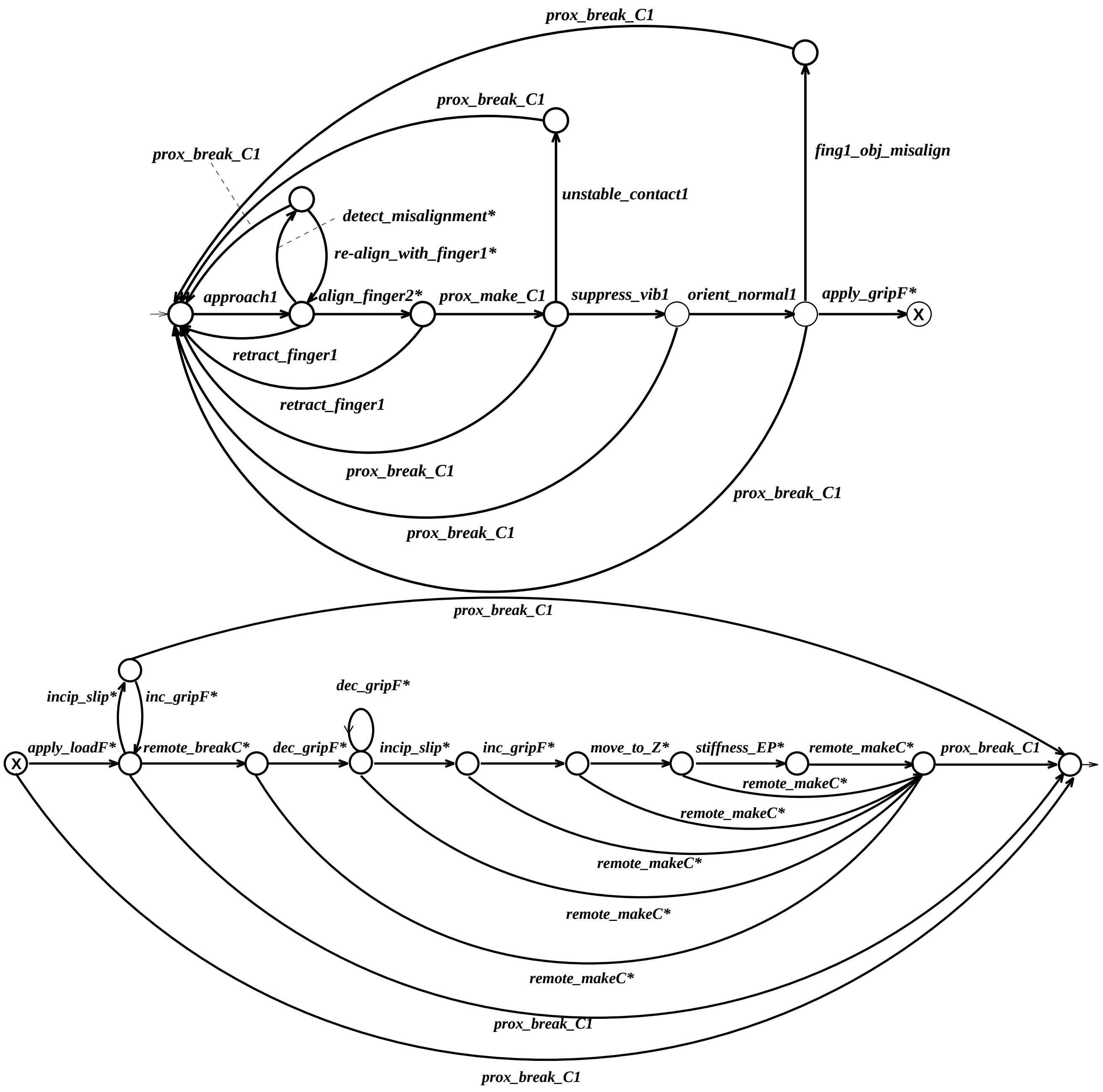

Figure 4: Finite-state automaton of Finger 1 for the Grasp-Lift-Replace task. Due to space limitations, the automaton has been separated into two parts, where state X of the top diagram is understood to be state X of the bottom diagram.

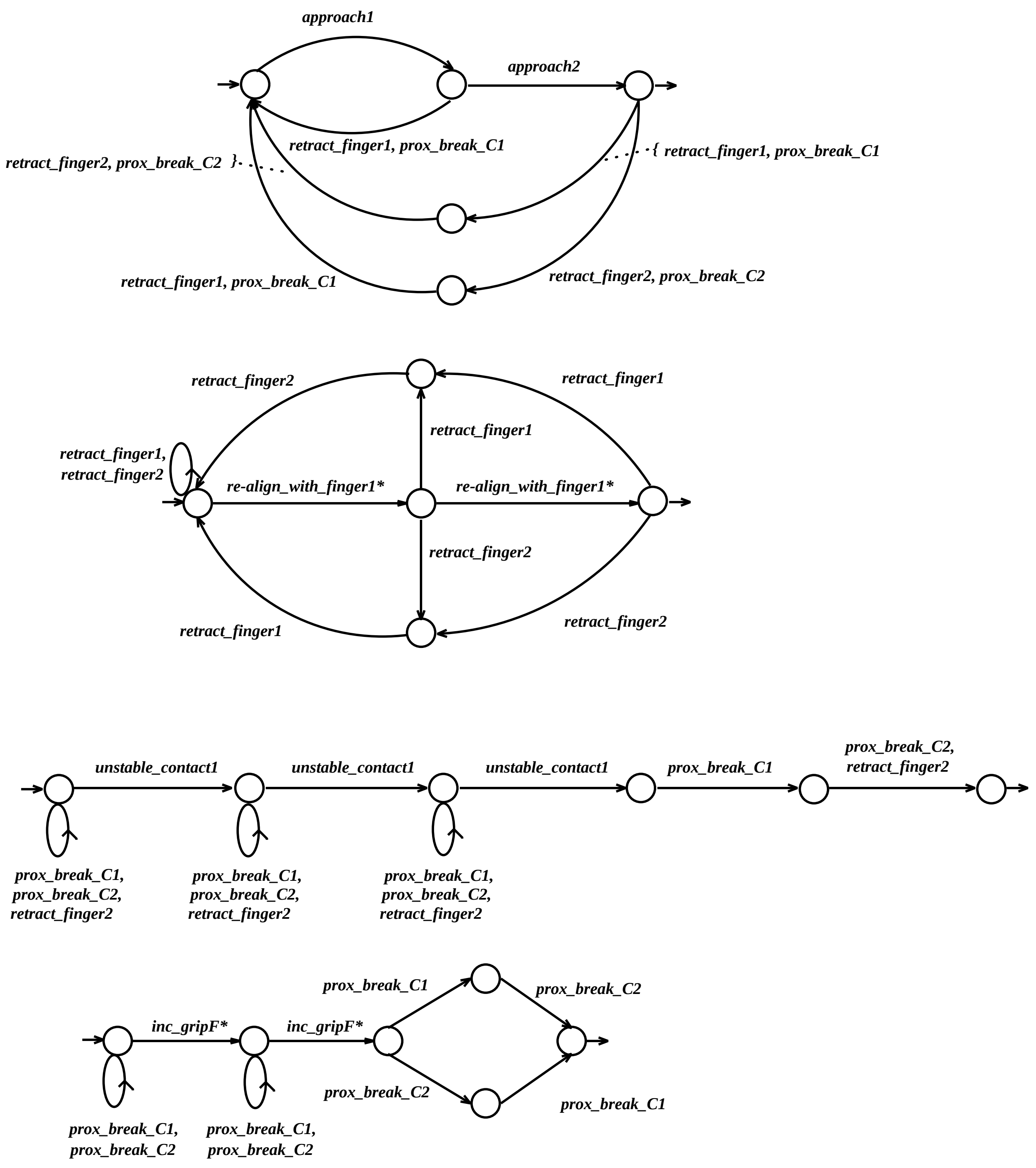
approach1
approach2
retract_finger1, prox_break_C1
retract_finger2, prox_break_C2
retract_finger1, prox_break_C1
retract_finger1, prox_break_C1
retract_finger2, prox_break_C2
retract_finger2
retract_finger1
retract_finger1
retract_finger1, retract_finger2
re-align_with_finger1*
re-align_with_finger1*
retract_finger2
retract_finger1
retract_finger2
unstable_contact1
unstable_contact1
unstable_contact1
prox_break_C1
prox_break_C2, retract_finger2
prox_break_C1, prox_break_C2, retract_finger2
prox_break_C1, prox_break_C2, retract_finger2
prox_break_C1, prox_break_C2, retract_finger2
prox_break_C1
prox_break_C2
inc_gripF*
inc_gripF*
prox_break_C2
prox_break_C1
prox_break_C1, prox_break_C2
prox_break_C1, prox_break_C2

Figure 5: Specifications 1, 2, 3 and 4 expressed as finite-state automata. It is assumed that all other events in Σ are self-looped (i.e., can occur at every state).