RATIONAL EQUIVALENCE RELATIONS

J. Howard Johnson

Department of Computer Science
University of Waterloo
Waterloo, Ontario
N2L 3G1

Research Report CS-86-16
April 1986

Rational Equivalence Relations*

J. Howard Johnson
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

April 11, 1986

Abstract

Rational relations (finite transductions) which are equivalence relations are discussed. After establishing a containment hierarchy, the complexity of canonical function computation and a number of class membership decision problems are studied. The following classes are considered:

(1) Rational Equivalence Relations,
(2) Equivalence Kernels of Rational Functions,
(3) Deterministic Rational Equivalence Relations,
(4) Equivalence Kernels of Subsequential Functions,
(5) Recognizable Equivalence Relations,
(6) Length-bounded Rational Equivalence Relations, and
(7) Finite Equivalence Relations.

Except for one open case ((1) = (2)?), Hasse diagrams are given to show the relative containments in the general and one-letter-alphabet cases. Canonical function application for an input of length n is shown to be $O(n^2)$ time and space for (1), $O(n)$ time and space for (2), (3) and (6), and $O(n)$ time and constant space for the others. It is shown that transitivity, symmetry, reflexivity, and membership in any of (1) through (5) are undecidable properties for rational relations whereas membership in (6) or (7) is decidable.

1 Introduction

Rational relations or finite transductions can be used to describe a natural class of relations on words just as regular languages describe a natural class of sets of words. Of particular interest are the functions or

*This work was supported by the Natural Sciences and Engineering Research Council of Canada, Grant No. A0237, and appears in abbreviated form in the proceedings of the 13th ICALP (1986).

single-valued relations which associate a unique range value to each domain value and thus provide a good model for parsing. Rational relations which are functions have been studied by a number of authors, for example, [BH77,Eil74,EM65,Niv68,Sch61,Sch75]. Chapter IV of [Ber79] provides an excellent introduction to the subject with references to the original literature.

Another important class of relations are the equivalence relations. These arise naturally as the equivalence kernels of functions or can be defined directly as relations satisfying the reflexive, symmetric, and transitive laws. Thus we can introduce rational equivalence relations either as the equivalence kernels of rational (or subsequential) functions, or directly as rational (or deterministic rational or recognizable or length-bounded rational or finite) relations which satisfy the three required properties.

The motivation for considering rational equivalence relations arose from the study of functions used to phonetically encode names for matching purposes as advocated by [Dav62,FS69,Knu73,MKTM77,NK62,Wie77]. For example, Soundex code [Knu73,NK62] maps surnames into four-character codes so that similar-sounding names are mapped to the same code and different-sounding names are assigned different codes. Thus "Johnson" and "Jansen" are both transformed into the code "J525" and "Smith" is transformed into "S530". In order to improve on Soundex, a number of attempts have been made to construct coding functions that more closely reflect the structure of surnames in the population of interest. For example, the NYSIIS code was designed to achieve greater accuracy on a population with a significant fraction of Spanish surnames. A number of coding functions, including NYSIIS, are discussed by Moore *et al* [MKTM77].

Both the Soundex and NYSIIS functions can be modelled as subsequential functions [Joh83] so that they can be computed deterministically from left to right by a finite state machine with output. Thus they are rational functions and the relation "has the same code" is a rational equivalence relation in both cases.

The principal advantage of using coding functions to identify similar strings is low cost. Even very large files can be partitioned cheaply according to a code value using any of a number of sorting or hashing algorithms. Since the coding function need only be computed once per record, a moderate amount of effort can be directed into computing better codes if accuracy can be improved. Thus more general equivalence relation models which more accurately mirror the true error process are worthy of consideration.

In spite of the fact that a moderate computation cost can be tolerated in the interest of greater accuracy, it is important to recognize special cases where more efficient or simpler algorithms are possible. The case of Soundex is illuminating. Since only a finite number of codes are possible, 6734 to be

exact, and each code corresponds to a regular language, we could construct a semi-automaton that recognizes all of these languages simultaneously and whose states identify the correct code. Such a semi-automaton will have 7845 states. On the other hand, Soundex has a 23-state subsequential transducer [Joh83]. If the length restriction to four characters is removed, only seven states are needed. As a result, it is of interest to study subclasses of rational equivalence relations from the point of view of their relative modelling power and canonical function costs as well as considering when transformations between classes exist and are effective. For this purpose the relative containments and decidabilities addressed in this paper are an important first step.

2 Terminology

A (binary) *relation* over sets S and T is a subset of $S \times T$. We will be interested in the Boolean operations on relations as well as composition, inversion, sub-identity, domain, range, cross-product, and application:

$$R_1 \circ R_2 = \{(u,w) | \exists v[(u,v) \in R_1 \wedge (v,w) \in R_2]\}$$

$$R^{(-1)} = \{(v,u) | (u,v) \in R\} \qquad \iota_L = \{(u,u) | u \in L\}$$

$$\mathrm{dom}R = \{u | \exists v[(u,v) \in R]\} \qquad \mathrm{ran}R = \{v | \exists u[(u,v) \in R]\}$$

$$L_1 \times L_2 = \{(u,v) | u \in L_1 \wedge v \in L_2\}$$

$$R(L) = \{v | \exists u[u \in L \wedge (u,v) \in R]\}.$$

An *equivalence relation* R over a set S is a relation satisfying the reflexive, symmetric, and transitive laws: $\iota_S \subseteq R$, $R^{(-1)} \subseteq R$, and $R \circ R \subseteq R$. The *(equivalence) kernel* of a function $f: S \rightarrow T$ is the equivalence relation over S:

$$\ker f = \{(u,v) \in S \times S | f(u) = f(v)\} = f \circ f^{(-1)}.$$

A *canonical function* for an equivalence relation R over S is any function $f: S \rightarrow T$ satisfying $R = \ker f$. A *cross-section* of an equivalence relation R over S is a set D containing one element from each class of R. Then $f = R \cap (S \times D)$ is a canonical function. The *restriction* $R|D$ of an equivalence relation R to a set D is the relation formed from R by restricting the domain and range to D. Thus $R|D = R \cap (D \times D)$. A *thinning* of an equivalence relation R is a restriction whose domain contains at least one member of each equivalence class of R. A relation R is *locally-finite* if for any $x \in \mathrm{dom}R$, the set $R(\{x\})$ is finite. If R is an equivalence relation then local-finiteness requires that every class be finite in size.

A *monoid* $\langle M, \odot, \mathbf{1} \rangle$ is a set M with an associative binary operation $\odot$ and a (left and right) identity element $\mathbf{1}$. We can extend $\odot$ to subsets of M and define the $\odot$-closure of a subset of M:

$$S \odot T = \{a \odot b | a \in S \wedge b \in T\}$$

$$S^1 = S \qquad S^k = S^{k-1} \odot S \qquad S^+ = \bigcup_{i=1}^{\infty} S^i \qquad \forall S, T \subseteq M$$

The class of *rational subsets* of a monoid $\langle M, \odot, \mathbf{1} \rangle$, denoted $\mathbf{Rat}(M)$, is defined as all sets derivable from finite sets using a finite number of applications of the (rational) operations union, $\odot$, and $\odot$-closure. Note that $S^* = \{\mathbf{1}\} \cup S^+$ will be rational if S is.

The first type of monoid that we will consider is the finitely generated free monoid $\langle \Sigma^*, \cdot, \Lambda \rangle$. The class $\mathbf{Rat}(\Sigma^*)$ corresponds to the family of regular languages over Σ^*. It is well known that this class is a Boolean algebra and can be characterized by deterministic finite automata (finite state machines). It is possible to characterize a disjoint collection of regular languages using a *semi-automaton*, a 3-tuple $\langle \Sigma, Q, \delta \rangle$ where Σ is the alphabet, Q is a set of states, and $\delta: Q \times \Sigma \to Q$ is a set of transitions.

The second type of monoid is the direct product of two finitely generated free monoids $\langle \Sigma^* \times \Delta^*, \cdot, (\Lambda, \Lambda) \rangle$ where the operation $\cdot$ is defined componentwise. The subsets in the class $\mathbf{Rat}(\Sigma^* \times \Delta^*)$ are called *(binary) rational relations*. In addition to the rational operations, they are closed under composition, inversion, sub-identity, domain, range, cross-product, application, and domain and range restriction:

$$\left.\begin{array}{c} R_1 \in \mathbf{Rat}(\Sigma^* \times \Delta^*) \\ R_2 \in \mathbf{Rat}(\Delta^* \times \Gamma^*) \\ L_1 \in \mathbf{Rat}(\Sigma^*) \\ L_2 \in \mathbf{Rat}(\Delta^*) \end{array}\right\} \Longrightarrow \left\{\begin{array}{c} R_1 \circ R_2 \in \mathbf{Rat}(\Sigma^* \times \Gamma^*) \\ R_1^{(-1)} \in \mathbf{Rat}(\Delta^* \times \Sigma^*) \\ \iota_{L_1} \in \mathbf{Rat}(\Sigma^* \times \Sigma^*) \\ \mathrm{dom} R_1 \in \mathbf{Rat}(\Sigma^*) \\ \mathrm{ran} R_1 \in \mathbf{Rat}(\Delta^*) \\ L_1 \times L_2 \in \mathbf{Rat}(\Sigma^* \times \Delta^*) \\ R_1(L_1) \in \mathbf{Rat}(\Delta^*) \\ R_1 \cap (L_1 \times L_2) \in \mathbf{Rat}(\Sigma^* \times \Delta^*) \end{array}\right.$$

They are not closed under the Boolean operations intersection or set difference except in special cases. One such special case occurs when Σ and Δ each contain exactly one letter. Such rational relations form a Boolean algebra [Gin66]. Another special case is the class of *recognizable relations* $\mathbf{Rec}(\Sigma^* \times \Delta^*)$ which contains rational relations expressible as a finite union of products of regular languages: $R = \bigcup_{i=1}^{m} A_i \times B_i$. This class is also a Boolean algebra but not closed under $^+$, *, or ι_L. Another special case is

the class of *length-bounded rational relations* $\mathbf{LBRat}(\Sigma^* \times \Delta^*)$ which contains rational relations satisfying the following property for some k:

$$(u,v) \in R \Longrightarrow |v| - k \leq |u| \leq |v| + k.$$

It is closed under $^+$ and * for relations satisfying the length property with $k = 0$ and under $\cap$ and $-$ but not under $\times$, and since the universal element $\Sigma^* \times \Delta^*$ is not length-bounded, it is not a Boolean algebra. The class $\mathbf{Fin}(\Sigma^* \times \Delta^*)$ is closed under $\cap$ and $-$ but not $^+$, *, or ι_L and is not a Boolean algebra because it also lacks the universal element.

A *finite transducer* $\mathbf{T} = \langle \Sigma, \Delta, Q, q_-, Q_+, E \rangle$ is a 6-tuple where Σ is the tape-one alphabet, Δ is the tape-two alphabet, Q is a finite set of states, $q_- \in Q$ is a distinguished start state, $Q_+ \subseteq Q$ is a distinguished set of final states, and $E \subseteq (Q \times \Sigma^* \times \Delta^* \times Q)$ is a set of transitions. A path from q_0 to q_k through $\mathbf{T}$ is a sequence of transitions from E of the form

$$(q_0, u_1, v_1, q_1)(q_1, u_2, v_2, q_2)(q_2, u_3, v_3, q_3) \ldots (q_{k-1}, u_k, v_k, q_k).$$

A successful path is one where $q_0 = q_-$ and $q_k \in Q_+$. The label of a path is the componentwise concatenation of the labels: $(u_1u_2u_3 \ldots u_k, v_1v_2v_3 \ldots v_k)$. The behaviour $|\mathbf{T}|$ of a transducer $\mathbf{T}$ is the set of labels of successful paths. The class $\mathbf{Rat}(\Sigma^* \times \Delta^*)$ is identical to the class of finite transductions. At the expense of adding some states, we can restrict finite transducers to those with transitions of the form:

$$(Q \times \Sigma \times \Lambda \times Q) \cup (Q \times \Lambda \times \Sigma \times Q) \cup (Q \times \Lambda \times \Lambda \times Q).$$

Such a transducer will be called *alphabetic*.

A *deterministic 2-tape finite automaton* $\mathbf{A} = \langle \Sigma, \Delta, Q_1, Q_2, q_-, Q_+, \delta_1, \delta_2 \rangle$ is an 8-tuple where Σ and Δ are the tape-one and tape-two alphabets, Q_1 and Q_2 are disjoint finite sets of states, $q_- \in Q_1 \cup Q_2$ is a distinguished start state, $Q_+ \subseteq Q_1 \cup Q_2$ is a distinguished set of final states, and

$$\delta_1 : Q_1 \times (\Sigma \cup \{\dashv\}) \to Q_1 \cup Q_2 \qquad \delta_2 : Q_2 \times (\Delta \cup \{\dashv\}) \to Q_1 \cup Q_2$$

are partial functions. The symbol $\dashv$ is a special endmarker character not occurring in either Σ or Δ. Every such machine can be interpreted as the alphabetic finite transducer:

$$\mathbf{T_A} = \langle \Sigma \cup \{\dashv\}, \Delta \cup \{\dashv\}, Q_1 \cup Q_2, q_-, Q_+, \bigcup_{q \in Q_1, a \in \Sigma \cup \{\dashv\}} (q, a, \Lambda, \delta_1(q,a)) \cup \bigcup_{q \in Q_2, b \in \Delta \cup \{\dashv\}} (q, \Lambda, b, \delta_2(q,b)) \rangle.$$

Paths, successful paths, and labels are then defined implicitly in terms of this transducer. The behaviour $|\mathbf{A}|$ of an automaton $\mathbf{A}$ is the set of pairs (u, v)

such that $(u \dashv, v \dashv) \in |\mathbf{T_A}|$. Deterministic 2-tape finite automata recognize a subclass of rational relations [FR68] which will be denoted as $\mathbf{DetRat}(\Sigma^* \times \Delta^*)$. Deterministic rational relations are closed under complement but not $\cup$, $\cdot$, $^+$, *, or $\circ$. However, the classes $\mathbf{RecEq}(\Sigma^* \times \Delta^*)$ and $\mathbf{LBRat}(\Sigma^* \times \Delta^*)$ with their rich closure properties are both contained in $\mathbf{DetRat}(\Sigma^* \times \Delta^*)$.

Although it is not possible to make the choice of tape to read deterministic without restricting the class of relations that are recognized, it is possible to make the choice of transition deterministic after the tape to read has been chosen. A *quasi-deterministic 2-tape finite automaton* $\mathbf{A} = \langle \Sigma, \Delta, Q, q_-, Q_+, \delta_1, \delta_2 \rangle$ is an 7-tuple where Σ and Δ are the tape-one and tape-two alphabets, Q is a set of states, $q_- \in Q$ is a distinguished start state, $Q_+ \subseteq Q$ is a distinguished set of final states, and

$$\delta_1 : Q \times (\Sigma \cup \{\dashv\}) \to Q \qquad \delta_2 : Q \times (\Delta \cup \{\dashv\}) \to Q$$

are partial functions. Successful paths and behaviour are defined in a similar fashion to deterministic automata except that the state set is not partitioned according to which tape should be read next. This class of automata recognize the full class $\mathbf{Rat}(\Sigma^* \times \Delta^*)$.

A *subsequential transducer* $\mathbf{S} = \langle \Sigma, \Delta, Q, q_-, \delta, \lambda, \rho \rangle$ is a 7-tuple where Σ is the input alphabet, Δ is the output alphabet, Q is a set of states, $q_- \in Q$ is a distinguished start state, and

$$\delta : Q \times \Sigma \to Q \qquad \lambda : Q \times \Sigma \to \Delta^* \qquad \rho : Q \to \Delta^*$$

are partial functions indicating the next state, the output function and the termination output function. Every such machine can be interpreted as a finite transducer:

$$\mathbf{T_S} = \langle \Sigma, \Delta, Q \cup \{q_+\}, q_-, \{q_+\}, \bigcup_{q \in Q, a \in \Sigma} (q, a, \lambda(q,a), \delta(q,a)) \cup \bigcup_{q \in Q} (q, \Lambda, \rho(q), q_+) \rangle.$$

Paths, successful paths, labels, and behaviour can be then defined implicitly in terms of this transducer (or directly as in [Ber79]). A *rational function* is a rational relation that is single-valued. Relations that are the behaviour of subsequential transducers constitute a subclass called *subsequential functions*. These two classes will be denoted $\mathbf{RatF}(\Sigma^* \times \Delta^*)$ and $\mathbf{SSeqF}(\Sigma^* \times \Delta^*)$. Note that the class $\mathbf{DetRatF}(\Sigma^* \times \Delta^*)$ of single-valued deterministic rational relations is different from both of these.

For the basic theory of rational relations and rational functions see [Ber79,Eil74]. For a discussion of the properties of deterministic rational relations see [FR68]. Length-bounded rational relations are discussed in [EM65].

A *lexicographic order* $\prec$ on Σ^* is the total order induced by an order on Σ using the rules:

$$u \prec uav, \qquad a \prec b \Longrightarrow uav \prec ubw \qquad \forall u, v, w \in \Sigma^* \quad \forall a, b \in \Sigma.$$

The set of words that are lexicographically minimal within their classes for a rational equivalence relation R will be denoted by lexmin(R). Since lexicographic ordering is not a well order, there may be classes without minimal elements so that lexmin(R) might not be a cross-section. The set of words that are minimal length within their classes for a rational equivalence relation R will be denoted by lenmin(R). This may fail to be a cross-section since classes may have more than one element of minimal length.

The subclass of $\mathbf{Rat}(\Sigma^* \times \Sigma^*)$ that are equivalence relations over their domains will be denoted $\mathbf{RatEq}(\Sigma^*)$ or simply **RatEq** if the alphabet is understood. In an analogous way the following notations will be used:

KerRatF:	Equivalence Kernels of Rational Functions,
DetRatEq:	Deterministic Rational Equivalence Relations,
KerSSeqF:	Equivalence Kernels of Subsequential Functions,
RecEq:	Recognizable Equivalence Relations,
LBRatEq:	Length-bounded Rational Equivalence Relations, and
FinEq:	Finite Equivalence Relations.

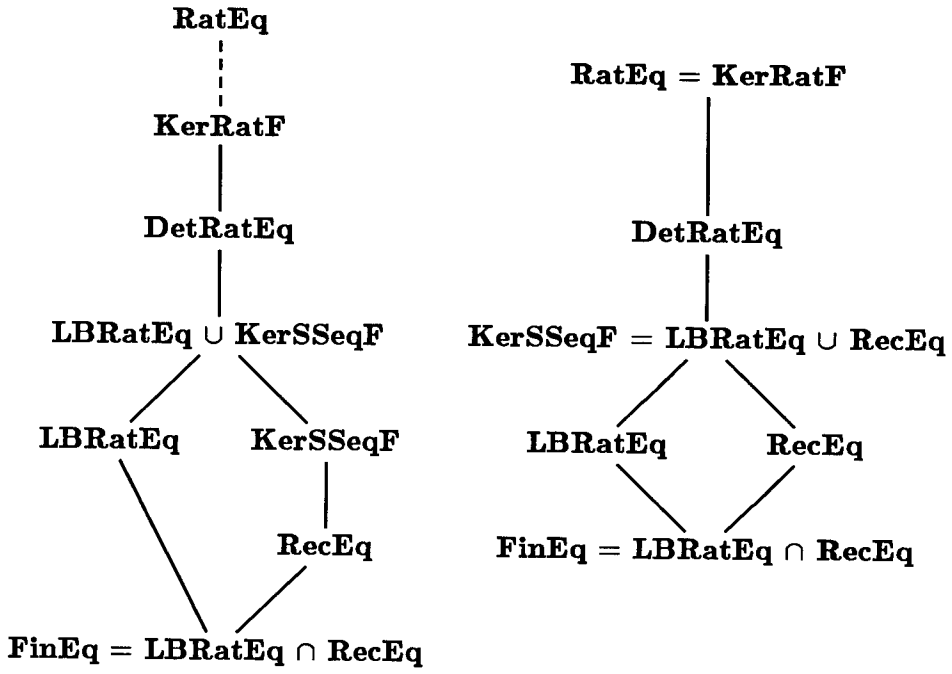
It is easily shown that the recognizable equivalence relations are exactly the rational equivalence relations of finite index [Joh83].

In section 3 the relative containments of these classes are shown in general and for the one-letter-alphabet case. Section 4, through a system of examples, shows that the containments presented in section 3 are proper and that incomparable classes are truly so. Section 5 discusses the question of complexity of canonical function application in terms of the input word length. Section 6 presents a number of decidability results concerning class membership. Finally, section 7 presents conclusions and some open problems remaining from this investigation.

3 A Containment Hierarchy

This section and the following one establish a hierarchy of rational equivalence relations as summarized in the following theorem:

Theorem 3.1 *Rational equivalence relations can be organized into the containment hierarchy as shown in the form of a Hasse diagram in Figure 1(a). If the alphabet is restricted to contain one letter the hierarchy collapses into that shown in Figure 1(b).*



(a) General Case (b) One-Letter-Alphabet Case

Figure 1: Containment hierarchies for rational equivalence relations

The remainder of this section establishes the containments of theorem 3.1. The next section shows that the indicated containments are proper with the exception of **KerRatF** $\subseteq$ **RatEq** which remains open. It has been conjectured that these two classes are the same [Joh85].

Lemma 3.2 **KerRatF** $\subseteq$ **RatEq**.

Proof: Let $R \in$ **KerRatF**. Then there is a rational function f such that $R = \ker f = f \circ f^{(-1)}$. Since rational relations are closed under inverse and composition, $R \in$ **RatEq**. □

Lemma 3.3 **DetRatEq** $\subseteq$ **KerRatF**.

Proof: This follows from Theorems 5.3 and 3.1 of [Joh85]. □

Lemma 3.4 **LBRatEq** $\subseteq$ **DetRatEq**.

Proof: This follows immediately from the observation that $\mathbf{LBRat}(\Sigma^* \times \Delta^*) \subseteq \mathbf{DetRat}(\Sigma^* \times \Delta^*)$. For if $R \in \mathbf{LBRat}(\Sigma^* \times \Delta^*)$ then, in the terms of Elgot and Mezei [EM65], R is finite automaton describable (FAD) so that R can be recognized by an finite automaton that reads one character (or blank) from each tape for each transition until both tapes are exhausted. Since this behaviour can be simulated by a deterministic 2-tape automaton which alternately reads one character from each tape, $R \in \mathbf{DetRat}(\Sigma^* \times \Delta^*)$. □

Lemma 3.5 $\mathbf{KerSSeqF} \subseteq \mathbf{DetRatEq}$.

Proof: Let $R \in \mathbf{KerSSeqF}$. Then there is a subsequential function f such that $R = \ker f = f \circ f^{(-1)}$ which can be characterized by some subsequential transducer: $\mathbf{T} = \langle \Sigma, \Delta, Q, q_-, \delta, \lambda, \rho \rangle$. From this we may construct a deterministic 2-tape automaton for R that simulates the running of $\mathbf{T}$ on each of its input tapes while checking that they agree on their outputs. The states of this derived machine will be composed of the states of the two copies of $\mathbf{T}$ as well as a pair of buffers each large enough to accommodate the largest λ or ρ string. At any time only one buffer will be allowed to be non-empty and the simulation will read the input tape corresponding to the other buffer. That buffer is loaded with the appropriate letters from the λ or ρ strings and the longest common prefix removed. If the result leaves both buffers non-empty, then a mismatch has occurred and the machine simply deadlocks. □

Lemma 3.6 $\mathbf{RecEq} \subseteq \mathbf{KerSSeqF}$.

Proof: Let $R \in \mathbf{RecEq}$. Then $R \in \mathbf{RatEq}$ and R has finite index, say k [Joh83]. Since each equivalence class C_i $(1 \leq i \leq k)$ is a regular set the relation $f = \bigcup_{i=1}^{k} C_i \times \{a^i\}$ from Σ^* to $\{a\}^*$ is a canonical function for R. Since the classes C_i together with $C_0 = \Sigma^* - \mathrm{dom}(R)$ form a regular partition of Σ^* there is a semi-automaton $\mathbf{A} = \langle \Sigma, Q, \delta \rangle$ whose states identify a refinement of this partition. From this a subsequential transducer for f can be constructed: $\mathbf{T} = \langle \Sigma, \Delta, Q, \delta, \lambda, \rho \rangle$ where $\lambda(q, x) = \Lambda$ for all $q \in Q$ and $x \in \Sigma$ and $\rho(q) = a^i$ if state q corresponds to a subclass of C_i for $i \geq 1$ and $\rho(q)$ is undefined if q identifies a subclass of C_0. □

Lemma 3.7 $\mathbf{FinEq} = \mathbf{LBRatEq} \cap \mathbf{RecEq}$.

Proof: If $R \in \mathbf{FinEq}$ then R is a finite relation and therefore rational. Since R is of finite index $R \in \mathbf{RecEq}$. Since the maximum difference in length of related words is bounded $R \in \mathbf{LBRatEq}$.

If $R \in \mathbf{LBRatEq} \cap \mathbf{RecEq}$ then R has only a finite number of classes and each class is finite in size. Thus R can be described by enumerating a finite number of pairs of related elements and thus is in $\mathbf{FinEq}$. □

Lemma 3.8 *If* $|\Sigma| = 1$ *then* $\mathbf{RatEq} = \mathbf{KerRatF}$.

Proof: Rational relations over a one letter alphabet are closed under intersection [Gin66]. The relation $R_> = \{(a^m, a^n) | m > n\}$ is rational so that $L = \mathrm{dom}R - \mathrm{dom}(R \cap R_>)$ is a regular cross-section. As a result $R \cap (\mathrm{dom}R \times L)$ is a rational canonical function for R. □

Corollary 3.9 *When* $|\Sigma| = 1$ *we can restrict the output alphabet of rational transductions to one letter without changing the class* $\mathbf{KerRatF}$. □

Lemma 3.10 *If* $|\Sigma| = 1$ *then* $\mathbf{KerSSeqF} = \mathbf{LBRatEq} \cup \mathbf{RecEq}$.

Proof: Since $\mathbf{RecEq} \subseteq \mathbf{KerSSeqF}$, we need only prove that when $|\Sigma| = 1$, $\mathbf{LBRatEq} \subseteq \mathbf{KerSSeqF}$ and $\mathbf{KerSSeqF} \subseteq \mathbf{RecEq} \cup \mathbf{LBRatEq}$.

Let $\Sigma = \{a\}$ and $R \in \mathbf{LBRatEq}$. The set of words that are of minimum length in their class is a regular set: $L = \mathrm{dom}R - \mathrm{dom}(R \cap R_>)$. Thus the function that maps a word in $\mathrm{dom}R$ to the minimum length word in its equivalence class is a rational function. It is clearly also a length-bounded rational function. It is also subsequential as can be shown by construction. We assume that we are given a deterministic 2-tape finite automaton which reads one character (or blank) from each tape in each transition. We will then construct a machine that reads the input it is given and simulates a second input that is shorter than the given input by no more than k letters. It can manage this by keeping track of the current states reached if the virtual input is terminated at i characters less than the current input for i between 0 and k as well as the state reached if neither is terminated. These states are maintained in the finite control of the subsequential transducer. As input is read, a new state is added to the memory. If the memory does not overflow no output is written. If it does overflow the oldest state is removed and a single a is written. When the input is terminated enough a's are written to make the output correspond to the shortest virtual string that led to a final state.

Let $\Sigma = \{a\}$ and $R \in \mathbf{KerSSeqF}$ so that $R = \ker|\mathbf{T}|$ for some $\mathbf{T} = \langle \Sigma, \Delta, Q, q_-, \delta, \lambda, \rho \rangle$. Let $Q_+ = \{q \in Q | \rho(q) \in \Delta^*\}$. Suppressing the output part of $\mathbf{T}$ we obtain a deterministic finite automaton with the same states and transition structure: $\mathbf{A} = \langle \Sigma, Q, q_-, Q_+, \delta \rangle$. The accessible part of $\mathbf{A}$ must be a simple path of $s+1$ states $\{q_-, q_1, q_2, \ldots, q_{s-1}, r_0\}$ connected to a loop of p states $\{r_0, r_1, r_2, \ldots, r_{p-1}, r_0\}$. See for example [Eil74] for details. The structure of $\mathbf{T}$ will be the same except that write labels will be attached to each arc and final states will have a terminating ρ string.

Let $f = |\mathbf{T}|$. If none of the r_i's are in Q_+ then the domain of f is finite and so $\ker f$ is in $\mathbf{FinEq}$. In order that $\ker f$ be infinite, at least one of the loop states, say r_i, must be final. Then $f(a^{s+i+kp}) = uv^k w$

where $u = \lambda(q_-, a^{s+i})$, $v = \lambda(r_i, a^p)$, and $w = \rho(r_i)$. Let $\phi(n) = |f(a^n)|$, the length function of f and $\alpha = |v|/p$. Then $\phi(n) \leq \alpha n + \beta_2$ where $\beta_2 = \max_{m \leq s+i+p}\{\phi(m) - \alpha m\}$ and and $\phi(n) \geq \alpha n + \beta_1$ where $\beta_1 = \min_{m \leq s+i+p}\{\phi(m) - \alpha m\}$. In other words, we need only check once around the loop in order to ensure that the bounds work since $\mathbf{T}$ can only pump up by (a^p, v) (or by some cyclic shift of v).

Suppose now that $(a^n, a^{n_0}) \in \ker f$ so that $f(a^n) = f(a^{n_0})$ and $\phi(n) = \phi(n_0)$. There are two cases depending on whether α is equal to zero or not. If $\alpha > 0$ then the constraint $\alpha n + \beta_2 \geq \alpha n_0 + \beta_1$ implies that $n \geq n_0 - (\beta_2 - \beta_1)/\alpha$ and $\alpha n + \beta_1 \leq \alpha n_0 + \beta_2$ implies that $n \leq n_0 + (\beta_2 - \beta_1)/\alpha$. As a result it follows that $\ker f$ must be in **LBRatEq**.

If $\alpha = 0$ then the constraint $\phi(n) \leq \beta_2$ implies that the length of $f(a^n)$ must be β_2 or less. Thus the cardinality of the range of f is restricted to $\sum_{k \leq \beta_2} |\Delta|^k$ implying that $\ker f$ is of finite index and therefore in **RecEq**. □

Corollary 3.11 *When $|\Sigma| = 1$ we can restrict the output alphabet of subsequential transductions to one letter without changing the class* **KerSSeqF**.

Proof: Let f be a subsequential function with a one letter input alphabet. By the previous lemma, $\ker f$ is either in **LBRatEq** or in **RecEq**. But in either case another subsequential function using only a one letter output alphabet can be constructed. □

4 Proofs of Proper Containment

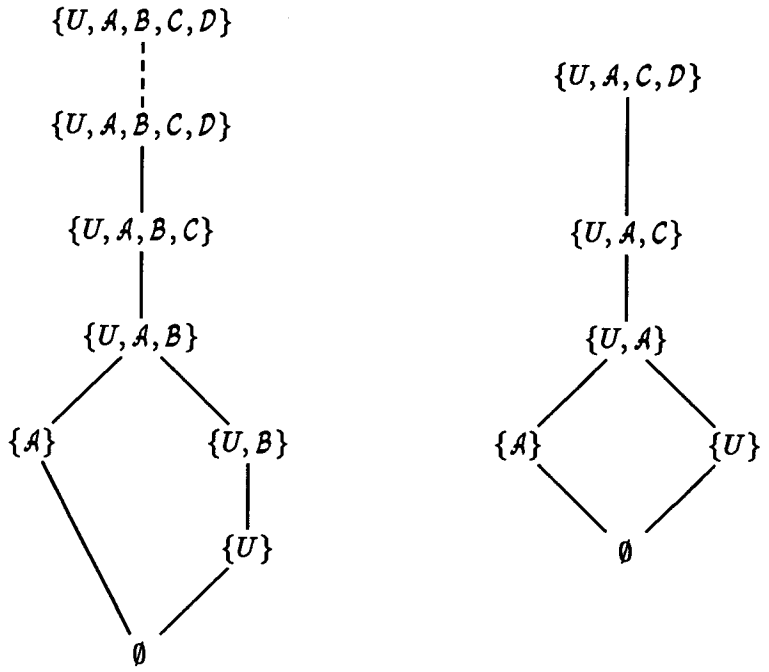
To show that the hierarchy of section 3 contains distinct classes, it is necessary to exhibit relations that occur in one class but not in a subclass. The following five relations will be used:

1. The universal relation $U = \Sigma^* \times \Sigma^*$ has one class containing all of Σ^*.

2. The relation $\mathcal{A} = ((\Sigma \times \Sigma)^2)^* \cup \iota_U$ has classes containing all words of the same even length and singleton classes for words of odd length. Note that $\mathcal{A}$ is the identity function when $|\Sigma| = 1$.

3. The relation $\mathcal{B} = ((a, a) \cdot U) \cup ((b, b) \cdot \iota_U)$ has one class containing all words beginning with a and singleton classes for each word beginning with b. This relation is undefined when $|\Sigma| = 1$.

4. The relation $\mathcal{C} = ((\Sigma^2)^* \times (\Sigma^2)^*) \cup (\Sigma \times \Sigma)^*$ has one class containing all even length words. Odd length words are grouped into classes by length.



(a) General Case (b) One-Letter-Alphabet Case

Figure 2: Containment hierarchies restricted to $\{U, A, B, C, D\}$

5. The relation $D = ((\Sigma\times\Sigma^2)\cdot(\Sigma^2\times\Sigma^4)^*)\cup((\Sigma^2\times\Sigma)\cdot(\Sigma^4\times\Sigma^2)^*)\cup(\Sigma\times\Sigma)^*$ has classes containing all words of some odd length together with all words of twice that length. Other words form classes by length.

Theorem 4.1 *The restriction of the hierarchies of theorem 3.1 to the five classes $\{U, A, B, C, D\}$ is as shown in the form of Hasse diagrams in Figures 2(a) and 2(b).*

Lemma 4.2 $U \in \mathbf{RecEq} - \mathbf{LBRatEq}$.

Proof: Since U is a rational equivalence relation with one class it is recognizable. It is not length-bounded since there is no value k satisfying the length-bound constraint. For any value k, $(\Lambda, a^{k+1}) \in U$. □

Lemma 4.3 $A \in \mathbf{LBRatEq} - \mathbf{RecEq}$. *If* $|\Sigma| \geq 2$ *then* $A \notin \mathbf{KerSSeqF}$.

Proof: Since $\mathcal{A}$ is described by a rational expression and satisfies a length-bound constraint $\mathcal{A} \in \mathbf{LBRatEq}$. It is of infinite index and thus not recognizable.

However if $|\Sigma| \geq 2$ it does not have a subsequential canonical function. Otherwise suppose $\mathcal{A} = f \circ f^{(-1)}$ where $\mathbf{S} = \langle \Sigma, \Delta, Q, q_-, \delta, \lambda, \rho \rangle$ and $f = |\mathbf{S}|$. Then for any even length $k = |w|$ $\lambda(q_-, w)\rho(\delta(q_-, w))$ must be a constant string independent of w. The second factor can only take on $|Q|$ different values since there are only $|Q|$ different ρ values. Thus for words of some fixed even length the first factor can take on at most $|Q|$ different values as well. However for each of the $|\Sigma|^k$ words of length k after reading an additional a, we must have $|\Sigma|^k$ different values $\lambda(q_-, w)\lambda(\delta(q_-, w), a)\rho(\delta(\delta(q_-, w), a))$. The last two factors are dependent on w only through $\delta(q_-, w)$ and so can only take on $|Q|$ distinct values. Combining these observations we have that $|\Sigma|^k \leq |Q|^2$. This is false when $k > 2 \log_{|\Sigma|} |Q|$. □

Lemma 4.4 *If* $|\Sigma| \geq 2$ *then* $\mathcal{B} \in \mathbf{KerSSeqF} - (\mathbf{LBRatEq} \cup \mathbf{RecEq})$.

Proof: After reading the first character, a subsequential transducer can either stop or begin copying its input depending on whether an a or a b was seen. Thus $\mathcal{B} \in \mathbf{KerSSeqF}$.

However, $\mathcal{B}$ cannot belong to **LBRatEq** since words beginning with a can differ arbitrarily in length. It cannot belong to **RecEq** since there are an infinite number of classes corresponding to the words beginning with b. □

Lemma 4.5 $\mathcal{C} \in \mathbf{DetRatEq} - (\mathbf{LBRatEq} \cup \mathbf{KerSSeqF})$.

Proof: To show that $\mathcal{C}$ is deterministic, consider a deterministic 2-tape automaton that repeatedly reads one character from each tape and keeps track of the parity (even or odd) of the lengths. When an end-marker is detected on one of the tapes we know the parity of the terminated string and can either check for termination or even parity of the other string as appropriate.

Formally, let

$$\mathbf{A} = \langle \Sigma, \Sigma, \{A, B, C, D, E\}, \{A', B', C', D', E'\}, A, \{E\}, \delta_1, \delta_2 \rangle$$

$$\begin{array}{llll}
\delta_1(A, x) = A' & \delta_2(A', x) = B & & \\
\delta_1(A, \dashv) = C' & \delta_2(A', \dashv) = C & & \\
\delta_1(B, x) = B' & \delta_2(B', x) = A & & \\
\delta_1(B, \dashv) = E' & \delta_2(C', x) = D' & & \forall x \in \Sigma \\
\delta_1(C, x) = D & \delta_2(C', \dashv) = E & & \\
\delta_1(D, x) = C & \delta_2(D', x) = C' & & \\
\delta_1(D, \dashv) = E & \delta_2(E', \dashv) = E & &
\end{array}$$

It is easily shown that this automaton behaves as described and thus recognizes C.

However C does not have a subsequential canonical function even if $|\Sigma| = 1$. Otherwise suppose $C = f \circ f^{(-1)}$ where $\mathbf{S} = \langle \Sigma, \Delta, Q, q_-, \delta, \lambda, \rho \rangle$ and $f = |\mathbf{S}|$. Let $k_1 = |f(\Lambda)|$ and $k_2 = \max_{q \in Q} |\rho(q)|$. Since R is of infinite index, the range of f is infinite so that there must be a word w satisfying $|f(w)| > k_1 + k_2$. Thus $|w|$ is odd since otherwise $|f(w)| = |f(\Lambda)| = k_1$. Let $\delta(q_-, w) = q_1$. Then $|\lambda(q_-, w)\rho(q_1)| > k_1 + k_2$ but $|\lambda(q_-, w)\lambda(q_1, a)\rho(\delta(q_1, a))| = k_1$ so that $|\lambda(q_-, w)| \leq k_1$. But then $|\rho(q_1)| > k_2$, a contradiction.

The relation C cannot be length-bounded for any k since one of the words a^{k+1} or a^{k+2} is of even length and thus in the same equivalence class as Λ. □

Lemma 4.6 $D \in \mathbf{KerRatF} - \mathbf{DetRatEq}$.

Proof: Let a be a letter from Σ. Then the set $D = (aa)^*$ of words that have an even number of a's is a regular cross-section for D. Thus $D \cap (\Sigma^* \times D)$ is a rational canonical function for D and $D \in \mathbf{KerRatF}$.

However, even if $|\Sigma| = 1$, D is not deterministically recognizable. Otherwise there is a deterministic 2-tape automaton accepting D:

$$\mathbf{A} = \langle \Sigma, \Sigma, Q_1, Q_2, q_-, Q_+, \delta_1, \delta_2 \rangle.$$

Since $Q = Q_1 \cup Q_2$ is finite there must exist i, j, i', and j' with $i \neq i'$ or $j \neq j'$ such that $\mathbf{A}$ is in the same state, say q, after reading (a^i, a^j) and $(a^{i'}, a^{j'})$. But then if $\mathbf{A}$ accepts (a^{i+m}, a^{j+n}), it must also accept $(a^{i'+m}, a^{j'+n})$ since there is a path from q to a final state with label $(a^m \dashv, a^n \dashv)$.

Let n be any number satisfying $n > \max\{0, i/2 - j, i - i' - 2j + j'\}$ and $n \not\equiv j \pmod 2$ and define $m = 2j + 2n - i$. Then it follows that (1) $m, n \geq 0$, (2) $i + m = 2(j + n)$ where $j + n$ is odd, and (3) $i' + m > j' + n$. Thus $(a^{i+m}, a^{j+n}) \in D$ and therefore $(a^{i'+m}, a^{j'+n}) \in D$ leading to the constraint $2j - i = 2j' - i'$. Alternatively we can set m to be any number satisfying $m > \max\{0, j/2 - i, j - j' - 2i + i'\}$ and $m \not\equiv i \pmod 2$ and define $n = 2i + 2m - j$. This leads to the constraint $2i - j = 2i' - j'$. The only way that both constraints can be satisfied is if $i = i'$ and $j = j'$. □

5 Canonical Function Computation

One of the principal motivations for studying rational equivalence relations was to discover more general models for string similarity which partition the set of strings over some alphabet into disjoint classes. Implicit in this is the requirement that for any given set of strings, the partitioning can be done efficiently. If there is an efficiently computable canonical function then

the partitioning can be achieved by computing the canonical form for each string and sorting the set by the canonical form value.

To be more specific the problem then is: Given a description of an equivalence relation, construct an efficient algorithm which accepts as input a word and writes a canonical form for the class to which the word belongs. The discussion of complexity will be restricted to that of the algorithm found and not consider the cost of of finding it. Furthermore the algorithm cost will be in terms of the input length only.

Theorem 5.1 *The cost of canonical function computation on an input of length n for the subclasses of* **RatEq** *can be summarized in the following table:*

Class	*Time*	*Space*
RatEq	$O(n^2)$	$O(n^2)$
RatEq	$O(n^3/\log n)$	$O(n)$
KerRatF	$O(n)$	$O(n)$
DetRatEq	$O(n)$	$O(n)$
LBRatEq	$O(n)$	$O(n)$
KerSSeqF	$O(n)$	$O(1)$
RecEq	$O(n)$	$O(1)$
FinEq	$O(n)$	$O(1)$

When restricted to the one letter case **RatEq** *has a canonical function computable in* $O(n)$ *time and space and* **LBRatEq** *has a canonical function computable in* $O(n)$ *time and* $O(1)$ *space.*

Because the proofs in section 3 were all effective reductions, we can prove theorem 5.1 by establishing the complexity bounds in the three cases: **RatEq**, **KerRatF**, and **KerSSeqF**.

Lemma 5.2 *For any* $R \in$ **RatEq** *there is a canonical function* f *computable in* $O(n^3/\log n)$ *time and* $O(n)$ *space. Alternatively, it may be computed in* $O(n^2)$ *time and space.*

Proof: Consider an enumeration of Σ^* that lists Λ, then strings of length one, strings of length two, and so on. Within each set of strings of a given length, we choose a reversed lexicographic ordering. For example, if our alphabet is $\{a, b, c\}$ and we choose to have $a \prec b \prec c$ then our enumeration will be:

$$\Lambda, a, b, c, aa, ba, ca, ab, bb, cb, ac, bc, cc, aaa, baa, \ldots$$

If we are now given a string u we will find the first element v in sequence such that $(u, v) \in R$. This is definitely a bounded search since we never

need to search past u. This exponential time algorithm provides a canonical function for R.

We can reduce the cost of finding the canonical element using a modification of the membership algorithm for rational relations. Assume that we are given a quasi-deterministic 2-tape automaton for a rational equivalence relation R: $\mathbf{A} = \langle \Sigma, \Sigma, Q, q_-, Q_+, \delta_1, \delta_2 \rangle$. Consider the modified automaton $\mathbf{A}' = \langle \Sigma, \Sigma \cup \{?\}, Q, q_-, Q_+, \delta_1, \delta_2' \rangle$ where ? is a new letter not in Σ and

$$\delta_2'(q,x) = \begin{cases} \delta_2(q,x) & \text{if } x \in \Sigma \\ \bigcup_{y \in \Sigma} \delta_2(q,y) & \text{if } x = ? \end{cases}$$

Thus δ_2' is the same as δ_2 except that it can read a wild card character instead of any other letter.

We can use $\mathbf{A}'$ to identify the canonical string. We first find its length by testing (u, Λ), $(u, ?)$, $(u, ??)$, and so on for membership. When we find a string of wild card characters that is related to u, we then know the length of the output word.

The second part of the algorithm attempts to replace each wild card character by a letter from Σ proceeding from right to left. Let $\Sigma = \{a_i | 1 \leq i \leq s\}$ and suppose our given ordering is $a_1 \prec a_2 \prec a_3 \prec \cdots \prec a_s$. First we replace the last wild card by a_1 and test the new string for membership. If this fails, we try a_2 and so on until we succeed. Then we move on to the second last wild card and repeat the process. When we finish the first wild card, we will have a string in Σ^* and it will be minimal according to the above ordering. This algorithm requires $O(n)$ space and $O(n^3/\log n)$ time since it requires a maximum of $n+|\Sigma|n$ passes of the standard membership algorithm each of which require $O(n^2/\log n)$ time and $O(n)$ space [MP80,vLN82].

It should be clear that we could have assigned wild cards from left to right instead of from right to left. The advantage with the above approach is that we can avoid many repeated computations as done in the algorithm in Figure 3. This is a modification of the standard dynamic programming algorithm for testing whether a pair of words are accepted by a non-deterministic 2-tape automaton. The basic data structure used is an array T with rows indexed by positions in the first word and columns indexed by positions in the second word, here being computed. The first "for" statement (at (A) in Figure 3) initializes column zero of T so that $T[i,0]$ contains all states reachable from q_- with a label $(u_1 u_2 \ldots u_i, \Lambda)$. The "while" statement at (B) continues adding columns until one is found whose last element contains a final state. Each column is added by extending the output word with a wild card character and determining the states reachable with label as the appropriate prefix of u and the correct number of wild cards. The length of the output is known and assigned to the variable L. The loop at (C) then prepares for a backward pass of the array T by removing from the last

{ Given a modified automaton $\mathbf{A}' = \langle \Sigma, \Sigma, Q, q_-, Q_+, \delta_1, \delta_2 \rangle$ and an input word u, compute the first word v in reversed lexicographic sequence such that $(u, v) \in |\mathbf{A}'|$. Let $\eta_i(k, x) = \{q | \delta_i(q, x) = k\}$ for $i = 1, 2$ }
T: **array** $[0 : |u|, 0 : |u|]$ **of set of** *states*;
W: **array** $[0 : |u|]$ **of set of** *states*;
begin $T[0,0] := \{q_-\}$;
(A) **for** $i := 1$ **to** $|u|$ **do** $T[i, 0] := \bigcup_{k \in T[i-1,0]} \delta_1(k, u_i)$;
$j := 0$;
(B) **while** $T[|u|, j] \cap Q_+ = \emptyset$ **do begin**
$j := j + 1$; $T[0, j] := \bigcup_{k \in T[0,j-1]} \delta_2(k, ?)$;
for $i := 1$ **to** $|u|$ **do**
$T[i, j] := \bigcup_{k \in T[i-1,j]} \delta_1(k, u_i) \cup \bigcup_{k \in T[i,j-1]} \delta_2(k, ?)$
end;
$L := j$; $T[|u|, L] := T[|u|, L] \cap Q_+$;
(C) **for** $i := |u| - 1$ **to** 0 **by** -1 **do**
$T[i, L] := T[i, L] \cap \bigcup_{k \in T[i+1,L]} \eta_1(k, u_{i+1})$;
(D) **for** $j := L - 1$ **to** 0 **by** -1 **do begin**
(E) **for** $l := 1$ **to** $|\Sigma|$ **until** $\bigcup_{i=0}^{|u|} W[i] \neq \emptyset$ **do begin**
$v_{j+1} := a_l$; $W[|u|] := T[|u|, j] \cap \bigcup_{k \in T[|u|,j+1]} \eta_2(k, v_{j+1})$;
for $i := |u| - 1$ **to** 0 **by** -1 **do**
$W[i] := T[i, j] \cap \left\{ \bigcup_{k \in W[i+1]} \eta_1(k, u_{i+1}) \cup \bigcup_{k \in T[i,j+1]} \eta_2(k, v_{j+1}) \right\}$
end;
for $i := 0$ **to** $|u|$ **do** $T[i, j] := W[i]$
end end

Figure 3: A canonical function for a rational equivalence relation

column all states which are not on a path to a final state. The loop at (D) continues the process by weeding out from T the states which cannot be on a successful path. The decision about the actual characters of v are made in the loop (E). Each character in Σ is tried in turn until one is found which does cause the column under consideration to be completely zeroed out and thus break all paths. This algorithm then requires $O(n^2)$ time and $O(n^2)$ space since each column of the table will be computed once in the first part of the algorithm and modified a maximum of $|\Sigma|$ times in the second part. □

The above function is not in general rational. For example, consider the relation $R = T \cup T^{(-1)} \cup \iota_U$ where $T = \{(a^{2i}b^{4j}, b^{4i}a^{2j+1}) | i, j \geq 0\}$. The above algorithm would select from each class the shorter of the two words. This is shown in [Joh85] not to be a regular set. As a result, it cannot be

the range of a rational function.

Lemma 5.3 *If $R \in$ **KerRatF** then there is a canonical function that requires $O(n)$ time and space for input of size n. Furthermore this function can be effectively computed from a finite transducer for R.*

Proof: Clearly if $R \in$ **KerRat** then there is a rational function f that is a canonical function. Thus we need to demonstrate that such a rational function can be found and that it can be used to construct a $O(n)$ time and space bounded canonical function. It is shown in [Joh85] that any $R \in$ **KerRatF** has a regular cross-section. Thus we can effectively find a rational function for R by enumerating regular languages and testing whether restricting the domain of R to each of these yields a function. Clearly such a function will be rational and will exist because of the existence of a regular cross-section.

Now any rational function can be expressed as the composition of a left sequential and a right sequential transduction [Ber79, Theorem 5.2 pp. 126–127]. Since a sequential transduction cannot generate an output whose length exceeds its input length by more than a constant factor, the time and space required will be $O(n)$. □

Lemma 5.4 *If $R \in$ **KerSSeqF** then there is a canonical function that requires $O(n)$ time and $O(1)$ space. Furthermore this function can be effectively computed from a deterministic finite automaton for R.*

Proof: The only information that needs to be stored is the current state and the next symbol of input. For each character read we will expend an effort proportional to the number of characters written. This is bounded by a constant so that the overall cost is linear in the size of the input.

We can effectively find such a transducer if it exists. Without loss of generality we can assume that the image alphabet has two letters. We simply enumerate transducers from Σ^* into $\{a,b\}^*$ until we find one f satisfying $f \circ f^{(-1)} = R$. This latter question is known to be decidable for deterministic rational relations [Bir73] and as a result of the reductions of section 3 we know that both sides of the equality are deterministic rational. □

6 Decision Problems

If rational equivalence relations are to be useful, it must be possible to construct them. One approach would be to start with a rational relation specified as a finite transducer and modify it until it is an equivalence relation. This section shows that this is very likely not to be a feasible approach since it is recursively undecidable whether the behaviour of a given transducer is an equivalence relation.

Theorem 6.1 *Given a finite transducer for a rational relation R, the following properties are recursively undecidable:*

1. *R is transitive (i.e., $R \circ R \subseteq R$)*
2. *R is symmetric (i.e., $R \subseteq R^{(-1)}$)*
3. *R is reflexive (i.e., $\iota_{\mathrm{dom}R} \subseteq R$)*
4. *$R \in$* **RatEq**
5. *$R \in$* **KerRatF**
6. *$R \in$* **DetRatEq**
7. *$R \in$* **KerSSeqF**
8. *$R \in$* **RecEq**

The following properties are decidable:

1. *$R \in$* **LBRatEq**
2. *$R \in$* **FinEq**
3. *$R \in$* **RecEq** *when R is known to be an equivalence relation.*

If the alphabet is restricted to contain one letter then the above undecidable properties become decidable except for $R \in$ **DetRatEq** *which remains open.*

This will be proven by exhibiting a rational relation which is either the universal relation or not an equivalence relation depending of whether an instance of Post's correspondence problem has a solution. Since the universal relation is in **RecEq**, by the hierarchy of theorem 3.1 it is undecidable whether a given rational relation is in any of the classes **RatEq**, **KerRatF**, **DetRatEq**, **KerSSeqF**, or **RecEq**. Since the relation can be made to fail any of the reflexive, symmetric, or transitive laws, these are undecidable properties. These arguments depend critically on Σ having at least two letters. All of these questions become decidable in the one letter case using the decidability of inclusion.

The following three lemmas will follow a model of Fischer and Rosenberg [FR68] using a reduction from Post's Correspondence Problem: Given sequences $u_1, u_2, \ldots, u_p$ and $v_1, v_2, \ldots, v_p$ of strings, determine a non-empty sequence of indices $i_1, i_2, \ldots, i_k$ such that $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$. The existence of a solution to PCP is a well-known recursively undecidable problem.

Lemma 6.2 *Transitivity testing for rational relations is undecidable.*

Proof: Let a and b be in Σ and $u_1, u_2, \ldots, u_p, v_1, v_2, \ldots, v_p \in \Sigma^*$ be an instance of PCP. We can construct a relation that is transitive if and only if this instance of PCP has no solution. Define

$$U = \{(ab, u_1), (a^2b, u_2), \ldots, (a^pb, u_p)\}$$

$$V = \{(ab, v_1), (a^2b, v_2), \ldots, (a^pb, v_p)\}.$$

The relation U^+ is subsequential since we can construct a subsequential transducer that counts a's and whenever it reads a b, emits the correct u_i. Thus U^+ is a deterministic rational relation. Similarly V^+ is a deterministic rational relation. The relations $(\Sigma^* \times \Sigma^*) - U^+$ and $(\Sigma^* \times \Sigma^*) - V^+$ are deterministic rational since deterministic rational relations are closed under complementation. Since any deterministic rational relation is rational

$$Z = \{(\Sigma^* \times \Sigma^*) - U^+\} \cup \{(\Sigma^* \times \Sigma^*) - V^+\} = (\Sigma^* \times \Sigma^*) - (U^+ \cap V^+)$$

is rational. Now $U^+ \cap V^+$ is not empty if and only if the corresponding instance of PCP has a solution. If $U^+ \cap V^+$ is empty then $Z = \Sigma^* \times \Sigma^*$ and is thus transitive. If $U^+ \cap V^+$ is not empty, it will be a function that is not total. Thus if $(w_1, w_2) \in U^+ \cap V^+$ then $z \neq w_2$ implies that $(w_1, z) \notin U^+ \cap V^+$. It is possible to choose $z \neq w_2$ such that $z \notin \mathrm{dom}(U^+ \cap V^+)$ since there are at least two words (and therefore at least one different from w_2) in $\Sigma^* - \mathrm{dom}(U^+ \cap V^+)$ (for example, aa and aaa). Now $(w_1, w_2) \notin Z$ but $(w_1, z), (z, w_2) \in Z$. Thus Z is not transitive. Since Z is transitive iff the given instance of PCP does not have a solution, transitivity is an undecidable property. □

Lemma 6.3 *Symmetry testing for rational relations is undecidable.*

Proof: Consider $Z = (\Sigma^* \times \Sigma^*) - (U^+ \cap V^+)$ with Σ, U, and V defined as above. Since Z is a rational relation, $(A \cdot Z) \cup B$ where $A = \Sigma \times \Lambda$ and $B = \Lambda \times \Sigma^*$ is a rational relation.

Now if $U^+ \cap V^+$ is empty, then $(A \cdot Z) \cup B$ is $\Sigma^* \times \Sigma^*$ and thus symmetric. If $U^+ \cap V^+$ is not empty, $A \cdot (U^+ \cap V^+)$ is a function but its inverse is not. Thus $(A \cdot Z) \cup B = (\Sigma^* \times \Sigma^*) - (A \cdot (U^+ \cap V^+))$ is not symmetric. □

Lemma 6.4 *Reflexivity testing for rational relations is undecidable.*

Proof: Let $\Delta = \{a, b, \$\} = \Sigma \cup \{\$\}$ and let $Z = (\Sigma^* \times \Sigma^*) - (U^+ \cap V^+)$ as above and $A = \Lambda \times (\Sigma^* \cdot \{\$\})$ and $B = (\{\$\} \cdot \Sigma^*) \times \Lambda$. Thus $A \cdot Z \cdot B$ has as domain $\{w_1\$w_2 | w_1, w_2 \in \Sigma^*\}$. But $(w_1\$w_2, w_1\$w_2) \in A \cdot Z \cdot B$ if and only if $(w_1, w_2) \notin U^+ \cap V^+$ since by construction $(w_1\$w_2, w_1\$w_2)$ must factor into $(\Lambda, w_1\$) \in A$, $(w_1, w_2) \in Z$, and $(\$w_2, \Lambda) \in B$. Thus $A \cdot Z \cdot B$ is reflexive if and only if the given instance of PCP is empty. □

Note that we can extend the domain of $A \cdot Z \cdot B$ to Δ^* by constructing

$$Z' = A \cdot Z \cdot B \cup (\Sigma^* \cup (\Sigma^*\$\Sigma^*(\$\Sigma^*)^+)) \times (\Sigma^* \cup (\Sigma^*\$\Sigma^*(\$\Sigma^*)^+))$$

without affecting the proof. In addition, we could code Δ back into Σ, so that two letters suffice.

Corollary 6.5 *It is undecidable whether a rational relation is in* **RecEq**.

Proof: The above lemmas construct a rational relation which is either a recognizable equivalence relation ($\Sigma^* \times \Sigma^*$) or not an equivalence relation depending on whether an instance of PCP has a solution. Thus membership in **RecEq** is undecidable. □

Lemma 6.6 *It is decidable whether a rational relation is in* **LBRatEq**.

Proof: We can check whether it is length-bounded by inspecting the transducer for loops that are not length-preserving and yet can occur on a successful path. If there is one such loop the transduction cannot be length-bounded since we can pump up one of the tapes arbitrarily and exceed any predefined bound. If there is no loop then the transduction is length-bounded following [EM65]. If the relation is length-bounded then it is deterministic. Symmetry can be tested using Bird's algorithm [Bir73] to see whether R equals $R^{(-1)}$. Transitivity can be tested by computing $R \circ R$ which is length-bounded if R is and testing whether this relation is equal to R. It is easily shown that $R = R^{(-1)}$ and $R = R \circ R$ if and only if R is an equivalence relation on its domain. □

Lemma 6.7 *It is decidable whether a rational relation is in* **FinEq**.

Proof: We can easily check whether the relation is finite by checking the domain and range for finiteness [Ber79, Proposition 8.2 p. 88] If they are finite then the finite set of ordered pairs in the relation can be enumerated and the reflexivity, symmetry and transitivity verified. □

Lemma 6.8 *It is decidable whether a rational relation is an equivalence relation in the one-letter-alphabet case.*

Proof: Inclusion testing for rational relations over a one letter alphabet is decidable. Thus testing of reflexivity, symmetry, and transitivity are all decidable. □

Lemma 6.9 *Given a rational equivalence relation R it is decidable whether* $R \in$ **RecEq**.

Proof: The idea is to compute a locally finite thinning of R as shown in [Joh85] and test whether the resulting relation is finite. If it is then the original relation must have been of finite index and therefore in **RecEq**. □

Lemma 6.10 *It is decidable whether a rational relation is in either of the classes* **RecEq** *or* **KerSSeqF** *in the one-letter-alphabet case.*

Proof: From above results we can decide whether a relation is in **LBRatEq** or in **RecEq** and therefore whether it is in **KerSSeqF** (since it the union of these two classes in the one letter case). □

There remains exactly one case and it is open. If $R \in$ **KerRatF** is it decidable whether $R \in$ **DetRatEq**?

7 Conclusions and Open Problems

A hierarchy of rational equivalence relations has been established. This provides a framework for identifying efficient canonical functions and a number of decidability questions. Although it is undecidable whether a given rational relation is an equivalence relation, canonical functions seem to be efficiently computable, at least in terms of the input length.

The most significant remaining open problem is whether the containment **KerRatF** $\subseteq$ **RatEq** is proper or alternatively that these classes are the same [Joh85]. There are several relative decidability questions that remain open.

Given	Membership Decision Problem		
$R \in$ **RatEq**	$R \in$ **KerRatF**?	$R \in$ **DetRatEq**?	$R \in$ **KerSSeqF**?
$R \in$ **KerRatF**	$R \in$ **DetRatEq**?	$R \in$ **KerSSeqF**?	
$R \in$ **DetRatEq**	$R \in$ **KerSSeqF**?		
$R \in$ **LBRatEq**	$R \in$ **KerSSeqF**?		
$R \in$ **DetRat**	$R \in$ **DetRatEq**?	$R \in$ **KerSSeqF**?	$R \in$ **RecEq**?

With regard to the question of whether $R \in$ **KerSSeqF** for $R \in$ **KerRatF**, it is known to be decidable whether a rational function is subsequential [Cho77]. However, this question is stronger: For the given rational function is there a subsequential function whose equivalence kernel is the same?

Another type of open problem is that of effectively characterizing the classes **RatEq** and **DetRatEq**, that is coming up with a presentation for these two classes such that any member of the class can be represented by the model and it is decidable whether the presentation is well formed. For example, a rational equivalence relation can be described by a rational transducer and an assurance that it describes an equivalence relation. Since

the assurance cannot be verified effectively, this is not an effective characterization. On the other hand, a member of **KerRatF** can be presented in terms of a rational function in the form of a transducer. It is possible to verify that this transducer describes a function.

Another open problem related to the effective characterization for **DetRatEq** is the decidability of the question whether $R \circ R = R$ when R is deterministic rational. If this is decidable then the class of **DetRatEq** can be effectively characterized by deterministic 2-tape finite automata.

8 Acknowledgements

I wish to thank Karel Culik II for his help at the early stages of this research and especially for pointing the way in the undecidability proofs of section 6. I also thank Christian Choffrut for some useful discussions leading to the resolution of the one letter alphabet case and Derick Wood and a referee for comments on the presentation of the material.

References

[Ber79] Jean Berstel. *Transductions and Context-Free Languages.* B. G. Teubner, Stuttgart, Germany, 1979.

[BH77] Meera Blattner and Tom Head. Single valued a-transducers. *Journal of Computer and System Sciences*, 15:310–327, 1977.

[Bir73] Malcolm Bird. The equivalence problem for deterministic two-tape automata. *Journal of Computer and System Sciences*, 7:218–236, 1973.

[Cho77] Christian Choffrut. Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles. *Theoretical Computer Science*, 5:325–338, 1977.

[Dav62] Leon Davidson. Retrieval of misspelled names in an airlines passenger record system. *Communications of the ACM*, 5(3):169–171, 1962.

[Eil74] Samuel Eilenberg. *Automata, Languages, and Machines, vol. A.* Academic Press, New York, 1974.

[EM65] C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM Journal of Research*, 9:47–65, 1965.

[FR68] Patrick C. Fischer and Arnold L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences*, 2:88–101, 1968.

[FS69] Ivan P. Fellegi and Alan B. Sunter. A theory of record linkage. *Journal of the Americal Statistical Association*, 64:1183–1210, 1969.

[Gin66] Seymour Ginsburg. *The Mathematical Theory of Context-Free Languages.* McGraw-Hill, New York, 1966.

[Joh83] J. Howard Johnson. *Formal Models for String Similarity.* PhD thesis, University of Waterloo, 1983. Available as University of Waterloo Research Report CS–83–32.

[Joh85] J. Howard Johnson. Do rational equivalence relations have regular cross-sections? In *Proceedings of the 12th International Conference on Automata, Languages, and Programming*, pages 300–309, Springer-Verlag LNCS 194, 1985.

[Knu73] Donald E. Knuth. *Sorting and Searching.* Addison-Wesley, Reading, Mass., 1973.

[MKTM77] Gwendolyn B. Moore, John L. Kuhns, Jeffrey L. Trefftzs, and Christine A. Montgomery. *Accessing Individual Records from Personal Data Files using Non-unique Identifiers.* Technical Report NBS Special Publication 500–2, U.S. Dept. of Commerce—National Bureau of Standards, 1977. Available from the National Technical Information Service.

[MP80] William J. Masek and Michael S. Paterson. A faster algorithm for computing string-edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.

[Niv68] Maurice Nivat. Transductions des langages de Chomsky. *Ann. de l'Inst. Fourier*, 18:339–456, 1968.

[NK62] H. B. Newcombe and J. M. Kennedy. Record linkage: making maximum use of the discriminating power of identifying information. *Communications of the ACM*, 5(11):563–566, 1962.

[Sch61] M. P. Schützenberger. A remark on finite transducers. *Information and Control*, 4:185–196, 1961.

[Sch75] M. P. Schützenberger. Sur les relations rationelles. In *Automata theory and formal languages: 2nd GI Conference*, pages 209–213, 1975.

[vLN82] Jan van Leeuwen and Maurice Nivat. Efficient recognition of rational relations. *Information Processing Letters*, 14(1):34–38, 1982.

[Wie77] Gio Wiederhold. *Database Design.* McGraw-Hill, New York, 1977.