INFORMATION AND COMPUTATION **86**, 43–68 (1990)

# CCS Expressions, Finite State Processes, and Three Problems of Equivalence*

PARIS C. KANELLAKIS[†]

*Department of Computer Science, Brown University, Providence, Rhode Island 02912*

AND

SCOTT A. SMOLKA[‡]

*Department of Computer Science, SUNY, Stony Brook, New York 11794*

We examine the computational complexity of testing finite state processes for equivalence in Milner's Calculus of Communicating Systems (CCS). The equivalence problems in CCS are presented as refinements of the familiar problem of testing whether two nondeterministic finite automata (NFA) are equivalent, i.e., accept the same language. Three notions of equivalence proposed for CCS are investigated, namely, *observational equivalence*, *strong observational equivalence*, and *failure equivalence*. We show that observational equivalence can be tested in polynomial time. As defined in CCS, observational equivalence is the limit of a sequence of successively finer equivalence relations, $\approx_k$, where $\approx_1$ is nondeterministic finite automaton equivalence. We prove that, for each fixed $k$, deciding $\approx_k$ is PSPACE-complete. We show that strong observational equivalence can be decided in polynomial time by reducing it to *generalized partitioning*, a new combinatorial problem of independent interest. Finally, we demonstrate that testing for failure equivalence is PSPACE-complete, even for a very restricted type of process. © 1990 Academic Press, Inc.

## 1. INTRODUCTION

The *Calculus of Communicating Systems* (CCS) is an elegant formalism (Milner, 1980, 1983, 1984; Hennessy and Milner, 1985) for specifying and

* A preliminary version of this paper appeared as an extended abstract in the Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing, Montreal, Canada, August, 1983, pp. 228–240.

[†] Research supported partly by the NSF under Grant MCS-8210830 and partly by the ONR-DARPA under Grant N00014-83-K-014, ARPA Order 4786.

[‡] Research supported by the NSF under Grants NSF DCR-8505873 and CCR-8704309.

0890-5401/90 $3.00

verifying concurrent systems. Together with other new formalisms, such as *Communicating Sequential Processes* (Brookes *et al.*, 1984; Brookes, 1983) and *Algebra of Communicating Processes* (Bergstra and Klop, 1986, 1987), it represents the algebraic approach to modeling concurrent computation. One of the nicest features of CCS as an algebraic theory is that it closely parallels the theory of regular expressions and finite automata (Milner, 1984; Benson and Ben-Shachar, 1988).

CCS is based on two central ideas (Milner, 1980):

(1) The notion of *observationally equivalent processes*, i.e., processes that are indistinguishable to an observer. Equivalence classes of processes are the basic objects of CCS.

(2) The *definition* and *manipulation* of these basic objects using *algebraic operators*, such as $\cup$, $\cdot$, $*$, and $|$. Of these, *composition*, $|$, captures many features of interleaved concurrent computation and message passing.

In this paper we focus on the notion of *observational equivalence*. We examine it from the point of view of computational complexity and in the context of *finite state processes*. We also investigate other notions, such as *strong observational equivalence* (Milner, 1984), or simply *strong equivalence*, and *failure equivalence* (Brookes *et al.*, 1984), which have been proposed as alternatives to observational equivalence, perhaps more natural for certain types of processes.

In our exposition we stress the similarity of CCS to the classical theory of regular expressions in order to show how new problems, which are meaningful in the context of distributed computing, can be derived from classical problems. As in (Milner, 1984), we consider only a restricted set of algebraic operators of the calculus ($\cup$, $\cdot$, $*$), which produce the *star expressions* in CCS. These expressions are syntactically identical to regular expressions, but instead of having as semantics "sets of strings," their semantics is "equivalence classes of processes" (see Section 2.3).

We rigorously define all relevant features from CCS, in a fashion similar to Milner (1984). We consider (Section 2.1) finite state processes, which are slightly more general than the familiar nondeterministic finite automata (NFA) with empty moves (Hopcroft and Ullman, 1979). Also, some useful subsets of these *general* processes are considered. For example, a *standard* process is just an NFA; an *observable* process is a general process without $\tau$-moves, the CCS equivalent of empty moves; and a *restricted* process is a standard process with all states accepting.

We review three basic notions of equivalence, i.e., observational equivalence, strong equivalence, and failure equivalence, and the relationships between them. We treat these notions as refinements of the classical notion of NFA equivalence, which is based on accepting the same language

(Section 2.2). Finally (Section 2.3), we relate star expressions in CCS to regular expressions and pose the *CCS-equivalence* problem for star expressions. This is essentially a problem of testing processes for observational equivalence, strong equivalence, or failure equivalence, depending upon the chosen notion of equivalence.

We first deal with observable processes. Strong equivalence, which is observational equivalence for observable finite state processes, can be decided in polynomial time. We show this via a reduction to *generalized partitioning*, a new combinatorial problem of independent interest. In Kanellakis and Smolka (1983) and Smolka (1984), we developed an efficient algorithm for this problem by extending the technique of Hopcroft (1971) for minimizing the number of states of a deterministic finite automaton. Our time bounds have been recently improved by Paige and Tarjan (1987). Their algorithm yields an $O(m \log n + n)$ test of strong equivalence for $n$-state observable processes having $m$ transitions.

For general finite state processes, observational equivalence is the central notion of Milner (1980). We demonstrate an important property of observational equivalence: unlike classical NFA equivalence (Stockmeyer and Meyer, 1973), it is efficiently decidable in polynomial time. As defined in CCS, observational equivalence is the limit of a sequence of successively finer equivalence relations, $\approx_k$, where $\approx_1$ is NFA equivalence. We show that $\approx_k$ is PSPACE-complete for every fixed $k$, a complexity that disappears when we take limits. Our negative results hold even for the restricted and observable model.

For failure equivalence, an equivalence notion between $\approx_1$ and $\approx_2$, we show PSPACE-completeness, again even for the restricted and observable model.

Section 2 contains the model, Section 3 the analysis of strong equivalence and the generalized partitioning problem, Section 4 the analysis of observational equivalence, and Section 5 the analysis of failure equivalence.

We would like to point out that we examine only one critical feature of CCS, namely, the choice of equivalence notion. We do not discuss other important algebraic features of CCS, such as the various composition operators. However, we believe that we identify and answer some fundamental complexity questions in the calculus. Another paper (Kanellakis and Smolka, 1988) applies the theory presented here to the analysis of cooperation and antagonism in networks of communicating processes (i.e., with the composition operator). A preliminary version of the results of Kanellakis and Smolka (1988) and the results of this paper first appeared in Smolka (1984).

## 2. The Model

### 2.1. *Finite State Processes*

The basic building block of our model for distributed computation is the finite state process, which very much resembles the nondeterministic finite automaton (NFA) of Hopcroft and Ullman (1979).

Definition 2.1.1. A *Finite State Process* (FSP) is a sextuple $\langle K, p_0, \Sigma, \Delta, V, E\rangle$, where:

(1) $K$ is a finite set of *states*.

(2) $p_0 \in K$ is the *start state*.

(3) $\Sigma$ is a finite set of symbols called *actions*, and $\tau$ is a special symbol not in $\Sigma$ called the *unobservable action*.

(4) $\Delta \subseteq K \times (\Sigma \cup \{\tau\}) \times K$ is a relation called the *transition relation*.

(5) $V$ is a finite set of symbols different from $\Sigma \cup \{\tau\}$ called *variables*.

(6) $E \subseteq K \times V$ is a relation called the *extension relation*.

We use the notation $\Delta(q) = \{\langle a, q'\rangle \mid \langle q, a, q'\rangle \in \Delta\}$ for the *transitions from* $q$, $E(q) = \{x \mid \langle q, x\rangle \in E\}$ for the *extensions of* $q$, and $\Delta(q, a) = \{q' \mid \langle q, a, q'\rangle \in \Delta\}$ for the *destinations of* $q$ *via action* $a$. We also write $q \rightarrow^{a} q'$ when $\langle q, a, q'\rangle \in \Delta$; thus, $\rightarrow$ is a $\Sigma$-indexed family of binary relations over $K$.

An FSP can be represented by a labeled directed graph whose nodes are the states and which has labels on the arcs (i.e., the transitions) from $\Sigma \cup \{\tau\}$, and labels on the nodes from $2^V$ (i.e., the extensions). The extensions are used in Milner (1984) to represent different flavors of acceptance; they are the only difference from the classical notion of an NFA with empty transitions (Hopcroft and Ullman, 1979). Extensions do not affect most of our combinatorial analysis, but are included to maintain consistency with the exposition of Milner (1984). Note also that we use different symbols for $\tau$ and the empty string $\varepsilon$, because of the particular role of $\tau$ as the unobservable action in distributed computation (Milner, 1980; Bergstra and Klop, 1987).

We will refer to the model of computation in Definition 2.1.1 as the *general* model of FSPs. We also consider several specialized versions of the general model. The *observable* model (Milner, 1984) is obtained from the general model by not allowing $\tau$-transitions, and, if in this case there is exactly one transition for each symbol in $\Sigma$, we have the *deterministic* model. The *standard* model is derived from the general model by restricting $V$ to be the set $\{x\}$. In this model, an FSP can be viewed as a classical NFA with empty transitions, where a state $q$ is accepting if $E(q) = \{x\}$ and nonaccepting if $E(q)=\varnothing$. If in the standard model, for all states $p$, $E(p)=\{x\}$, we have the *restricted* model. Thus all states of a restricted FSP are accepting, but states could be missing transitions.

Nontrivial subsets of the restricted model are the *restricted observable* model—called *restricted observable unary* (r.o.u.) when there is only one action in $\Sigma$—and the model where the FSP is a *finite tree* rooted at $p_0$. Subsets *standard observable* and *standard observable unary* (s.o.u.) of the standard model are defined analogously.

The hierarchy of models is depicted in Fig. 1a. Every one of these models corresponds to some nontrivial case in our exposition. Examples of such FSPs are presented in Fig. 1b; if the extension label is missing from a node it is assumed to be $\varnothing$. A guide to the various models of FSPs is included in Appendix A.

We will always deal with *states of FSPs*, e.g., equivalent states, and the processes to which these states belong will be clearly defined from the context.

Let $s\in\Sigma^*$ and $p$, $p'$ be states of an FSP. Also, let $\varepsilon$ denote the empty string. We say that $p\Rightarrow^{\varepsilon}p'$ when there is a sequence of $k$ arcs in the FSP's graph from $p$ to $p'$ whose string of labels is $\tau^k$, where $k\geqslant 0$. (Obviously always $p\Rightarrow^{\varepsilon}p$.) If $s=\sigma_1\sigma_2\cdots\sigma_n$, $\sigma_i\in\Sigma$, $1\leqslant i\leqslant n$, we say that $p\Rightarrow^{s}p'$ when there is a sequence of $k_0+k_1+\cdots+k_n+n$ arcs in the FSP's graph from $p$ to $p'$ whose string of labels is $\tau^{k_0}\sigma_1\tau^{k_1}\sigma_2\cdots\sigma_n\tau^{k_n}$, where $k_0, k_1, ..., k_n\geqslant 0$. We refer to $p'$ as an *s-derivative* of $p$.

In the *restricted* model of FSPs, the only feature that distinguishes states is the absence of certain transitions. This concept is formalized in Brookes *et al.* (1984) as the *failures*$(p)$ for state $p$ of a restricted FSP. We say that $\neg(p\Rightarrow^{s})$ when there is no $p'$ such that $p\Rightarrow^{s}p'$.

$$\textit{failures}(p)=\{(s,Z)\mid s\in\Sigma^*, Z\subseteq\Sigma \text{ such that}$$
$$\exists p'\in K\colon p\overset{s}{\Longrightarrow}p' \text{ and } \forall z\in Z\colon \neg(p'\overset{z}{\Longrightarrow})\}.$$

For example, assuming $\Sigma=\{a,b,c\}$, the failures for the start state of the finite tree process of Fig. 1b are

$$\{\varepsilon\}\times 2^{\{b,c\}}\cup\{a\}\times 2^{\{a\}}\cup\{ab\}\times 2^{\Sigma}\cup\{ac\}\times 2^{\Sigma}.$$

## 2.2. *Equivalences of FSP States*

*The essential new idea in CCS is a new notion of equivalence between states of FSPs.* A number of "candidates" for the correct notion of equivalence have been proposed and investigated (e.g., Milner, 1980, 1983, 1984; Brookes, 1983; Brookes *et al.*, 1984; Hennessy, 1985; de Nicola and Hennessy, 1984). We will deal with three such notions: *observational equiva-*

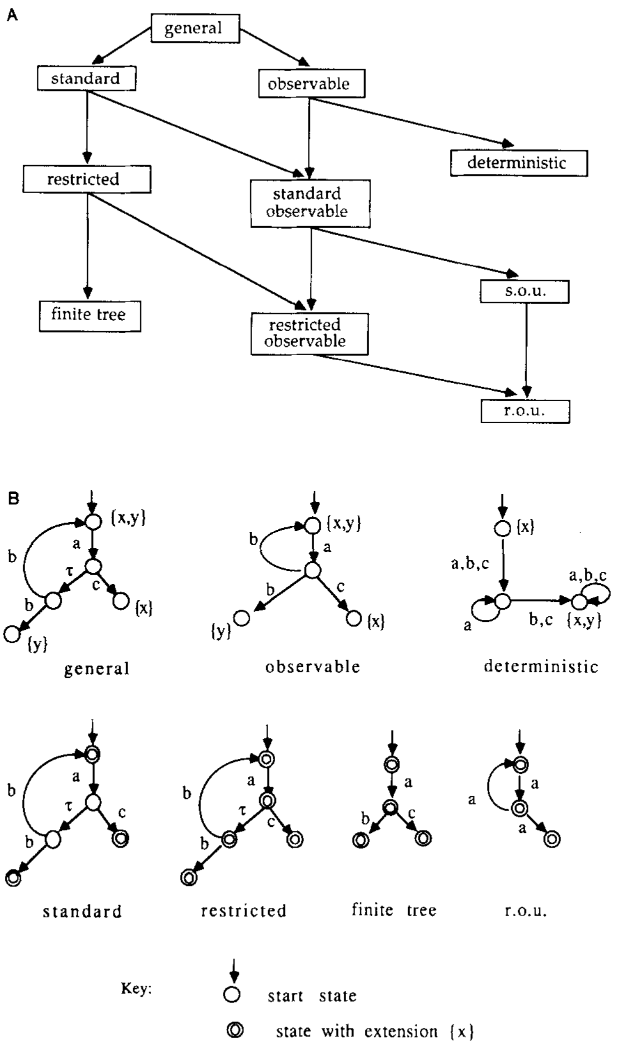


FIG. 1. (A) Hierarchy of FSP types. (B) Examples of FSPs.

*lence*, *strong observational equivalence*, and *failure equivalence*. The expositions of Milner (1980) and Brookes *et al.* (1984) are, to a great extent, devoted to establishing the practical relevance of such choices in the context of distributed computation.

In general, for two FSP states $p$, $q$ to be equivalent, it is not enough to say that they represent start states of NFAs accepting the same language, as in the classical case. In particular:

DEFINITION 2.2.1 (Milner, 1980). Let $p, q$ be states of general FSPs which have the same $\Sigma$ and $V$. We define $p, q$ to be *k-observationally equivalent* ($p \approx_k q$) if:

(1) $E(p) = E(q)$, when $k = 0$. Otherwise, when $k > 0$,

(2) For every $s \in \Sigma^*$,

(i) if $p \Rightarrow^s p_1$ then $(\exists q_1 : q \Rightarrow^s q_1$ and $p_1 \approx_{k-1} q_1)$,

(ii) if $q \Rightarrow^s q_2$ then $(\exists p_2 : p \Rightarrow^s p_2$ and $q_2 \approx_{k-1} p_2)$.

States $p, q$ are *observationally equivalent* ($p \approx q$) if $p \approx_k q$ for all $k \geqslant 0$.

Limited observational equivalence is similar to observational equivalence except that only strings of length zero or one are considered.

DEFINITION 2.2.2. Let $p, q$ be states of general FSPs which have the same $\Sigma$ and $V$. We define $p, q$ to be *k-limited observationally equivalent* ($p \simeq_k q$) if:

(1) $E(p) = E(q)$, when $k = 0$. Otherwise, when $k > 0$,

(2) For every $\sigma \in \Sigma \cup \{\varepsilon\}$,

(i) if $p \Rightarrow^\sigma p_1$ then $(\exists q_1 : q \Rightarrow^\sigma q_1$ and $p_1 \simeq_{k-1} q_1)$,

(ii) if $q \Rightarrow^\sigma q_2$ then $(\exists p_2 : p \Rightarrow^\sigma p_2$ and $q_2 \simeq_{k-1} p_2)$.

States $p, q$ are *limited-observationally equivalent* ($p \simeq q$) if $p \simeq_k q$ for all $k \geqslant 0$.

Strong equivalence and failure equivalence are defined for less general models of FSPs.

DEFINITION 2.2.3. Let $p, q$ be states of observable FSPs which have the same $\Sigma$ and $V$. We define $p, q$ to be *strongly equivalent* ($p \sim q$) if $p \simeq q$.

DEFINITION 2.2.4 (Brookes *et al.*, 1984). Let $p, q$ be states of restricted FSPs which have the same $\Sigma$ and $V$. We define $p, q$ to be *failure equivalent* ($p \equiv q$) if

$$failures(p) = failures(q).$$

It is easy to verify that $k$-observational equivalence, observational equivalence, $k$-limited observational equivalence, limited observational equivalence, strong equivalence, and failure equivalence are true equivalence relations on FSP states; namely, they are reflexive, symmetric, and transitive.

DEFINITION 2.2.5. Let $\Lambda \subseteq \Sigma^*$ be a set of strings. We call a relation **R** between states of FSPs having the same $\Sigma$ and $V$ a *$\Lambda$-fixed-point* when $p \mathbf{R} q$ implies that

(1) $E(p) = E(q)$.

(2) For every $s \in \Lambda$,

(i) if $p \Rightarrow^s p_1$ then $(\exists q_1 : q \Rightarrow^s q_1$ and $p_1 \mathbf{R} q_1)$,

(ii) if $q \Rightarrow^s q_2$ then $(\exists p_2 : p \Rightarrow^s p_2$ and $q_2 \mathbf{R} p_2)$.

We will use $\Sigma^*$-fixed-points and $(\Sigma \cup \{\varepsilon\})$-fixed-points in order to investigate the relationship between $\approx$ and $\simeq$.

PROPOSITION 2.2.1. *For FSP states p, q in the* general *model,*

(a) *$\simeq$ is a $(\Sigma \cup \{\varepsilon\})$-fixed-point.*

(b) *$\simeq$ is a $\Sigma^*$-fixed-point.*

(c) *$p \simeq q$ iff $p \approx q$.*

The arguments in the proof of Proposition 2.2.1 are simple generalizations of the arguments in Milner (1980, Theorems 5.6 and 7.2). For completeness of exposition, we include a detailed proof in Appendix B. The use of the pigeonhole principle in part (a) is critical and, as shown in Sanderson (1982), these fixed-point properties do not necessarily hold for infinite state processes.

For observable FSPs, Milner refers to a binary relation which is a $\Sigma$-fixed-point as a *strong bisimulation* (Milner, 1983, 1984). Note that for the observable case, $(\Sigma \cup \{\varepsilon\})$- and $\Sigma$-fixed-points are the same. He then shows that strong equivalence ($\sim$) is the largest, under set inclusion, strong bisimulation. As such, strong equivalence is often referred to as "strong bisimulation equivalence." Similarly, using the Knaster–Tarski fixed-point theorem, Definitions 2.2.1–2.2.2, and Proposition 2.2.1 we have that:

PROPOSITION 2.2.2. *In the* general *model,*

$$\approx \; = \; \simeq \; = \bigcup \{\mathbf{R} \mid \mathbf{R} \text{ is a } \Sigma \cup \{\varepsilon\}\text{-fixed-point}\}$$

*and is the largest, under set inclusion, $\Sigma \cup \{\varepsilon\}$-fixed-point.*

Let $p, q$ represent start states of NFAs. Then we will denote the languages accepted by these NFAs as $L(p)$, $L(q)$, respectively. Recalling that $\equiv$ denotes failure equivalence, we have for the restricted model:

PROPOSITION 2.2.3 (Brookes, 1983). *For FSP states p, q in the* restricted *model,*

(a) *$p \approx_2 q$ implies $p \equiv q$ implies $p \approx_1 q$.*

(b) *$p \approx_1 q$ iff $L(p) = L(q)$.*

In the deterministic case the equivalence relations $\approx_k$, $k \geqslant 1$, collapse to $\approx_1$; this is an easy consequence of determinism.

Proposition 2.2.4. *For FSP states $p$, $q$ in the* deterministic *model,*

(a) *$p \approx_1 q$ iff $p \approx q$* (or *$p \simeq q$ or $p \sim q$*).

(b) *$p \approx_1 q$ iff $L(p) = L(q)$ (for the* deterministic standard *model*).

In this paper, we study the complexity of testing whether two states are observationally equivalent in the general model, strongly equivalent in the observable model, or failure equivalent in the restricted model.

We should note that even in the r.o.u. model, the equivalence notions $\approx_k$, $\equiv$, and $\approx$ (or $\simeq$, or $\sim$) are different. This is illustrated by the examples of Fig. 2. A guide to the various equivalence relations for FSPs is included in Appendix A.

## 2.3. *Regular Expressions for Languages vs Star Expressions for CCS*

The theory of *CCS expressions* is developed in Milner (1984). Their syntax is very similar to that of regular expressions for languages. The basic novelty is that the semantics of a CCS expression is no longer a language but a class of observable FSPs with strongly equivalent start states.

A particularly interesting class of these expressions are the *star expressions*. In this section, we describe star expressions in detail and refer to Milner (1984) for the complete definition of CCS expressions. The rationale for this is that star expressions use the familiar $\cup$, $\cdot$, $*$ symbols, with new semantics. They provide the link between the algebraic theories of CCS and regular sets.

Definition 2.3.1. The *syntax of star expressions over $\Sigma$* is the same as that of regular expressions over $\Sigma$. The *semantics of star expression $r$* is the class of observable and standard FSPs whose start states are strongly equivalent to $p$, the start state of the *representative* FSP of $r$. The

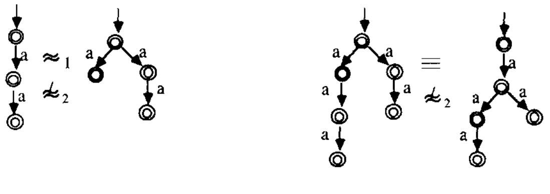


Fig. 2. Example r.o.u. FSPs distinguishing the various equivalences.

representative FSP of $r$ is the NFA, without empty moves, $P = \langle K, p, \Sigma, \Delta, \{x\}, E\rangle$, constructed inductively as follows:

$\mathbf{r} = \boldsymbol{\emptyset}$: $P = \langle \{p\}, p, \Sigma, \emptyset, \{x\}, \emptyset\rangle$.

$\mathbf{r} = \mathbf{a}$: $P = \langle \{p, q\}, p, \Sigma, \{\langle p, a, q\rangle\}, \{x\}, \{(q, x)\}\rangle$.

Let $r_1, r_2$ be star expressions having representative FSPs $P_i = \langle K_i, p_i, \Sigma, \Delta_i, \{x\}, E_i\rangle$, $i = 1, 2$, such that $K_1 \cap K_2 = \emptyset$. Then

$\mathbf{r} = \mathbf{r}_1 \cup \mathbf{r}_2$: $P = \langle K_1 \cup K_2 \cup \{p\}, p, \Sigma, \Delta_1 \cup \Delta_2 \cup \Delta', \{x\}, E_1 \cup E_2 \cup E'\rangle$, where $p$ is a new state not in $K_1 \cup K_2$, and $\Delta' = \{p\} \times (\Delta_1(p_1) \cup \Delta_2(p_2))$, $E' = \{p\} \times (E_1(p_1) \cup E_2(p_2))$.

$\mathbf{r} = \mathbf{r}_1 \cdot \mathbf{r}_2$: $P = \langle K_1 \cup K_2, p, \Sigma, \Delta_1 \cup \Delta_2 \cup \Delta', \{x\}, E_2\rangle$, where $p = p_1$, and $\Delta' = \{q \in K_1 \mid E_1(q) = \{x\}\} \times \Delta_2(p_2)$.

$\mathbf{r} = \mathbf{r}_1^*$: $P = \langle K_1 \cup \{p\}, p, \Sigma, (\{p\} \times \Delta_1(p_1)) \cup \Delta^+, \{x\}, E_1 \cup \{(p, x)\}\rangle$,
where $p$ is a new state not in $K_1$,
and $\Delta^+(q) = \Delta_1(q) \cup \Delta_1(p_1)$ if $E_1(q) = \{x\}$
$= \Delta_1(q)$ otherwise.

Intuitively, the semantics of a star expression $r$ is the class of FSPs whose start states are equivalent to the start state of the representative FSP of $r$. The representative FSP is the one constructed inductively in Definition 2.3.1 and illustrated in Fig. 3. This definition follows closely the classical construction used in showing that the language denoted by regular expression $r$ is accepted by some NFA. Since we are dealing with strong equivalence classes of observable FSPs, the representative FSP is constructed so that it too is observable, i.e., free of $\tau$-actions. In Milner (1984), it is shown that using strong equivalence as the equivalence notion makes the semantics independent of the representative FSP chosen.

The *CCS equivalence* problem is: "Given two CCS expressions, do they have the same semantics?" This parallels the equivalence problem for regular expressions (Hopcroft and Ullman, 1979; Stockmeyer and Meyer, 1973). Let the *length* of a star expression $r$ be the number of symbols in the string $r$. Using Definition 2.3.1 we have:

LEMMA 2.3.1. *Let $r$ be a star expression of length $n$ over a fixed alphabet $\Sigma$. Then the representative FSP of $r$ is observable and standard; it has $O(n)$ states and $O(n^2)$ transitions, and can be constructed in $O(n^2)$ time.*

In Definition 2.3.1 we described the syntax and semantics of star expressions. The CCS expressions of Milner (1984) are slightly more general, because of the presence of extensions. Their semantics are strong equivalence classes of observable, though not necessarily standard, FSPs, and a

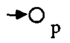
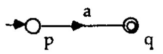
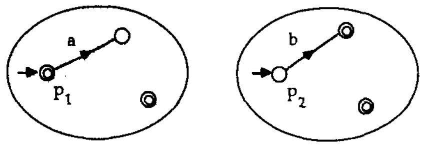
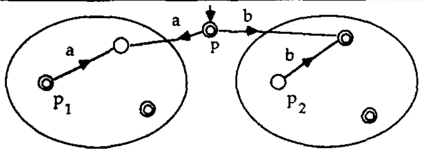
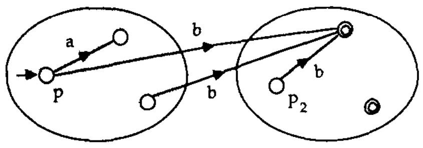
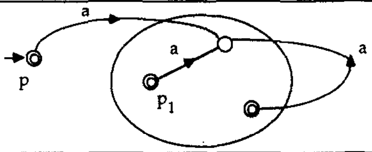
| Syntax | Semantics (the representative FSP) |
|---|---|
| $\emptyset$ |  |
| a |  |
| Let $r_1, r_2$ have semantics |  |
| $(r_1 \cup r_2)$ |  |
| $(r_1 \cdot r_2)$ |  |
| $(r_1)^*$ |  |

FIG. 3. Construction procedure for representative FSP.

straightforward generalization of Lemma 2.3.1 also holds for CCS expressions. Thus we have that the CCS equivalence problem is in essence one of testing FSPs for observational equivalence, strong equivalence, or failure equivalence.

Finally, let us briefly mention a number of interesting connections between CCS expressions and regular expressions from Milner (1984).

(1) Every observable FSP is a member of some set of observable FSPs that is the semantics, i.e., strong equivalence class, of a CCS expression.

(2) There is a complete inference system for identities of CCS expressions paralleling that of Salomaa (1986) for regular expressions.

(3) The significant algebraic properties that regular expressions have and star expressions lack are the following two identities. Let $r, s, t$ be arbitrary regular expressions. Then: $r \cdot (s \cup t) = r \cdot s \cup r \cdot t$ and $r \cdot \emptyset = \emptyset$.

## 3. Strong Equivalence is Efficiently Decidable

The problem of language equivalence of two finite state automata of size $N$, i.e., whether they accept the same language, has received a great deal of attention in the literature. For DFAs there is an $O(N\,G(N))$ algorithm that uses UNION-FIND (Aho *et al.*, 1974, Sect. 4.8), and for NFAs the problem has been shown to be PSPACE-complete (Stockmeyer and Meyer, 1973). Also, the problem of minimizing the states of a DFA of size $N$ has an elegant $O(N \log N)$ solution, and is related to a combinatorial partitioning problem (Aho *et al.*, 1974, Sect. 4.13; Hopcroft, 1971).

For testing strong equivalence of states of deterministic FSPs, the above techniques for DFAs are directly applicable (see Proposition 2.2.4(b)). For the larger class of observable FSPs, strong equivalence of states can still be tested efficiently. In this case, unfortunately, the UNION-FIND technique does not lead to an efficient algorithm because of possible multiple transitions for one symbol of the alphabet. However, we can show that strong equivalence of states can be tested by solving the following partitioning problem, which is also of independent interest.

*Generalized Partitioning*

*Input*: A set $S$, an initial partition of $S$ into disjoint blocks $\pi = \{B_1, B_2, ..., B_p\}$, and $k$ functions $f_l \colon S \to 2^S$, $1 \leqslant l \leqslant k$.

*Output*: A partition of $S$ into disjoint blocks $\pi' = \{E_1, E_2, ..., E_q\}$, such that:

(1) $\pi'$ is *consistent* with $\pi$, i.e., each $E_i$ is a subset of some $B_j$.

(2) For $a, b$ in block $E_i$, any block $E_j$, $1 \leqslant i, j \leqslant q$, and any function $f_l$, $1 \leqslant l \leqslant k$:

$$f_l(a) \cap E_j \neq \varnothing \qquad \text{iff} \quad f_l(b) \cap E_j \neq \varnothing.$$

(3) $\pi'$ is the coarsest such partition, i.e., has the fewest blocks.

The generalized partitioning problem is well-posed, i.e., there always exists a coarsest partition with properties (1) and (2). To see this, consider the lattice of partitions of the set $S$ under the ordering: $\pi_1 \leqslant \pi_2$ if every block of partition $\pi_1$ is contained in some block of partition $\pi_2$. Properties (1) and (2) of the generalized partitioning problem can be used to define a monotone function on this lattice whose greatest fixed-point exists and is unique, by the Knaster–Tarski theorem. This greatest fixed-point is the coarsest partition satisfying (1) and (2).

Intuitively, the initial partition $\pi$ is refined into the final partition $\pi'$, in the coarsest fashion possible, so that each $f_l$ induces a mapping from blocks of $\pi'$ to sets of blocks of $\pi'$. It is easy to see that the generalized partitioning problem is different from that of minimizing the states of a DFA.

Obviously, each function $f_l$ can be represented as a directed graph with node set $S$ and arcs $(i, j)$ iff $j$ is in $f_l(i)$. The size of an instance of generalized partitioning is $(n, m)$, where the cardinality of set $S$ (denoted $|S|$) is $n$ and the total number of arcs in the corresponding $k$ graphs is $m$. In the deterministic case, we have $f_l$: $S \rightarrow S$, for each $l$, and $m = k \cdot n$.

LEMMA 3.1. *Let $p, q$ be states of observable FSPs having a total of $n$ states and a total of $m$ transitions. We can test whether $p \sim q$ by reducing this problem, in $O(n+m)$ time, to a generalized partitioning problem of size at most $(n, m)$.*

*Proof.* Let $p, q$ be states of the observable FSP $\langle K, p_0, \Sigma, \Delta, V, E \rangle$; the proof is similar if $p, q$ belong to two distinct observable FSPs having the same $\Sigma$ and $V$. We can construct an instance of generalized partitioning in $O(n+m)$ time as follows. The set $S$ is identified with $K$. For the initial partition $\pi$, two states $p$ and $q$ are in the same block iff they have the same extensions, i.e., $E(p) = E(q)$. Finally, for each $\sigma \in \Sigma$, there is a function $f_\sigma$ corresponding to the restriction of $\Delta$ to $\sigma$, i.e., $f_\sigma(p) = \{p' \mid p' \in \Delta(p, \sigma)\}$. We are left to show that:

*Claim.* $p \sim q$ iff $p$ and $q$ belong to the same block of $\pi'$.

The relation $\sim$ is an equivalence relation on states. It therefore defines a partition $\pi_\sim$. By Proposition 2.2.2, $\sim$ is a $\Sigma$-fixed-point. From Definition 2.2.5 it follows that $\pi_\sim$ satisfies properties (1) and (2) of the generalized partitioning problem. Assume that $\pi_\sim$ is not the coarsest partition satisfying these properties. Then the coarsest such partition would give us a $\Sigma$-fixed-point larger than $\sim$, contradicting Proposition 2.2.2. This completes the proof of the claim and hence the lemma. ▮

An obvious solution to the generalized partitioning problem is, starting from $\pi$, refine the blocks of the partition by the following method. Let $B_i$ be a block. For each of the $k$ functions $f_l$, examine $f_l(a) \subseteq S$, for each $a$ in $B_i$. We can think of $f_l(a)$ as denoting a set of blocks, i.e., those blocks such that each one contains some element of $f_l(a)$. Now we partition $B_i$ so that two elements $a$ and $b$ are put in the same block if and only if $f_l(a)$ and $f_l(b)$ denote the same set of blocks. We will refer to this method as the *naive method.*

LEMMA 3.2. *The* naive method *correctly solves an instance of the generalized partitioning problem of size $(n, m)$, and can be implemented in $O(nm)$ time.*

*Proof.* It is easy to see that the method described above gives the correct output partition. However, the $O(mn)$ implementation is less obvious. We perform $O(n)$ iterations since there can be at most $n$ blocks. In each iteration the lexicographic sorting method from Aho *et al.* (1974) is used and takes $O(n+m)$ time. Also, simple examples show that this bound is tight. ▮

The time bound of the naive method can be improved upon substantially. In Kanellakis and Smolka (1983), a generalization of the divide-and-conquer method of Hopcroft (1971) was presented for the case of bounded fanout, i.e., for all $a$ in $S$, $|f_l(a)| \leqslant c$, for some constant $c$. This case corresponds to FSPs that have at most $c$ transitions out of any state for each symbol of the action alphabet. The algorithm of Kanellakis and Smolka (1983) runs in $O(c^2 \cdot n \log n)$ time. Recently, Paige and Tarjan (1987) have developed an algorithm that solves the generalized partitioning problem (which they refer to as "relational coarsest partitioning") of size $(n, m)$ in time $O(m \log n + n)$. This resolves an open problem in Kanellakis and Smolka (1983). Therefore from Lemma 3.1 and Paige and Tarjan (1987) we have that:

THEOREM 3.1. *Let $p, q$ be states of observable FSPs having a total of $n$ states and $m$ transitions. Strong equivalence of $p$ and $q$ can be decided in $O(m \log n + n)$ time.*

## 4. THE COMPLEXITY OF OBSERVATIONAL EQUIVALENCE

In this section, we examine the complexity of testing for observational equivalence. This equivalence notion may be used for FSP states from all of our models. The upper bounds presented in this section hold even for FSPs of the general kind, and the lower bounds even for FSPs that are restricted and observable, and in some cases r.o.u.

We begin by presenting two lemmas that will be used in the proof of Theorem 4.1, which contains the main results of the section. To concisely state the first of these lemmas, we use the syntax and semantics of the star expressions given in Definition 2.3.1. For any standard, observable FSP state $p$, we alternatively view $p$ as a star expression whose representative FSP is the one having start state $p$. We will limit our usage of this notation to restricted and observable FSPs. So, for example, the star expression $a \cdot p$ denotes the restricted and observable FSP consisting of an $a$-transition into state $p$. The second lemma is due to Chandra and Stockmeyer (1982) and gives a sharper lower bound on language equivalence of NFAs.

LEMMA 4.1. *In the restricted and observable model, for any* $k \geqslant 0$,

$$p \approx_k q \qquad \textit{iff} \quad (p \cup q \approx_k p \text{ and } p \cup q \approx_k q).$$

*Proof. only if*: Assume $p \approx_k q$ and consider all strings $s \in \Sigma^*$. We show the first conjunct $p \cup q \approx_k p$. Let $r$ be an $s$-derivative of $p \cup q$. Case (1): $r$ is an $s$-derivative of $q$. The fact that $p \approx_k q$ guarantees we can find a suitable $s$-derivative $p'$ of $p$, i.e., one for which $p' \approx_{k-1} r$. Case (2): $r$ is an $s$-derivative of $p$, and the result is obvious. Showing the second condition of $k$-observational equivalence in which the $s$-derivatives of $p$ are first considered is also obvious. The other conjunct $p \cup q \approx_k q$ is proved similarly.

*if*: Assume $p \cup q \approx_k p$ and $p \cup q \approx_k q$, and consider all strings $s \in \Sigma^*$. We show that $p \approx_k q$. Let $p'$ be an $s$-derivative of $p$. The fact that $p \cup q \approx_k q$ guarantees we can find a suitable $s$-derivative $q'$ of $q$. The proof of the second condition of $k$-observational equivalence is symmetric. ▌

LEMMA 4.2 (Chandra and Stockmeyer, 1982). *In the restricted and observable model, deciding* $p \approx_1 q$ *is PSPACE-complete.*

*Proof.* Membership in PSPACE for this problem is immediate since a restricted and observable FSP is also a standard FSP. To show PSPACE-hardness, let $p$ be a state of a standard FSP. We reduce the PSPACE-complete problem of whether $L(p) = \Sigma^*$ (Stockmeyer and Meyer, 1973) to the corresponding problem for restricted observable FSPs. Consider the standard FSP $M = \langle K, p_0, \Sigma, \Delta, \{x\}, E\rangle$, having the set of accept states $F = \{p_f \in K \mid E(p_f) = \{x\}\}$. By a simple reduction whose details we do not present, assume that $\Sigma = \{a, b\}$, and that $M$ is observable with both $a$- and $b$-transitions leaving each state. Transform $M$ to the restricted observable FSP $M' = \langle K', p'_0 = p_0, \Sigma, \Delta', \{x\}, E'\rangle$ as follows (see Fig. 4):

$$K' = K \cup \{p_{\text{trap}}\} \cup \{p_\delta \mid \delta \in \Delta\},$$

where $p_{\text{trap}}$ and $p_\delta$, $\delta \in \Delta$, are *new* states not in $K$;

$$E' = K' \times \{x\}, \text{ i.e., every state of } M' \text{ is accepting;}$$

$$\begin{aligned} \Delta' = {} & \{\langle p_f, a, p_{\text{trap}}\rangle \mid p_f \in F \text{ is an accept state of } M\} \\ & \cup \{\langle p, b, p_\delta\rangle, \langle p_\delta, \sigma, q\rangle \mid \delta = \langle p, \sigma, q\rangle \text{ is in } \Delta\} \\ & \cup \{\langle p_{\text{trap}}, a, p_{\text{trap}}\rangle, \langle p_{\text{trap}}, b, p_{\text{trap}}\rangle\}. \end{aligned}$$

We now show that $L(p_0) \neq \Sigma^*$ iff $L(p'_0) \neq \Sigma^*$. For the "only if" direction, suppose $s \notin L(p_0)$ with $s = \sigma_1 \sigma_2 \cdots \sigma_n$, $n \geqslant 0$, and all prefixes of $s$ are in $L(p_0)$. Then $b\sigma_1 b\sigma_2 \cdots b\sigma_n a \notin L(p'_0)$ for this would otherwise mean that $s$ can take $p_0$ to an accept state.

For the "if" direction, suppose $s \notin L(p'_0)$ and all prefixes of $s$ are in $L(p'_0)$.

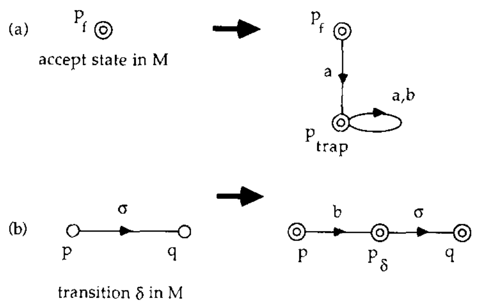


FIG. 4. Reduction of Lemma 4.2.

Then $s$ can be written as $b\sigma_1 b\sigma_2 \cdots b\sigma_k a$, since if an "$a$" appeared in any of the first $k$ odd positions then $s$ would be in $L(p_0')$. It follows that $\sigma_1\sigma_2 \cdots \sigma_k \notin L(p_0)$ since otherwise $s \in L(p_0')$. ∎

THEOREM 4.1. *Let $p$, $q$ be FSP states and assume that the FSPs to which these states belong have a total of $n$ states and $m$ transitions.*

(a) *In the general model, $p \approx q$ can be decided in $O(n^2 m \log(n) + mn^{2.376})$ time.*

(b) *If $p$ and $q$ are restricted and observable, then $p \approx_k q$, for any fixed $k \geqslant 1$, is PSPACE-complete.*

(c) *If $p$ and $q$ are r.o.u. states, then $p \approx_k q$, for any fixed $k \geqslant 2$, is co-NP-complete, and decidable in linear time for $k = 1$.*

*Proof.* (a) We know from Proposition 2.2.1(c) that $p \approx q$ iff $p \simeq q$. By the definition of $\simeq$ (Definition 2.2.2), we see that we have a problem which is very similar to that of Section 3; our only additional consideration is $\tau$-transitions. In particular, let $p$, $q$ be states of the general FSP $P = \langle K, p_0, \Sigma, \Delta, V, E\rangle$. The problem of deciding $p \approx q$ is reducible to deciding strong equivalence as follows. Compute the transition relation $\bar{\Delta}$ of the observable FSP $\bar{P} = \langle K, p_0, \Sigma \cup \{\varepsilon\}, \bar{\Delta}, V, E\rangle$ such that

$$\bar{\Delta} = \{\langle p, \sigma, q\rangle \mid p \overset{\sigma}{\Longrightarrow} q \text{ in } P, \sigma \in \Sigma \cup \{\varepsilon\}\}.$$

To do this, we first note that, for each $\sigma \in \Sigma$, $p \Rightarrow^{\sigma} q$ in $P$ iff there exist a $p'$ and $p''$ such that $p \Rightarrow^{\varepsilon} p' \rightarrow^{\sigma} p'' \Rightarrow^{\varepsilon} q$. Thus, $\bar{\Delta}$ can be determined by the following procedure:

(1) Compute the adjacency matrix $M_\varepsilon$ for the binary relation $\Rightarrow^{\varepsilon}$ over $K$, i.e., the reflexive transitive closure of $\Delta$ with respect to the symbol $\tau$; and then

(2) For each $\sigma \in \Sigma$, compute the matrix products $M_{\varepsilon} \cdot M_{\sigma} \cdot M_{\varepsilon}$, where $M_{\sigma}$ is the adjacency matrix of the binary relation $\rightarrow^{\sigma}$. The resulting matrix is the adjacency matrix of the binary relation $\{(p, q) \mid \langle p, \sigma, q \rangle \in \bar{\Delta}\}$.

Note that the size of $\bar{\Delta}$ is $O(n^2m)$, as there can be at most $m$ distinct symbols $\sigma$ labelling the transitions of $\Delta$. By the definition of $\bar{\Delta}$ and Proposition 2.2.1(c), we have that

$$p \approx q \text{ in } P \quad \text{iff} \quad p \sim q \text{ in } \bar{P}.$$

Using Lemma 3.1, we can now directly apply the algorithm of Paige and Tarjan (1987) for generalized partitioning to the states of $\bar{P}$ to obtain an algorithm for observational equivalence whose complexity is $O(n^2m \log(n) + mn^{\alpha})$. Here $O(n^{\alpha})$ is the time needed to perform transitive closure on a directed graph having $n$ nodes. The smallest such $\alpha$ currently known is 2.376 (Coppersmith and Winograd, 1987).

If only a constant number of different symbols $\sigma$ label the transitions of $\Delta$, then the size of $\bar{\Delta}$ is $O(n^2)$, and the term $n^{\alpha}$ will dominate the time complexity of deciding observational equivalence.

(b) Let $p$ be a state of a standard FSP (an NFA) and let $L(p)$ be the language accepted by this NFA with start state $p$. We know from Proposition 2.2.3(b) that $p \approx_1 q$ iff $L(p) = L(q)$. Thus, it is PSPACE-complete (Stockmeyer and Meyer, 1973) to decide $\approx_1$ in the standard, observable model. From Lemma 4.2 (Chandra and Stockmeyer, 1982), we have that deciding $\approx_1$ is PSPACE-complete even for restricted and observable FSPs. Obviously, testing for $\approx_0$ is trivial.

Membership in PSPACE for $\approx_k$, $k \geqslant 1$, can be established by a reduction to the classical problem of NFA equivalence. Let $p$ and $q$ be states of general FSPs such that $p \in K$ and $q \in K'$. Let $\{B_i \mid 1 \leqslant i \leqslant l\}$ be the partition induced by $\approx_k$ over $K \cup K'$, i.e., each $B_i$ is an equivalence class of $K \cup K'$ with respect to $\approx_k$. Then we can restate Definition 2.2.1, the definition of $\approx_{k+1}$, as

$$p \approx_{k+1} q \qquad \text{iff} \quad \forall i, 1 \leqslant i \leqslant l, L_i(p) = L_i(q),$$

where $L_i(p)$ denotes the language accepted by the standard FSP having start state $p$ and accept states $K \cap B_i$. Similarly, $L_i(q)$ denotes the language accepted by the standard FSP having start state $q$ and accept states $K' \cap B_i$, and we have the reduction.

We show that deciding $\approx_k$, for any fixed $k \geqslant 1$, is PSPACE-hard in the restricted, observable model. The following technique is used which allows us to inductively reduce the problem of $\approx_1$ to $\approx_k$:

Given two FSP states $p, q$ we construct two states $p', q'$ of new FSPs such that

$$p \approx_k q \qquad \text{iff} \quad p' \approx_{k+1} q', \text{for } k \geqslant 1.$$

The reduction uses one symbol, the symbol $a$, from the alphabet $\Sigma$, and is illustrated in Fig. 5a. We can once again use the syntax of star expressions to formalize the reduction in a concise manner:

$$p' = a \cdot (p \cup q)$$

$$q' = (a \cdot p) \cup (a \cdot q).$$

*if*: We prove the contrapositive, namely, $p \not\approx_k q$ implies $p' \not\approx_{k+1} q'$. Consider the $a$-derivatives of $p'$ and $q'$. State $p'$ has only one, viz., $p \cup q$, while $q'$ has both $p$ and $q$ as $a$-derivatives. Assuming $p \not\approx_k q$, we have, by Lemma 4.1, that either $p \cup q \not\approx_k p$ or $p \cup q \not\approx_k q$, and thus $p' \not\approx_{k+1} q'$.

*only if*: We prove the contrapositive, namely, $p' \not\approx_{k+1} q'$ implies $p \not\approx_k q$. Assuming $p' \not\approx_{k+1} q'$, then the string that distinguishes $p'$ and $q'$ must consist of the single character $a$. This is because for any longer string $s$, $p'$ and $q'$ have identical sets of $s$-derivatives. Thus, either $p \cup q \not\approx_k p$ or $p \cup q \not\approx_k q$ and, by Lemma 4.1, $p \not\approx_k q$ as desired.

Starting with $\approx_1$ and applying this reduction inductively $k-1$ times gives us PSPACE-hardness of $\approx_k$ in the restricted and observable model, for any fixed $k \geqslant 1$.

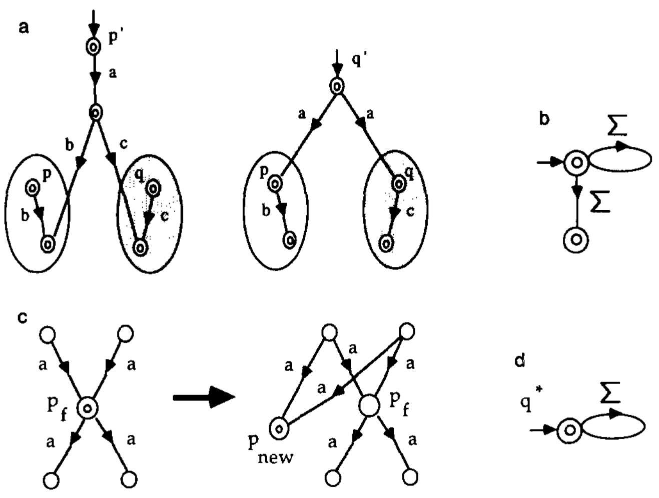


FIG. 5. (a) Reduction of Theorem 4.1(b). (b) The FSP *chaos*. (c) Reduction of Theorem 4.1(c). (d) The trivial NFA.

(c) Consider the r.o.u. model in which $\Sigma = \{a\}$. Here, it is easy to decide $p \approx_1 q$ or $L(p) = L(q)$, as the languages $L(p)$, $L(q)$ are closed under prefix and are therefore either $\{a\}^*$ or finite initial segments of $\{a\}^*$. However, if $p$ and $q$ are from the s.o.u. model, and thus not all states need be accepting, the problem of deciding $p \approx_1 q$ becomes co-NP-complete. This is an easy consequence of the co-NP-completeness of deciding $L(r) = \{a\}^*$, for star expression $r$ (Stockmeyer and Meyer, 1973). For $k \geqslant 2$, the technique of part (b) can be used to reduce the problem of r.o.u. $\approx_k$ to s.o.u. $\approx_1$. Thus it is in co-NP.

Let a *dead* state be a state devoid of outgoing transitions. To show the co-NP-hardness of deciding $\approx_k$, $k \geqslant 2$, in the r.o.u. model, we consider the problem $L(p) = \{a\}^+$, where $p$ is a state of an s.o.u. FSP having no dead states. The co-NP-hardness of this s.o.u. equivalence problem is, again, an easy consequence of the Stockmeyer and Meyer (1973) result. We now reduce this problem to $q \approx_2$ *chaos*, where $q$ is an r.o.u. state and *chaos* is the start state of the r.o.u. FSP given in Fig. 5b. Observe that, for any $q$, $q \approx_2$ *chaos* iff

(i) for every $s \in \{a\}^+, q$ has an $s$-derivative $q_1$ such that $L(q_1) = \varnothing$,

(ii) for every $s \in \{a\}^*, q$ has an $s$-derivative $q_2$ such that $L(q_2) = \{a\}^*$,

(iii) for every $s \in \{a\}^*$, these are the only types of $s$-derivatives of $q$.

We begin by transforming $p$ into $p'$, the start state of an s.o.u. FSP in which a state is an accept state iff it is a dead state, and $L(p) = L(p')$. We use the following procedure, illustrated in Fig. 5c. Let $p_f$ be an accept state that is not a dead state. Change $p_f$ to be a non-accept state and then create a new state $p_{\text{new}}$ which is both an accept and dead state, having as incoming transitions exactly those of $p_f$. Intuitively, a string that emanated from $p$ and was accepted at $p_f$ now emanates from $p'$ and is accepted at $p_{\text{new}}$.

Now we obtain $q$ from $p'$ by making every state of $p'$ accepting, and complete the proof of (c) by showing that $L(p) = \{a\}^+$ iff $q \approx_2$ *chaos*.

*only if*: Assume $L(p) = \{a\}^+$. Then in $q$ there is clearly a dead state $s$-derivative for every $s \in \{a\}^+$. Hence (i). Also, $q$ must lead to a cycle and hence (ii). Condition (iii) follows from the fact that the s.o.u. FSP of which $p$ was a state did not contain any dead states, and from the construction procedure for $p'$.

*if*: Assume $q \approx_2$ *chaos*. That $L(p) = \{a\}^+$ follows immediately from condition (i).

Now that we have established co-NP-hardness for deciding r.o.u. $\approx_2$, we can apply the PSPACE-hardness reduction of part (b) inductively $k-1$ times to obtain the co-NP-hardness of r.o.u. $\approx_k$, for any fixed $k \geqslant 2$. (Recall that the reduction of part (b) used only one symbol, $a$, and is thus still applicable in the r.o.u. case.) ▌

Finally, we would like to point out that in classical complextity theory (Stockmeyer and Meyer, 1973), there are results for the problem $p \approx_1 q^*$, where $q^*$ is the trivial NFA that accepts $\Sigma^*$ (see Fig. 5d). Using Definition 2.2.1, we can show that testing $p \approx_2 q^*$ in the general model is easy. Namely, $p \approx_2 q^*$ iff "every state reachable from $p$ has outgoing transitions for every symbol from $\Sigma$." This is a consequence of the fact that in going from $\approx_1$ to $\approx_2$, we examine all $s \in \Sigma^*$, whereas in going from $\simeq_1$ to $\simeq_2$, we examine only $s \in \Sigma \cup \{\varepsilon\}$.

## 5. The Complexity of Failure Equivalence

In this section, we analyze the complexity of testing two FSP states for failure equivalence. We will therefore be working in the restricted model. As described in Section 2, for restricted states $p$, $q$ we have

$$failures(p) = \{(s, Z) \mid s \in \Sigma^*, Z \subseteq \Sigma \text{ such that}$$
$$\exists p' \in K\colon p \overset{s}{\Longrightarrow} p' \text{ and } \forall z \in Z\colon \neg(p' \overset{z}{\Longrightarrow})\}$$

and

$$p \equiv q \qquad \text{iff} \quad failures(p) = failures(q).$$

Failure equivalence can be tested efficiently for the case of finite trees with $\tau$-transitions (Smolka, 1984). However, for slightly more complex finite processes, even without $\tau$-transitions, we have:

Theorem 5.1. *For FSP states $p$, $q$ in the restricted model, deciding $p \equiv q$ is in PSPACE and in co-NP if $|\Sigma| = 1$. Even for the restricted observable model with $|\Sigma| = 2$, deciding $p \equiv q$ is PSPACE-complete. For the r.o.u. model it is co-NP-complete.*

*Proof.* We exhibit a nondeterministic, polynomial space algorithm for deciding the failure equivalence of $p$ and $q$ in the restricted model. Since PSPACE = NPSPACE, this will give us membership in PSPACE. Assuming that $p \not\equiv q$, the algorithm guesses a failure pair $(s, Z) \in failures(p)$, and then verifies in polynomial space that $(s, Z) \notin failures(q)$ (or vice versa). In particular, let $s = a_1 \cdots a_k$. The algorithm guesses $s$ one symbol at a time. In response to the $i$th guess, i.e., the symbol $a_i$, the algorithm computes the

$a_1 \cdots a_i$-derivatives, i.e., the set of states reachable from $q$ by the observable string $a_1 \cdots a_i$, from the $a_1 \cdots a_{i-1}$-derivatives of $q$. Finally, each $s$-derivative of $q$ is checked separately for the existence of an $a$-derivative, for each $a \in Z$. Note that, because of nondeterminism, $s$ may be exponentially long in terms of the total number of transitions in $p$ and $q$.

To prove the PSPACE-hardness of deciding failure equivalence in the restricted observable model, we establish a reduction from the PSPACE-hard problem of restricted observable NFA equivalence (Lemma 4.2). Let $p$ and $q$ be the start states of two restricted observable FSPs. They can also be viewed as the start states of two NFAs which accept the languages $L(p)$ and $L(q)$, respectively. Note that, as a consequence of the restricted model, the only way a string $s$ is not in $L(p)$ or $L(q)$ is if $\neg(p \overset{s}{\Rightarrow})$ or $\neg(q \overset{s}{\Rightarrow})$.

Given $p, q$, we will produce two states $p', q'$ such that $L(p) = L(q)$ iff $p' \equiv q'$. Let $p$ be the start state of the restricted and observable FSP $\langle K, p, \Sigma, \Delta, \{x\}, E \rangle$, and obtain the restricted and observable FSP $\langle K', p', \Sigma, \Delta', \{x\}, E' \rangle$ as follows:

(i) $p' = p$.

(ii) $K' = K \cup \{p_{\text{dead}}\}$, where $p_{\text{dead}} \notin K$ is a new state, devoid of outgoing transitions.

(iii) $\Delta' = \Delta \cup \{\langle p_i, \sigma, p_{\text{dead}} \rangle \mid p_i \in K \text{ and } \sigma \in \Sigma\}$. Note that $p_{\text{dead}}$ has incoming transitions from all other states.

(iv) $E' = K' \times \{x\}$, i.e., all states are accepting.

Similarly, obtain $q'$ from $q$.

Obviously, $L(p') = L(p) \cup L(p) \cdot \Sigma$, $L(q') = L(q) \cup L(q) \cdot \Sigma$, and

$$\begin{aligned} \mathit{failures}(p') &= \{(s, \varnothing) \mid s \in L(p)\} \cup \{(s, Z) \mid s \in L(p) \cdot \Sigma \\ &\qquad \text{and } Z \subseteq \Sigma\}, \\ \mathit{failures}(q') &= \{(s, \varnothing) \mid s \in L(q)\} \cup \{(s, Z) \mid s \in L(q) \cdot \Sigma \\ &\qquad \text{and } Z \subseteq \Sigma\}. \end{aligned}$$

Hence, $p' \equiv q'$ iff

(a) $L(p) \cdot \Sigma = L(q) \cdot \Sigma$, and

(b) $L(p) \cup L(p) \cdot \Sigma = L(q) \cup L(q) \cdot \Sigma$.

It is easy to see that $L(p) = L(q)$ implies $p' \equiv q'$. If $p' \equiv q'$ then $L(p) \cdot \Sigma = L(q) \cdot \Sigma$ and, because in the restricted model $L(p)$ and $L(q)$ are prefix-closed, we have $L(p) = L(q)$.

This completes the reduction.

In the r.o.u. model, the problem of deciding $p \equiv q$ is easily shown to be in co-NP by reduction to the co-NP-complete problem of s.o.u. $\approx_1$ (Stockmeyer and Meyer, 1973). In particular, assuming $\Sigma = \{a\}$,

$$p \equiv q \quad \text{iff} \quad (L_0(p) = L_0(q) \text{ and } L_1(p) = L_1(q)),$$

where $L_0(p) = \{s \mid (s, \emptyset) \in \mathit{failures}(p)\}$ and $L_1(p) = \{s \mid (s, \{a\}) \in \mathit{failures}(p)\}$. The languages $L_0(q)$ and $L_1(q)$ are defined similarly.

To show the co-NP-hardness of deciding $p \equiv q$ in the r.o.u. model we resort, as in Theorem 4.1(c), to the problem of $L(p) = L(q)$ in the standard observable model with $\Sigma = \{a\}$. Let $p$ be a state of an s.o.u. FSP. We may assume that the set of accept states of this FSP is exactly the set of dead states. If not, apply the transformation described in the proof of Theorem 4.1(c) and depicted in Fig. 5c. The resulting FSP has the desired property and still has language $L(p)$.

Add to $p$ an $a$-transition to a newly introduced state which contains a single $a$-transition back to itself. If we now rename $p$ as $p'$ and view $p'$ as an r.o.u. state, then $\mathit{failures}(p') = \{(s, \emptyset) \mid s \in \{a\}^*\} \cup \{(s, \{a\}) \mid s \in L(p)\}$. The first set of the union is a consequence of the newly added state. The second set represents the fact that a failure of the form $(s, \emptyset)$ can arise only by following string $s$ to a dead state. Let $q$ also be a state of an s.o.u. FSP and obtain $q'$ from $q$ in the same way. The argument now that $L(p) = L(q)$ iff $p' \equiv q'$ is immediate. ∎

The PSPACE-completeness of failure equivalence for restricted processes was shown independently by Brookes and Rounds (1983).

## 6. Discussion and Open Problems

We have investigated the complexity of three equivalence notions that are central to the definition of CCS semantics. We have tried to draw a close analogy between finite state processes and expressions in CCS on one hand, and finite state automata and regular expressions on the other. We believe that in CCS, an algebraic model for distributed computation, many of the classical problems are cast in a new light. An example is the star height question about star expressions raised in Milner (1984). We would like to point out an open problem which we believe is both interesting and important.

*CCS Equivalence.* For the star expressions defined in Section 2.3, the CCS equivalence problem is essentially that of testing finite state processes, of size comparable to that of the expressions, for strong equivalence of start states. CCS, being a calculus, provides a number of other algebraic operators besides $\cup$, $\cdot$, $*$. Therefore, as we have extended regular expressions in the classical theory (Stockmeyer and Meyer, 1973), we have extended star expressions in CCS. Since the semantics of CCS is in terms of sets of processes rather than strings, an operator such as complement ($\neg$) would make little sense in the new context. However, operators such as composition (Milner, 1980) or intersection can be given new semantics. (Composition is one of the main distributed features of CCS.) The new semantics is, predictably, in terms of a "direct product of states" construction. In the spirit of Definition 2.3.1, the representative process of the whole is the result of taking the direct product of the representative processes of the parts. With extended star expressions, the CCS equivalence problem acquires new interest. Extended star expressions are succinct programs with large representative finite state processes, because of possible nesting of the new operators. So perhaps the CCS equivalence problem becomes hard, as do its counterparts in Stockmeyer and Meyer (1973).

We would like to end this discussion on a note of optimism. For regular expressions, the MEMBER problem—i.e., Is string $s$ in the language denoted by regular expression $r$?—is different from the EQUIVALENCE problem and solvable efficiently by dynamic programming (Hopcroft and Ullman, 1979, Sect. 4.5). For extended star expressions, the distinction between the membership problem for CCS—i.e., Is state $P$ in the equivalence class denoted by CCS expression $r$?—and the equivalence problem for CCS is much weaker. Therefore, it is possible that the CCS equivalence problem for extended star expressions has an elegant and efficient solution.

## APPENDIX A: FSP TYPES AND EQUIVALENCES

TABLE I

FSP Types

| FSP type | Definition |
|---|---|
| *General* | The most general type of FSP as given by Definition 2.1.1. |
| *Observable* | General FSPs without $\tau$-transitions. |
| *Standard* | General FSPs in which each state is either an accept state or non-accept state. |
| *Deterministic* | Observable FSPs where for each state there is exactly one transition for each symbol in $\Sigma$. |
| *Restricted* | Standard FSPs in which every state is an accept state. |
| *Restricted observable* | FSPs that are both restricted and observable. |
| *Restricted observable unary* (r.o.u.) | FSPs that are both restricted and observable, with $\|\Sigma\| = 1$. |
| *Standard observable* | FSPs that are both standard and observable. |
| *Standard observable unary* (s.o.u.) | FSPs that are both standard and observable, with $\|\Sigma\| = 1$. |
| *Finite tree* | Restricted FSPs whose underlying directed graph is a tree. |

TABLE II

Equivalences

| Equivalence relation symbol | Name and definition |
|---|---|
| $\approx_k$ | $k$-observational equivalence (Definition 2.2.1) |
| $\approx$ | Observational equivalence (Definition 2.2.1) |
| $\simeq_k$ | $k$-limited observational equivalence (Definition 2.2.2) |
| $\simeq$ | Limited observational equivalence (Definition 2.2.2) |
| $\sim$ | Strong (observational) equivalence (Definition 2.2.3) |
| $\equiv$ | Failure equivalence (Definition 2.2.4) |

## APPENDIX B: Proof of Proposition 2.2.1

PROPOSITION 2.2.1. *For FSP states $p, q$ in the* general *model,*

(a) $\simeq$ *is a* $(\Sigma \cup \{\varepsilon\})$*-fixed-point.*

(b) $\simeq$ *is a* $\Sigma^*$*-fixed-point.*

(c) $p \simeq q$ *iff* $p \approx q$.

*Proof.* (a) We have to show that $p \simeq q$ iff conditions (1) and (2) for a $\Sigma \cup \{\varepsilon\}$-fixed-point hold (see Definition 2.2.5).

*only if*: Let $p \simeq q$. Then by Definition 2.2.2, $E(p) = E(q)$ and, for each $k \geqslant 0$, $p \simeq_{k+1} q$. Thus for each $k \geqslant 0$ and each $\sigma$ in $\Sigma \cup \{\varepsilon\}$ we have

$$\text{if } p \overset{\sigma}{\Longrightarrow} p' \text{ then } (\exists q_k\colon q \overset{\sigma}{\Longrightarrow} q_k \text{ and } p' \simeq_k q_k).$$

Since our processes are finite state, the set $\{q_k \mid q \Rightarrow^\sigma q_k \text{ and } p' \simeq_k q_k\}$ is finite. By the pigeonhole principle there is a $q'$ such that ($q \Rightarrow^\sigma q'$ and $p' \simeq_k q'$) for infinitely many $k$. Since $p' \simeq_{i+1} q'$ implies $p' \simeq_i q'$ we must have $p' \simeq q'$. Therefore for each $\sigma$ in $\Sigma \cup \{\varepsilon\}$ we have

$$\text{if } p \overset{\sigma}{\Longrightarrow} p' \text{ then } (\exists q'\colon q \overset{\sigma}{\Longrightarrow} q' \text{ and } p' \simeq q').$$

A symmetric argument completes the only if direction.

*if*: If conditions (1) and (2) hold for states $p$ and $q$ it is easy to see that $p \simeq_k q$ for all $k$ and thus $p \simeq q$.

(b) We have to show that $p \simeq q$ iff conditions (1) and (2) for a $\Sigma^*$-fixed-point hold.

*if*: Since $\Sigma \cup \{\varepsilon\} \subseteq \Sigma^*$ this direction is immediate from the if direction of part (a).

*only if*: We proceed by induction on the length of $s$ in $\Sigma^*$. For $s = \varepsilon$ the result follows immediately from the only if direction of part (a).

Consider next $s=\sigma_1 \cdots \sigma_n$, $n \geqslant 1$, $\sigma_i$ in $\Sigma$. Assume $p \Rightarrow^{\sigma_1} p_1 \Rightarrow^{\sigma_2} \cdots \Rightarrow^{\sigma_n} p_n$. Then by part (a) used repeatedly $\exists q_1, ..., q_n$: $q \Rightarrow^{\sigma_1} q_1 \Rightarrow^{\sigma_2} \cdots \Rightarrow^{\sigma_n} q_n$ with $p_i \simeq q_i$, $1 \leqslant i \leqslant n$, and thus

$$\text{if } p \overset{s}{\Longrightarrow} p_n \text{ then } (\exists q_n\colon q \overset{s}{\Longrightarrow} q_n \text{ and } p_n \simeq q_n).$$

A symmetric argument completes the only if direction.

(c) *only if*: We show by induction that $p \simeq q$ implies $p \approx_k q$ for all $k \geqslant 0$. At $k=0$ it is trivial. Assume it for $k$ (for all $p$ and $q$); we will prove it for $k+1$. From part (b), $p \simeq q$ implies

(1) $E(p)=E(q)$.

(2) For every $s \in \Sigma^*$,

(i) if $p \Rightarrow^{s} p_1$ then $(\exists q_1: q \Rightarrow^{s} q_1$ and $p_1 \simeq q_1)$,

(ii) if $q \Rightarrow^{s} q_2$ then $(\exists p_2: p \Rightarrow^{s} p_2$ and $q_2 \simeq p_2)$.

Using the inductive hypothesis, we can replace $p_1 \simeq q_1$ with $p_1 \approx_k q_1$ and $p_2 \simeq q_2$ with $p_2 \approx_k q_2$, and by Definition 2.2.1 we have that $p \simeq q$ implies $p \approx_{k+1} q$.

*if*: We show by induction that $p \approx_k q$ implies $p \simeq_k q$ for all $k$. At $k=0$ it is trivial. The inductive step is an easy consequence of Definitions 2.2.1 and 2.2.2. ▮

## Acknowledgments

We thank the anonymous referees for their critical reading of the manuscript which produced numerous constructive comments. We are also grateful to Tommaso Bolognesi, Ashok Chandra, Gordon Plotkin, and Larry Stockmeyer for many helpful discussions, and to Ashfaq Munshi for his comments on the presentation of these results.

Received October 1, 1986; accepted January 24, 1989

## References

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1974), "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, MA.

Benson, D. B., and Ben-Shachar, O. (1988), Bisimulation of automata, *Inform. and Comput.* **79**, No. 1, 60–83.

Bergstra, J. A., and Klop, J. W. (1986), Algebra of communicating processes, *in* "CWI Monographs I, Proceedings of the CWI Symposium on Mathematics and Computer Science" (J. W. de Bakker, M. Hazewinkel, and J. K. Lenstra, Eds.), pp. 89–138, North-Holland, Amsterdam.

Bergstra, J. A., and Klop, J. W. (1987), "ACP$_\tau$: A Universal Axiom System for Process Specification," Report CS-R8725, Computer Science/Department of Software Technology, CWI, Amsterdam, The Netherlands.

Brookes, S. D. (1983), On the relationship of CCS and CSP, *in* "Proceedings, 10th International Conference on Automata, Languages, and Programming, Barcelona, Spain," pp 83–96, Lecture Notes in Computer Science, Vol. 154, Springer-Verlag, New York/Berlin.

Brookes, S. D., Hoare, C. A. R., and Roscoe, A. W. (1984), A theory of communicating sequential processes, *J. Assoc. Comput. Mach.* **31**, No. 3, 560–599.

Brookes, S. D., and Rounds, W. C. (1983), Behavioural equivalence relationships induced by programming logics, *in* "Proceedings, 10th International Conference on Automata, Languages, and Programming, Barcelona, Spain," pp. 97–108, Lecture Notes in Computer Science, Vol. 154, Springer-Verlag, New York/Berlin.

Chandra, A. K., and Stockmeyer, L. J. (1982), Private communication.

Coppersmith, D., and Winograd, S. (1987), Matrix multiplication via arithmetic progressions, *in* "Proceedings, 19th ACM Symposium on Theory of Computing, New York," pp. 1–6.

Hennessy, M. C. B. (1985), Acceptance trees, *J. Assoc. Comput. Mach.* **34**, No. 4, 896–928.

Hennessy, M. C. B., and Milner, R. (1985), Algebraic laws for nondeterminism and concurrency, *J. Assoc. Comput. Mach.* **32**, No. 1, 137–161.

Hopcroft, J. E. (1971), An $n \log n$ algorithm for minimizing states in a finite automaton, *in* "Theory of Machines and Computations" (Z. Kohavi and A. Paz, Eds.), pp. 189–196, Academic Press, San Diego, CA/New York.

Hopcroft, J. E., and Ullman, J. D. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, MA.

Kanellakis, P. C., and Smolka, S. A. (1983), CCS expressions, finite state processes, and three problems of equivalence, *in* "Proceedings, 2nd Annual ACM Symposium on Principles of Distributed Computing, Montreal, Canada," pp. 228–240.

Kanellakis, P. C., and Smolka, S. A. (1988), On the analysis of cooperation and antagonism in networks of communicating processes, *Algorithmica* **3**, 421–450.

Milner, R. (1980), A calculus of communicating systems, *in* "Lecture Notes in Computer Science, Vol. 92," Springer-Verlag, New York, Berlin.

Milner, R. (1983), Calculi for synchrony and asynchrony, *Theoret. Comput. Sci.* **25**, 267–310.

Milner, R. (1984), A complete inference system for a class of regular behaviors, *J. Comput. System Sci.* **28**, 439–466.

de Nicola, R., and Hennessy, M. C. B. (1984), Testing equivalences for processes, *Theoret. Comput. Sci.* **34**, No. 1, 83–133.

Paige, R., and Tarjan, R. E. (1987), Three partition refinement algorithms, *SIAM J. Comput.* **16**, No. 6, 973–989.

Salomaa, A. (1986), Two complete axiom systems for the algebra of regular events, *J. Assoc. Comput. Mach.* **13**, No. 1, 158–169.

Sanderson, M. T. (1982), "Proof Techniques for CCS," Internal Report CST-19-82, Department of Computer Science, University of Edinburgh.

Smolka, S. A. (1984), "Analysis of Communicating Finite-State Processes," Ph. D. dissertation, Technical Report CS-84-05, Department of Computer Science, Brown University, Providence, RI.

Stockmeyer, L. J., and Meyer, A. R. (1973), Word problems requiring exponential time, *in* "Proceedings, 5th ACM Symposium on Theory of Computing, Austin, Texas," pp. 1–9.