Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Electronic Notes in Theoretical Computer Science 128 (2005) 21–36
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Regularity Results for FIFO Channels[1]

Nils Klarlund

Bell Labs, Lucent Technologies
Murray Hill, New Jersey

Richard Trefler

School of Computer Science
University of Waterloo, Waterloo, Ontario

Abstract

FIFO channel systems, in which messages between processes are cached in queues, are fundamental to the modeling of concurrency. A great deal of effort has gone into identifying scenarios where reasoning about such systems is decidable, often through establishing that the language of all channel contents is regular. Most prior results in this area focus on the effect of repetitions of individual operations sequences or they constrain the channels either to be lossy or to be polynomially bounded (that is, the number of words of a given length describing channel contents is bounded by a polynomial).
We focus on *piecewise* languages for both describing operations and channel contents. Piecewise languages restrict the Kleene star operation to be applied to sets of letters only. For example, $a(b+c)^*$ is piecewise (but not polynomially bounded). These languages correspond to the Σ_2 class of the first-order quantifier hierarchy. It is already known that piecewiseness plays a key role in establishing regularity results about parameterized systems subjected to rewritings according to semi-commutation rules.
In this paper, we show that piecewiseness is central to the understanding of FIFO channel systems. Our contribution is to study the effect of iterating sets of operations, while extending and unifying previous work on both lossy and perfect FIFO systems. In particular, we show that well-quasi-orderings are important to Σ_2, not only to the lossy systems of Π_1. Moreover, we show that Σ_2 also describes limits in a class of FIFO systems that include iterations of arbitrary sets of simultaneous read and write operations.

Keywords: Regularity, FIFO channels, queues, concurrency.

[1] The authors are supported in part by grant CCR-0341658 from the National Science Foundation and by grants from NSERC, CITO, and Nortel.
[2] Email: klarlund@research.bell-labs.com
[3] Email: trefler@cs.uwaterloo.ca

doi:10.1016/j.entcs.2005.04.002

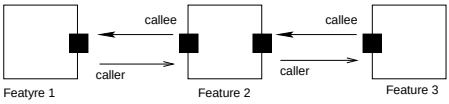

Fig. 1. ECLIPSE call structure

1 Introduction

We show that the class of *piecewise languages* are important to the understanding of finite-state systems that communicate over unbounded channels. Piecewise languages are regular languages that are finite unions of *simply piecewise* languages of the form $M_1^* a_1 \cdots M_n^* a_n M_{n+1}^*$, where the M_i's are subsets of a finite alphabet Σ of symbols and the a_i's are elements of Σ. To express the language consisting of ϵ (the empty word), we allow $n = 0$. Note that if $M_i = \{\}$, then M_i^* is ϵ; also, it can be seen that replacing the a_i's by finite strings of u_i's does not affect the class of languages defined.

Piecewise languages can also be characterized as Σ_2-languages of the quantifier alternation hierarchy for first-order logic on words over the signature $(<, a(\cdot)_{a\in\Sigma})$ or as the level $1\frac{1}{2}$ of the concatenation hierarchy of Straubing-Thérien, see [19]. More simply, piecewise languages are those recognized by nondeterministic automata whose only nontrivial strongly connected components are states with self-loops.

1.1 Motivating example

Our investigations have a practical background. ECLIPSE (now called BoxOS) is a next-generation telephony service over IP infrastructure developed at AT&T Labs; see [8] for our earlier work on model-checking ECLIPSE. Telephone calls are structured as in Figure 1. Boxes at the end points represent telephones, intermediate boxes represent call features, for example call-forwarding-on-busy, and the arrows represent, possibly unbounded, perfect communication channels, or queues, that pass messages from endpoint to endpoint. However, at a sufficiently high level of granularity, each box represents a finite state transducer.

Our focus is on the problems inherent in checking properties of systems composed of several boxes. Consider the *transparent box* described on the left of Figure 2. This transparent box represents a communication template that all system boxes must implement. Figure 2 has been simplified for this presentation in several important ways: the feature box in the picture communicates with only one neighbor, in general, communication may be n-way, and is usually two-way. Simple replication of the one-way communication functionality to create a two-way machine results in a feature with 17 states — the neighbors are not symmetric, one will be the initiating, upstream feature, the other the receiving, downstream feature.

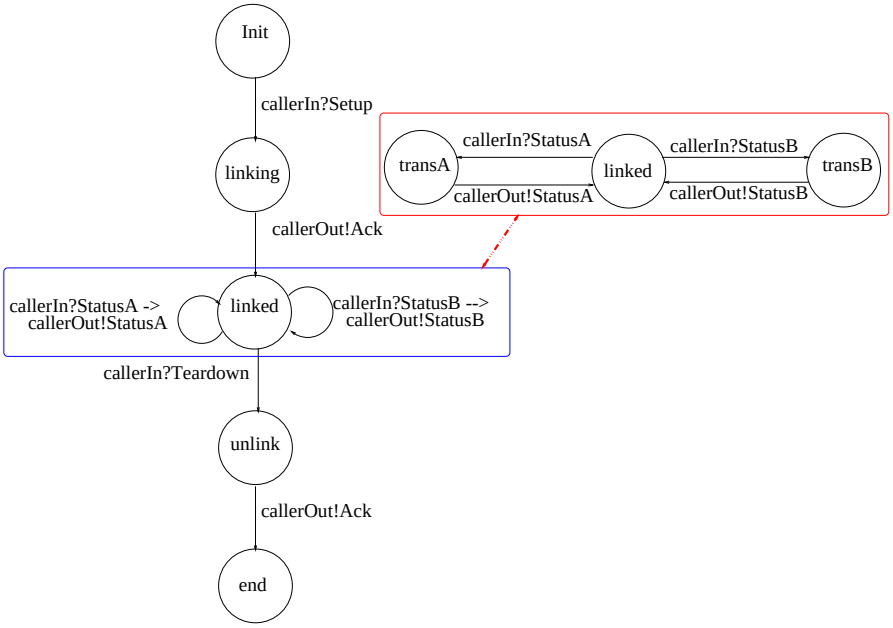

Fig. 2. Abstract Transparent Feature Box

The transparent box communicates with one neighbor across two separate channels. Messages from the neighbor, in this case an initiating, upstream, environment (*cf.*[8]), representing the *caller*, are received via the `callerIn` channel. Messages sent back to the caller are sent on the `callerOut` channel. A message is received with the '?' symbol and sent with the '!' symbol. So `callerIn?Setup` indicates a call setup message received on the `callerIn` channel.

At present, we are not concerned with the full details, but we note that the structure of the transparent box in Figure 2 is piecewise. To achieve piecewiseness we have abstracted the transparent box by replacing the original linked state and its left and right neighbors, given in the figure on the right by the linked state given in the full box on the left. The state on the left has the same functionality as the figure on the right. The difference is the addition of *conditional operations* of the form `callerIn?Status` → `callerOut!Status` where the Status message is sent to the caller only if the Status message has been received from the caller first.

The global state of a system is describable by the states of the automata corresponding to the boxes in addition to *joint content*, which is a word listing the contents of each channel in some predetermined order. In other words, we operate with *recognizable* relations describing the channels (see Proposition 2.7). Although Figure 2 presents a single feature as a piecewise automaton, it is possible to compose piecewise automata, representing individual

processes, and obtain a piecewise product automaton, whose states are called *control states*. A pair consisting of a control state and joint content is a *global state*. The set of possible joint contents at a control state is the *channel language* of the state.

For each control state, we are interested in describing the *limit set* of all join contents that may arise from a set of initial global states. The ability to calculate the limit set is important to the automatic verification of properties of the system. Generally, of course, this is not possible; in fact the limit set is usually not a regular language.

1.2 Our contributions

Our main insight is that limit languages of systems described with piecewise operations languages remain regular even if conditional operations are added to a set of transitions iterated on FIFO systems. In particular, we show that for single channel systems, the action of piecewise languages of operations preserves regularity (and piecewiseness) of initial channel languages (Proposition 3.2).

For multiple channel systems, we show that the limit language is piecewise if the initial language is piecewise. Our construction is not effective, but we provide an algorithm for calculating the limit language if no message may circulate indefinitely. A technical tool we introduce is that of the *repetition piecewise* languages, that are a subclass of Π_1; repetition piecewise languages enjoy attractive closure properties, for example, closure under arbitrary unions.

The paper is organized as follows. We discuss related work below. In Section 2, we provide an overview of the mathematics of regular languages as applied to the description of perfect and lossy FIFO systems; we include several known results that bear on piecewise languages along with new contributions. In Section 3, we introduce operation languages and their actions on channel languages. In addition to Proposition 3.2, we consider multiple-channel systems with and without conditional operations. Finally, we provide a summary of our results in Section 4 along with a statement of problems still to be solved.

1.3 Previous work

FIFO systems with finite control and unbounded channels have been studied extensively, so our survey will not be comprehensive. Despite the undecidability of interesting questions about even one-channel systems [10], tractable scenarios have been investigated since Pachl's work [18,17]. Among other precursors to modern techniques, Pachl introduces assertions for proving protocol properties. These assertions are in the form of recognizable or rational descriptions of the joint content of the channels. (Rational descriptions are

more general than recognizable ones, but have apparently not been considered since Pachl's work.) In the framework of [10,17], a limit language, which may not be regular, is naturally associated to each control state. But Pachl notes that if recognizable, but not rational, assertions are associated to control states, it is decidable whether the assertions hold across transitions (and whether the assertion associated with the initial control state holds). Consequently, Pachl shows that if the limit language L is known to be regular, then the *reachability question*, whether $w \in L$ for joint content w, is decidable (it suffices to interleave the two algorithms: one that enumerates words in the limit language and one that verifies enumerated assertions and tests whether w is outside the enumerated assertions). Of course, this algorithm is unusable in practice, and it does not explain when limit languages are regular. Also, a reduction from the Post Correspondence Problem shows that even when the limit language is known to be regular, determining it may still be undecidable [14].

An appealing general model to distributed systems with queues is that of *FIFO nets*, which are formulated as Petri nets except that places are replaced by FIFO channels. The survey [16] contains several decidability results, but they depend on the channel languages being *bounded* (that is, a subset of some language $w_0^* \cdots w_{n-1}^*$, where the w_i's are words).

For *lossy FIFO systems*, where it is assumed that any message may be lost, otherwise undecidable problems such as reachability become decidable thanks to well-quasi-orderings on channel contents [4,14].

The work mentioned so far does generally speaking not offer methods for calculating the limit languages; instead, the central theme is on deciding properties such as reachability and deadlock freedom. Also, safety properties about the control states (which do not include the joint content) may easily be reduced to reachability; see [4].

Boigelot *et al.* studied the problem of calculating limit languages. For f a receive, send, or simultaneous read and write operation, [6] proposes detailed automata-theoretic algorithms that calculate the effect $f : L$ of the operation when applied to channel language L as well of the effect $f^* : L$ of the iterated operation. Using derivatives [12], we reformulate these results in elementary terms. Also, [6] proposes a heuristic for calculating limit sets: if there is a transition from control state p to control state q on an operation f, then the channel language $L(q)$ of q is updated to be $L(q) := L(q) \cup f : L(p)$ and for *loops*, that is, transitions from p to p, the update is *accelerated* as $L(p) := f^* : L(p)$; in this way, an arbitrary number of loop traversals is considered at once. Moreover, it is proposed that the accelerated updates are prioritized over transitions among different control states. The resulting algorithm is called *loop-first search*. Our Proposition 3.3 shows that our results on limit languages for single-channel systems permit a stronger acceleration algorithm when multiple loops are considered simultaneously. Another heuristic of [6]

is to combine read and write operations that would otherwise constitute a two-cycle. We call these operations *conditional* and our present study focuses on iterations of sets that include such operations.

While [6] only considers the effect of individual actions, [5] presents a detailed treatment of the iteration of sets of operation sequences. In particular, various conditions are provided on such sets that characterize when their iterations preserve recognizability. However, the conditions are applicable only to sets that satisfy a condition of *read synchronization*: if $t?$ and $t'?$ are sequences in the set, then the subsequence of read operations in t and that of t' are the same. We do not impose such restrictions, but we consider only sets of operation sequences that consist of a single operation or of a combined read and write operation.

In [7], the operation sequences that preserve regularity of channel languages is characterized; they are called *non-counting* and the set of such sequences is shown to be recognizable.

A special kind of regular expression, called *semilinear*, was introduced in [15] as a symbolic representation of regular, bounded languages describing channel contents. Unfortunately, a bounded language L has polynomial density: there are at most $P(n)$ words of size n for some polynomial P. This may be a severe restriction; for example, it precludes sending a's and b's that are arbitrarily interspersed.

Again based on well-quasi-orderings, [2] identifies languages at the Π_1 level of the first-order hierarchy as fundamental to representing the content of lossy channels in FIFO systems. The *simple regular expressions (SRE's)* introduced there encode Π_1 sets; they look superficially similar to piecewise expressions (the difference is that a_i is substituted for $a_i?$ in SRE's); that is, these languages are the downward closed languages under the scattered subword ordering. This class is properly contained in Σ_2 since the first-order hierarchy is strict.

The importance of piecewise languages to parameterized networks of finite-state processes is established in [9], where it is shown that applying *semicommutation* rewriting rules of the form $ab \to ba$ to piecewise languages yields limit languages that are also piecewise. This result also follows from the observation that semicommutation preserves the anchor length of piecewise expressions (defined in Section 2) combined with our Proposition 2.9 that any anchor-length-bounded union of piecewise languages is piecewise. Importantly, in [9], an effective way of calculating the limit language under semicommutative rewriting is given. In our case, we do not know if the limit of piecewise operation languages acting on multiple FIFO channel systems is effectively piecewise, but we do know that this is the case for a single-channel system (Proposition 3.2). Based on [20], representations for piecewise languages and complexity results are also discussed in [9]. A general approach to the Straubing-Thérien Hierarchy, including characterizations of piecewise

languages, is provided in [20,19].

Piecewise languages include the *piecewise testable* languages [21] (a piecewise testable language is a piecewise language L for which the sets M_i in the expression defining it are all equal to Σ). So there is a historic reason for naming the Σ_2 class "piecewise languages". Previously, the name "Alphabetic Pattern Constraints" has been suggested [9].

2 Preliminaries and notation

2.1 Regular and piecewise languages

A regular expression R over Σ is of the form a, where $a \in \Sigma$, $R' \cdot R''$, $R' + R''$, R'^*, $\mathbf{0}$, or $\mathbf{1}$. The symbol $\mathbf{0}$ denotes the empty language, and $\mathbf{1}$ denotes the language $\{\epsilon\}$; in particular, we have $\mathbf{1} = \mathbf{0}^*$. The language $\mathcal{L}(R)$ denoted by a regular expression is defined in the usual way. We sometimes just write R when we mean $\mathcal{L}(R)$. In particular, we have $R \cdot \mathbf{1} = R$ and $R \cdot \mathbf{0} = \mathbf{0}$. In a further abuse of notation, we sometimes regard a set $M \subseteq \Sigma \cup \{\epsilon\}$ as a regular expression, namely the sum of elements in M. The expression $\mathrm{test}(R)$ is $\mathbf{1}$ if $\mathcal{L}(R) \neq \emptyset$ and $\mathbf{0}$ if $\mathcal{L}(R) = \emptyset$. For notational convenience inside tests, we use the operator $\cap$ to denote the intersection operator of extended regular expressions.

An automaton A over an alphabet Σ is of the form $(\Sigma, Q, \Delta, q^0, Q^F)$. The automaton is *deterministic* if for all q and a there is exactly one q' such that $(q, a, q') \in \Delta$. A *partially ordered automaton* $(A, \preceq)$ is an automaton together with a partial ordering $\preceq$ on its states such that for any $(q, a, q') \in \Delta$ it holds that $q \preceq q'$.

A language is *simply piecewise* if it can be expressed by a regular expression of the form $M_1^* a_1 \cdots M_k^* a_k M_{k+1}^*$, where $M_i \subseteq \Sigma$ and $a_i \in \Sigma \cup \{\epsilon\}$. A language L is *piecewise* if it is a union of simply piecewise languages. L is *repetition piecewise* if all a_i's are ϵ. If L is a union of sets of the form $M_1^* a_1 \cdots M_k^* a_k M_{k+1}^*$ such that for all i it holds that $a_i \neq \epsilon$ and $a_i \notin M_i$, then L is *deterministic piecewise* . In particular, $\mathbf{0}$ $(= \overline{\Sigma^*})$ and $\mathbf{1}$ $(= \emptyset^*)$ are deterministic piecewise.

We summarize properties that are well-known (although we believe (e) has not been reported before):

Proposition 2.1 *(a) $(a+b)^*ab$ is piecewise, but not deterministic piecewise. (b) Every deterministic piecewise language is piecewise. (c) Piecewise languages are star-free; more precisely, they are models of Σ_2-sentences (first-order logic formulas in prenex form with quantifier prefix $\exists \cdots \exists \forall \cdots \forall$) over the signature $(<, a(\cdot)_{a \in \Sigma})$; they are closed under finite unions, finite intersections, concatenation, shuffle, and projections (defined by letter-to-letter mappings) and inverse homomorphisms, but not under complementation and substitutions. (d) Deterministic piecewise languages are closed under boolean operations and inverse homomorphisms, but not under concatenation, shuffle,*

or projections. (e) Repetition piecewise languages are deterministic piecewise and closed under finite unions and intersections, concatenation, shuffle, and projections, but not under substitutions or inverse homomorphisms.

Let $\leq$ be the partial order on words defined by $x \leq y$ if x is a scattered subword of y, that is, by deleting letters in y one may obtain x. For a set of words A define $A^{\leq}$, the *upward closure of* A, to be the set of words y for which $x \leq y$ for some $x \in A$. Define $A^{\geq}$, the *downward closure* of A, to consist of words y for $x \geq y$ and $x \in A$. The partial ordering $\leq$ is a well-quasi-ordering according to Higman's Lemma [1] (a *well-quasi-ordering* is a transitive, reflexive relation such that any upwards closed subset has finitely many minimal elements). Note that a repetition piecewise language L is downwards closed, that is $L = L^{\geq}$. For $L \subseteq \Sigma^*$, define $\complement L$ to be $\Sigma^* \backslash L$. The following is well-known:

Proposition 2.2

(a) A repetition piecewise language L is of the form $L = \complement A^{\leq}$ for some finite A.

(b) Any downward closed L is piecewise (in fact Π_1, see [19], or SRE[3]).

Proof.

(a) L is downward closed. So, $\complement L$ is upward closed, that is, $(\complement L)^{\leq} = \complement L$. By Higman's Lemma, there is a finite A such that $A^{\leq} = \complement L$.

(b) (Hint) If L is downward closed, then $\complement L$ is upward closed and hence regular by Higman's Lemma. So, $L = L^{\geq}$ is regular. Now a simple inductive transformation of a regular expression R with $\mathcal{L}(R) = L$ yields an equivalent piecewise expression.

□

We have not found the following property in the literature:

Proposition 2.3 *The following are alternative characterizations of repetition piecewise languages L.*

(a) L is recognized by a deterministic, partially ordered, minimal automaton such that at most one state is rejecting (and it is a sink state) and for every transition (s, a, s') there is also a transition (s', a, s').

(b) The canonical congruence $\equiv_L$ (defined by $x \equiv y$ if and only if for all u, v it holds that $uxv \in L \Leftrightarrow uyv \in L$) is of finite index, $\forall a \in \Sigma : a \equiv_L a^2$, and $\forall u : u \cdot v \in L \Rightarrow u \in L$.

Proposition 2.4 *[13] The following are alternative characterizations of deterministic piecewise languages L.*

(a) L is recognized by a deterministic, partially ordered automaton.

(b) *The canonical congruence* $\equiv_L$ *is of finite index and there is an* $n > 0$ *such that* $\forall x, y : (x \cdot y)^n \equiv (x \cdot y)^n \cdot x$.

Thus, it is also straightforward to determine whether a given language is deterministic piecewise.

Proposition 2.5 *The following are alternative characterizations of piecewise languages* L.

(a) L *is recognized by a (possibly nondeterministic) partially ordered automaton.*

(b) *[20] The canonical congruence* $\equiv_L$ *is of finite index and there is an* $n > 0$ *such that* $\forall x, y, u, v : (c(x) = c(y) \ \wedge \ u \cdot x^n \cdot v \in L) \Rightarrow u \cdot (x^n \cdot y \cdot x^n) \cdot v \in L$, *where* $c(x)$ *is the set of letters occurring in* x.

Proposition 2.5 can be used to decide whether a given regular language is piecewise; an efficient algorithm is provided in [9].

Proposition 2.6 *The union* L *of any family of repetition piecewise languages is repetition piecewise.*

Proof. (Hint) By Proposition 2.2(b), L is regular. Use Proposition 2.3(b). □

Recall (from say[22]) that a relation $\rho \subseteq (\Sigma^*)^K$ is *recognizable* if and only if

$$\rho = \bigcup_{0 \le i < I} \mathcal{L}(R_0^i) \times \cdots \times \mathcal{L}(R_{K-1}^i) \tag{1}$$

for some number I and some regular expressions R_j^i over Σ. The following is known:

Proposition 2.7 *Let* ρ *be a* K*-ary relation over* Σ^*. *Define* $\mathcal{L}(\rho) = \{w_0 \cdot \# \cdots \# \cdot w_{K-1} \mid (w_0, \ldots, w_{K-1}) \in \rho\}$. *Then* $\mathcal{L}(\rho)$ *is a regular language over* $\Sigma \cup \{\#\}$ *if and if only if* ρ *is recognizable.*

Moreover, $\mathcal{L}(\rho)$ is piecewise if and only if the $R_{i,j}$ can be chosen to be piecewise. In that case, we say that a recognizable relation is *repetition piecewise.*

Proposition 2.8 *The union* L *of any family of repetition piecewise relations is repetition piecewise.*

Proof. The proof is similar to that of Proposition 2.6. We use the fact that a product of well-quasi-orderings is well-quasi-ordered. □

The *anchor sequences* of a piecewise expression

$$R = \sum_{i<I} M_1^{i*} a_1^i \cdots M_k^* a_{k(i)}^i M_{k(i+1)}^{i*}$$

is the set $\{a_1^i \cdots a_{k(i)}^i | i < I\}$. The *anchor length* of R is $\max_{i<I} k(i)$ and the anchor length of piecewise L is the minimum anchor length of an R such that $\mathcal{L}(R) = L$.

Proposition 2.9 *The union L of any family of piecewise languages of bounded anchor length is piecewise.*

Proof. (Idea) By a rearrangement of the simply piecewise expressions of the languages of the family, it suffices to consider a finite union of languages L_ℓ, where L_ℓ is a union of simply piecewise languages all having the same anchor sequence. We use here the facts that there are only finitely many such anchor sequences and that piecewise languages are closed under finite union. Thus, consider $L_\ell = \bigcup_{i \geq 0} \mathcal{L}(R^i)$, where

$$R^i = M_1^{i*} a_1^i \cdots M_k^* a_{k(\ell)}^i M_{k(\ell+1)}^{i*}$$

Then use Proposition 2.8. □

We now recall that the *left residual operation* (or derivative [11]) is defined as

$$\begin{aligned}
a^{-1}\mathbf{0} &= \mathbf{0} \\
a^{-1}\mathbf{1} &= \mathbf{0} \\
a^{-1}b &= \mathrm{test}(a \cap b) \\
a^{-1}(R \cdot S) &= a^{-1}R \cdot S + \ \mathrm{test}(R \cap \mathbf{1}) \cdot a^{-1}S \\
a^{-1}(R + S) &= a^{-1}R \ + \ a^{-1}S \\
a^{-1}(R^*) &= a^{-1}R \cdot R^*
\end{aligned}$$

and that $\mathcal{L}(a^{-1}R) = \{v \mid a \cdot v \in \mathcal{L}(R)\}$. Similarly, we may define a residual operation for M^*, where $M \subseteq \Sigma$:

$$\begin{aligned}
(M^*)^{-1}\mathbf{0} &= \mathbf{0} \\
(M^*)^{-1}\mathbf{1} &= \mathbf{1} \\
(M^*)^{-1}a &= a \ + \ \mathrm{test}(a \cap M) \\
(M^*)^{-1}(R \cdot S) &= (M^*)^{-1}R \cdot S \ + \ \mathrm{test}(R \cap M^*) \cdot (M^*)^{-1}S \\
(M^*)^{-1}(R + S) &= (M^*)^{-1}R \ + \ (M^*)^{-1}S \\
(M^*)^{-1}(R^*) &= (M^*)^{-1}R \cdot R^* + \mathbf{1}
\end{aligned}$$

Then, it can be verified that

$$\mathcal{L}((M^*)^{-1}R) = \{v \mid \exists u \in \mathcal{L}(M^*) : u \cdot v \in \mathcal{L}(R)\}$$

2.2 Channels and transformations

A *channel over* Σ is a FIFO queue whose contents is described by a word $w \in \Sigma^*$. A *channel operation* f is of the form $?a$, $!a$, or $?a \rightarrow !b$, where $a, b \in \Sigma$. We use the notation $f : w$ for the action of an operation f applied to channel contents w. This action is defined as follows:

- $?a$ is an *input operation* that removes an a from the channel. The channel contents must be of the form $a \cdot w$ for this operation to be *enabled*. The action of $?a$ on channel contents $a \cdot w$ is w; that is, $?a : a \cdot w = w$.
- $!a$ is an *output operation* that transform the channel from w to $w \cdot a$; that is, $!a : w = w \cdot a$.
- $?a \rightarrow !b$ is a *conditional output operation* that is enabled if the channel contents is of the form $a \cdot w$. The resulting channel contents is $w \cdot b$; that is, $?a \rightarrow !b : a \cdot w = w \cdot b$.

For any Σ, the set of operations is denoted Σ_{op}. If we denote the set of input operations by Σ_{in}, the set of output operations by Σ_{out}, and the set of conditional output operations by Σ_{cout}, then we have $\Sigma_{\text{op}} = \Sigma_{\text{in}} \cup \Sigma_{\text{out}} \cup \Sigma_{\text{cout}}$.

We sometimes consider systems with more than one channel. In those cases, the channels have names in a finite set assumed to be of the form $\{0, \ldots, K-1\}$. A *channel operation* f in a multiple channel system is of the form $k?a$ (read a from channel k), $k!a$ (write a to channel k) or $k?a \rightarrow k'!b$ (write b to channel k' while reading a from channel k), where $0 \leq k, k' < K$. For systems with more than one channel, we define Σ_{op} to be the set of all operations; Σ_{in}, Σ_{out}, and Σ_{cout} are defined as for one-channel systems.

2.3 Describing channel contents

The *joint content* at a given instant for systems with K channels is a mapping $\mathbf{w} \in \{0, \ldots, K-1\} \mapsto \Sigma^*$. We usually regard joint content as a word of the form $w = w_0 \cdot \# \cdots \# \cdot w_{K-1}$, where $\#$ is not in Σ and $w_i = \mathbf{w}(i)$ for $0 \leq i < K$. A set of such words is called a *channel language*.

3 Operations languages acting on channel languages

3.1 Operation languages

Operations are extended to act on channel languages:

$$f : L = \{f : w \mid f \text{ is enabled on } w\}$$

Furthermore, operations are homomorphically extended to strings of operations according to:

$$\epsilon : L = L \qquad \text{and} \qquad f \cdot t : L = t : f : L,$$

where $f \in \Sigma_{\text{op}}$ and $t \in \Sigma_{\text{op}}^*$. An *operation language* T is a subset of Σ_{op}^*. The action $T : L$ of T on L is defined as $\{t : w \mid w \in L, t \in T\}$. We will also sometimes use T as a symbol denoting a regular expression over Σ_{op}. The following is well-known, see [6], except for the emphasis on piecewiseness:

Proposition 3.1 *For the single channel case, the following hold:*

(a) For regular (piecewise) L, it holds that $?a : L$, $!a : L$, $?a \rightarrow !b : L$, $?a^* : L$, and $!a^* : L$ are regular (piecewise).

(b) For any $F \subseteq \Sigma_{\text{op}}$ and any regular (piecewise) L, $F : L$ is regular (piecewise).

(c) For any $F \subseteq \Sigma_{\text{in}} \cup \Sigma_{\text{out}}$ and any regular (piecewise) L, $F^* : L$ is regular (piecewise).

Proof.

(a) *Case* $?a : L$. We have $?a : L = a^{-1}L$. *Case* $!a : L$. Clearly, $!a : L = L \cdot a$. *Case* $?a \rightarrow !b : L$. We have $?a \rightarrow !b : L = a^{-1}(L \cdot b)$.

(b) We have that $F : L = \sum_{f \in F} f : L$. The statement follows from this observation.

(c) Let $M = \{a \mid ?a \in F\}$ and $N = \{a \mid !a \in F\}$. Then it can be seen that $F^* : L = (M^*)^{-1}(L \cdot N^*)$.

□

3.2 One-channel systems with conditional operations

We note that an occurrence of letter a in an initial word w may be transformed to an occurrence of b through the use of a conditional operation $?a \rightarrow !b$ when the occurrence of a is at the beginning of the channel. We call the occurrence of b a *transform* of the occurrence of a. During repeated applications of F, an initial letter occurrence may be repeatedly transformed, each time the occurrence again is at the front of the channel. But it is also possible that a transform is dropped due to an unconditional input operation. In this case, we insert an invisible marker ϵ so that the channel contents always can be understood as a transformed, cyclic permutation of the initial word. We shall make these notions more precise below in the proof of the following:

Proposition 3.2

(a) For any $F \subseteq \Sigma_{\text{op}}$ and any regular (piecewise) L, $F^* : L$ is regular (piecewise).

(b) For any piecewise regular $T \subseteq \Sigma_{\mathrm{op}}$, $T : L$ is regular (piecewise) if L is regular (piecewise).

Proof. (a) [Very brief idea] Let $F_{\mathrm{in}} = F \cap \Sigma_{\mathrm{in}}$, $F_{\mathrm{out}} = F \cap \Sigma_{\mathrm{out}}$, and $F_{\mathrm{cout}} = F \cap \Sigma_{\mathrm{cout}}$. To keep track of unconditionally introduced letters and their transforms, define the monotone function ψ on 2^{Σ} according to

$$\psi(M) = F_{\mathrm{out}} \cup F_{\mathrm{cout}}(M)$$

Use ϕ to keep track of what may happen to a letter occurrence a in a word of L:

$$\phi(M) = F_{\mathrm{cout}}(M) \cup \begin{cases} \{\epsilon\} & \text{if } \epsilon \in M \text{ or } M \cap F_{\mathrm{in}} \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

where we use the symbol ϵ to represent the case that the letter or its transform has been dropped. We shall study the effect of repeated applications of operations in F to an initial word $w = a_0 \cdots a_{n-1}$. To do so, we keep track of the *abstract transformations* determined by the transforms of the letter occurrences a_k for $k < n$. We write a series of languages $L_{i,j}$ that describe the various stages and use properties of ϕ and ψ to prove that finitely many of these languages describe the limit.

□

Iterating a set of operations is stronger than repeatedly iterating each of the operations:

Proposition 3.3 *There is a set $F = \{f, g\}$ of single-channel operations and a piecewise language L such that $F^* : L$ can be calculated from Proposition 3.2 and such that $(f^* \cdot g^*)^k : L$ denotes a strictly increasing sequence of languages.*

Proof. Consider $\Sigma = \{a, b, c\}$, $f = ?a!b$, $g = ?a!c$, $L = \mathcal{L}(a^*)$. Then, $F^* : L = \Sigma^* \cdot \{b, c\}^*$ and $(f^* \cdot g^*)^k : L = \Sigma^* \cdot (b^* \cdot c^*)^k$. □

This shows that our results strengthen the acceleration technique of [6] (see the discussion in Section 1.3). The more elaborate techniques of [5] do also work for this example, because each element of F reads the same letter—thus the condition of read synchronization (see Section 1.3) holds. If we instead use $\Sigma = \{a, a', b, c\}$, $f = ?a!b$, and $g = ?a'!c$, then this condition does not hold; thus Proposition 3.2 strengthens the results of [5] as well. Note, that an instance of this scenario is found in our feature described in Figure 2.

3.3 Multiple-channel systems with unconditional operations

A *K-operation language* is a set of finite sequences of channel operations for a system with K channels. We consider here the case $K = 2$, since the generalization to systems with more than two channels will be straightforward.

Proposition 3.4 *Consider a multiple channel system subjected to only unconditional operations.*

(a) For F with $F \subseteq \Sigma_{\text{in}} \cup \Sigma_{\text{out}}$, $F^* : L$ is regular (piecewise) if L is regular (piecewise). The same holds for $f \in \Sigma_{\text{op}}$ acting on L.

(b) For a piecewise expression T containing only unconditional operations, $T : L$ is regular (piecewise) if L is regular (piecewise).

Proof.

(a) According to Proposition 2.7, we may assume that L is $\sum_{0 \le i < I} R_{i,0} \# R_{i,1}$ for some I. Define $M_k = \{a \mid k?a \in F\}$ and $N_k = \{a \mid k!a \in F\}$ for $k = 0$ and $k = 1$. Then,

$$F^* : L = \sum_{0 \le i < I} (M_0^*)^{-1}(R_{i,0} \cdot N_0^*) \# (M_1^*)^{-1}(R_{i,1} \cdot N_1^*)$$

Thus, $F^* : L$ is regular (piecewise) if L is regular (piecewise). In a similar vein, using Proposition 3.1(a), we have for any single $f \in \Sigma_{\text{in}} \cup \Sigma_{\text{out}}$ that f acting on L preserves regularity and piecewiseness.

(b) By assumption, T is the sum of simply piecewise expressions of the form $F_0^* f_0 \cdots f_\ell F_{\ell+1}^*$ on which (a) can be applied inductively.

□

These results also do not rely on a condition of read synchronization [5] imposed on F.

3.4 Multiple-channel systems with conditional operations

Even when we consider just two-channel systems seeded by a regular language, the limit language for a set of operations F may be non-regular. Consider $F = \{0?a \to 1!a, 0?b \to 0!b', 0?b' \to 1!b\}$ and the initial channel language $L_{ini} = (ab)^*\#$. The idea is that first all the a's are transferred to channel 1, then all the b's (after each b has temporarily been renamed to b'). We have that $F^*(L_{ini}) \cap \mathcal{L}(\#a^*b^*) = \{\#a^n b^n \mid n \ge 0\}$; hence, $F^* : L_{ini}$ is not regular.

However, if the initial language is piecewise, then the limit language is also piecewise.

Proposition 3.5 *For a multiple channel system, $F^* : L$ is piecewise if L is piecewise.*

Proof. (Idea) $F^* : L$ is a countable union of languages that express arbitrary finite interleavings of channel operations. We work with piecewise expressions describing the joint content, and we show that operations do not change the anchor length. We then use Proposition 2.9. □

Unfortunately, the proof of Proposition 3.5 is non-effective: it provides us with no algorithm for calculating the limit language as a function of F and L.

For systems that do not allow a letter a to be passed around in cycles through different channels, we are able to provide an effective algorithm for calculating $T : L$.

We say that the *communication structure* $CS(F)$ is the directed graph on $\{0, \ldots, K-1\}$ with edges $\{(k, k') \mid \exists a, b : k?a \to k'!b \in F\}$. The communication structure of F is *acyclic* if $CS(F)$ is acyclic. And, the communication structure of a piecewise expression T is acyclic if each set F occurring in T is acyclic.

Proposition 3.6 *From a piecewise expression T (denoting a K-operation language) with acyclic communication structure and a piecewise expression L (denoting a K-channel language) a piecewise expression for $T : L$ can be effectively constructed.*

4 Summary

Our results for piecewise operations languages can be summarized in the following table:

Channel configuration

	one	multiple w/o conditionals	multiple w acyclic comm. struct.	multiple
piecewise	eff. piecewise	eff. piecewise	eff. piecewise	piecewise
regular	eff. regular	eff. regular	non-regular	non-regular

The row is the type of language describing initial channel content and the column is the type of the channel configuration. In all cases, we assume that the operation language is itself piecewise. As indicated, our results for extracting the regular or piecewise limit language are generally effective (algorithms are implicit in our proofs).

References

[1] Well quasi ordering. PlanetMath.Org, http://planetmath.org/encyclopedia/Wqo.html.

[2] P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In *Proc. of TACAS*, volume 1579 of *LNCS*, pages 208–222, 1999.

[3] P. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy Fifo channels. In *Proc. 10th Intern. Conf. on Computer Aided Verification*, volume 1427 of *LNCS*, 1998.

[4] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Logic in Computer Science*, pages 160–170, 1993.

[5] B. Boigelot. *Symbolic method for exploring infinite states spaces.* PhD thesis, Université de Liège, 1998.

[6] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. *Formal Methods in Systems Design*, 14(3), 1999.

[7] B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In *Proceedings of the 4th International Symposium on Static Analysis*, volume 1302 of *LNCS*, pages 172 – 186, 1997.

[8] G. W. Bond, F. Ivančić, N. Klarlund, and R. Trefler. Eclipse feature logic analysis. *Second IP Telephony Workshop*, 2001.

[9] A. Bouajjani, A. Muscholl, and T. Touili. Permutation rewriting and algorithmic verification. In *Proc. 16th IEEE Symp. on Logic in Computer Science (LICS'01)*, 2001. Full version available at http://www.liafa.jussieu.fr/~abou/Papers/acp-versionlongue.ps.gz.

[10] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.

[11] A. Brzozowski and I. Simon. Characterization of locally testable events. *Discrete Math.*, 4:243–271, 1973.

[12] J. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4), 1964.

[13] J. A. Brzozowski and F. E. Fich. Languages of r-trivial monoids. *J. Comput. Syst. Sci.*, 20:32–49, 1980.

[14] G. Cece, A. Finkel, and S. P. Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[15] A. Finkel, S. P. Iyer, and G. Sutre. Well-abstracted transition systems: Application to FIFO automata. *Information and Computation*, 181(1):1–31, 2003.

[16] A. Finkel and L. Rosier. A survey on the decidability questions for classes of FIFO nets. In *Advances in Petri Nets*, LNCS 340, pages 106–132. Springer, 1988.

[17] J. K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. *Proc. of Protocol Specification, Testing and Verification*, 1987.

[18] J.K. Pachl. Reachability problems for communicating finite state machines. Technical report, University of Waterloo, May 1982. Research Report CS-82-12.

[19] J.-E. Pin. Syntactic semigroups. In Rozenberg and Salomaa, editors, *Handbook of language theory, Vol. I.* Springer Verlag, 1997.

[20] J.-E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory Comput. Systems*, 30:1–39, 1997.

[21] I. Simon. Piecewise testable events. *Proc. 2nd GI Conf.*, 33:214–222, 1975.

[22] S. Yu. Regular languages. In Rozenberg and Salomaa, editors, *Handbook of language theory, Vol. I.* Springer Verlag, 1997.