Basic Research in Computer Science

BRICS RS-94-11

A Homomorphism Concept for ω-Regularity

Nils Klarlund

BRICS Report Series **RS-94-11**

ISSN 0909-0878 **May 1994**

See back inner page for a list of recent publications in the BRICS Report Series. Copies may be obtained by contacting:

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark**

**Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through WWW and anonymous FTP:

`http://www.brics.dk/`
`ftp ftp.brics.dk (cd pub/BRICS)`

A HOMOMORPHISM CONCEPT FOR ω-REGULARITY

NILS KLARLUND*
BRICS †
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF AARHUS
NY MUNKEGADE
DK-8000 AARHUS C, DENMARK.

Abstract. The Myhill-Nerode Theorem (that for any regular language, there is a canonical recognizing device) is of paramount importance for the computational handling of many formalisms about finite words.

For infinite words, no prior concept of homomorphism or structural comparison seems to have generalized the Myhill-Nerode Theorem in the sense that the concept is both language preserving and representable by automata.

In this paper, we propose such a concept based on Families of Right Congruences [3], which we view as a recognizing structures.

We also establish an exponential lower and upper bound on the change in size when a representation is reduced to its canonical form.

1. Overview

An important and only partially solved problem in the theory of ω-regular languages is whether representations can be minimized. For usual regular languages, deterministic finite-state automata (DFAs) are recognizing structures that can be minimized easily in polynomial time by virtue of the Myhill-Nerode Theorem. The lack of similar algorithms in the ω-case is a major impediment to building verification tools for concurrent programs.

The syntactic congruences of Arnold [2] provide canonical algebraic structures for ω-regular languages. By themselves, these congruences provide no explicit acceptance criteria just as in the situation for a regular language: the canonical right congruence, whose classes are automata states, does not define a language—unless certain states are designated as being final. Similarly, Arnold's congruences have only the *ability* to recognize, which is a property called *saturation*. Arnold's congruences can

*The author was partially supported by a Fellowship from the Danish Research Council.
†Basic Research in Computer Science, Centre of the Danish National Research Foundation.

be extended so that acceptance becomes explicit and thus a language preserving homomorphism concept arises. But, unlike the Myhill-Nerode Theorem, which is based on right congruences, canonicity in [2] is obtained for full congruences, which are usually exponentially bigger than one-sided congruences.

Maler and Staiger [3] focus on the *canonical right congruence* $\equiv_L$ on finite words of a language L of infinite words. This congruence is defined by $x \equiv_L y$ if and only if for all infinite α, $x \cdot \alpha \in L$ if and only if $y \cdot \alpha \in L$. (We use x, y, u, v, w to denote finite words and α, β to denote infinite words). The concept of a *Family of Right Congruences* (*FORC*) suggested in [3] is there used to characterize ω-regular languages that are accepted by their canonical right congruence $\equiv_L$ extended to a Muller automaton.

FORCs are also not language recognizing. But they do enjoy canonical properties with respect to saturation as we prove in this paper. Similarly, the *right binoids* of Wilke [4] are algebraic devices that characterize regular sets of finite and infinite words based on a saturation concept embedded in a notion of recognition by homomorphism.

In this paper. In this paper we regard FORCS as language accepting devices rather than as the transition structures of underlying Muller automata. Then FORCS may be viewed as separating the characterization of the *topological closure* of the language from that of the *dense part*.

The closure corresponds to the canonical right congruence. The classes of this relation for which there is an infinite suffix that makes words in the class belong to L describe the closure of L: an infinite word is in the closure if and only if all of its prefixes belong to these classes. The closure is also called a *safety property* in the theory of concurrent systems. A FORC represents the closure by what we here call a *safety congruence*, which is a refinement of the natural right congruence. (The results of [3] shows under which conditions this safety congruence may be used with a Muller condition to accept languages that are not necessarily closed.)

The dense part of L is described by a collection of right congruences, here called *progress congruences*, that specify the cyclic behavior that any word eventually exhibits according to Ramsey's Theorem about finite partitions of the natural numbers. Thus it is natural to view these congruences as an algebraic formalization of progress towards the dense part, known as a *liveness property* in concurrency [1] .

We show that a Myhill-Nerode Theorem exists that declares a unique minimum representation of an ω-regular language under a structural comparison that is language preserving. Also, we clarify the notion of refinement of FORCs presented in [3].

Our representation is that of a FORC extended by explicit enumeration of accepting progress states. We call such a device an *LFORC*, since it is Language accepting. Under the automata-theoretic view, an LFORC is a *Family of DFAs* (*FDFA*).

We introduce a concept of *retraction* between LFORCs and show that it is language

preserving. We also formulate a retraction under the automata-theoretic view as an *FDFA homomorphism*. From a given FDFA, the homomorphism involves implicitly formed product state spaces that may be exponentially larger than the FDFA itself.

Our main result is that among all LFORCs recognizing a language L there is a *canonical* or *minimum* one. Thus all such LFORCs retract to this minimum LFORC.

The canonical LFORC was already defined, as a FORC, in [3]. It was reported there that with respect to saturation this FORC is canonical for a straightforward notion of refinement. This result, however, does hold only in certain situations. We provide a simple counter-example for the general case.

The primary consequence of our generalization of the Myhill-Nerode Theorem is that minimization of ω-regular representations is reducible to calculations involving only regularity or usual finite-state automata. We show how any FDFA can be retracted to the minimum FDFA by structural operations that do not refer to acceptance of infinite words.

The minimization of FDFAs may yield an exponential blow-up in size. We establish both the lower and the upper bound. This blow-up can occur only for the progress congruences, whose number of equivalence classes may grow exponentially. The safety congruence, however, can only shrink.

We also show that a kind of inverse refinement holds for the progress congruences: for any FORC, every progress congruence that is minimized with respect to the safety congruence is refined by the product of the safety congruence and the canonical progress congruence. Thus during minimization, the progress congruences become less coarse whereas the safety congruence becomes coarser.

Applications to minimization. From [3], it follows that there are polynomial translations from FORCs to deterministic Rabin or Streett automata (with a number of acceptance pairs that is roughly logarithmic in the state space size). This is unlike the situation for Arnold's congruence that may be exponentially bigger than its automaton representation.

But if minimization is involved, there need not be an exponential gain in using LFORCs instead of Arnold's syntactic congruences, since during minimization LFORCs may blow up whereas Arnold's congruences can only shrink, i.e. become coarser. It appears though that FORCs grow drastically in size only if the progress part is more involved than the safety part.

In practice, the liveness part is usually quite simple, so the algebraic framework we suggest here might, despite the difficult calculations involved, make it possible to do theorem proving for simple temporal properties by automata-theoretic methods.

2. FORCs and LFORCs

Let Σ be a finite or infinite *alphabet*. The empty word is denoted ϵ. The set of finite words is denoted Σ^* and the set of infinite words is denoted Σ^ω. A *right congruence*

$\sim$ on Σ^* is an equivalence relation that satisfies

$$x \sim y \text{ implies for all } a, xa \sim ya.$$

Then each $u \in \Sigma^*$ defines an operation of right concatenation on any equivalence class s by $su = s'$, where s' is defined as $[xu]$ with x any member of s.

A FORC $\mathfrak{F} = (\sim, \mathrel{\mathring{\sim}})$ consists of right congruences of finite index on Σ^*. We call the relation $\sim$ the *safety congruence*. An equivalence class s is also called a *safety state*. The safety state of $u \in \Sigma$, i.e. the s such that $u \in s$, is denoted $[u]$. To each safety state s is associated the right congruence $\mathrel{\mathring{\sim}}_s$, called the *progress congruence* of s. An equivalence class p of $\sim_s$ is called a *progress state*. The progress state of u with respect to $\mathrel{\mathring{\sim}}_s$ is denoted $[u]_s$. The following requirement must hold:

(FORC) $$x \mathrel{\mathring{\sim}}_s y \text{ implies } sx = sy.$$

A non-empty word x such that $s = s \cdot x$ is called *s-cyclic*.

By (FORC), an operation of right-concatenating a progress state p of $\sim_s$ to s is defined by

$$s \cdot p = s \cdot x,$$

where x is chosen so that $x \in p$. A progress state p such that $s \cdot p = s$ is called *cyclic*. Thus the progress state according to $\mathrel{\mathring{\sim}}_s$ of an s-cyclic word is cyclic.

An *(s,p)-factorization* of a word $\alpha \in \Sigma^\omega$, where s is a safety state and p is a progress state of $\mathrel{\mathring{\sim}}_s$, is a collection $v_0, v_1, v_2, \ldots$ of non-empty *factors* such that $\alpha = v_0 v_1 v_2 \ldots$ and for all $i > 0$, $v_0 \cdots v_i \in s = sv_i$ and $v_i \in p$. If in addition, $p = pv_i$ (for all $i > 0$), then the factorization is said to be *progress cyclic*.

The following lemma summarizes results in [2] and [3].

Lemma 1. (Factorization) Given a FORC $\mathcal{F} = (\sim, \mathrel{\mathring{\sim}}_s)$.

(a) Every $\alpha \in \Sigma^\omega$ admits a cyclic (s,p)-factorization for some (s,p).

(b) Moreover, if $\alpha = xy^\omega$, then it admits some (s,p)-factorization $v_0 = xy^m$ and $v_i = y^n$, $i > 0$ for some $m, n > 0$. This factorization is also denoted

$$\alpha = \underbrace{xy^m}_{s}\underbrace{(y^n)}_{p}^{\omega}$$

(c) Every $\alpha = xy^\omega$ admitting an (s,p)-factorization has a factorization

$$\alpha = \underbrace{v_0}_{s}\underbrace{v}_{p}^{\omega}$$

(These factorizations may even by assumed progress cyclic.)

Proof. (a) Using the finiteness assumption, we can find s of $\sim$ such that infinitely many prefixes $u_0 \prec u_1 \cdots$ of α are in s. By Ramsey's Theorem and the finiteness assumption, there is a p of $\mathrel{\mathring{\sim}}_s$ and an increasing sequence $k_i, i > 0$ such that for all i and j $(i < j)$, $u_{k_j} - u_{k_i} \in p$. Define $v_0 = u_{k_1}$ and $v_i = u_{k_{i+1}} - u_{k_i}$. Then $v_i \in s$ and

$v_i \in p$. Also, since for $i > 0$, $u_{k_{i+1}} - u_{k_{i-1}} \in p$, $u_{k_i} - u_{k_{i-1}} \in p$, $u_{k_{i+1}} - u_{k_i} = v_i \in p$, and $u_{k_{i+1}} - u_{k_{i-1}} = (u_{k_i} - u_{k_{i-1}}) \cdot v_i$, we see that $p = p \cdot v_i$. Also, $s \cdot v_i = [u_{k_i}] \cdot v_i = [u_{k_i} \cdot v_i] = [u_{k_{i+1}}] = s$.

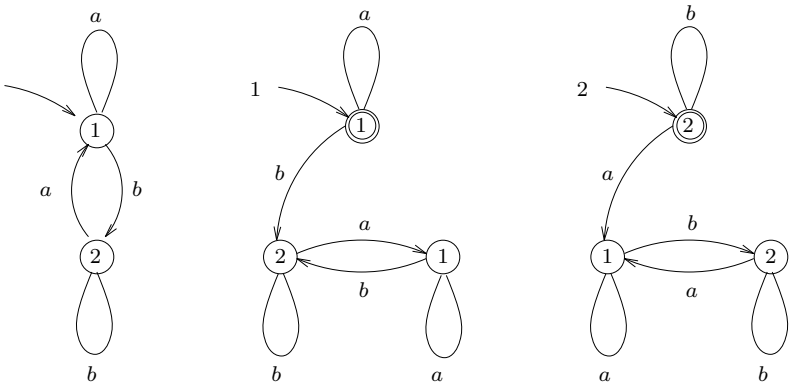
(b) We can find s such that infinitely many prefixes are in s and of the form xy^i. By (a), we obtain a factorization $v_0 = xy^m$, $v_i' = y^{n_i}$. Let $n = n_1$ and $v_i = y^n$. Then $\alpha = v_0 v^\omega$ with $v_0 \in s = sv_i$ and $v_i \in p = pv_i$.

(c) We can find an (s,p)-factorization such that $\alpha = xy^n y_1 (y_2 y^m y_1)^\omega = v_0 v^\omega$ with $y = y_1 y_2$, $v_0 = xy^n y_1$, $v = y_2 y^m y_1$, $v_0 \in s = sv$, $v \in p = pv$.

□

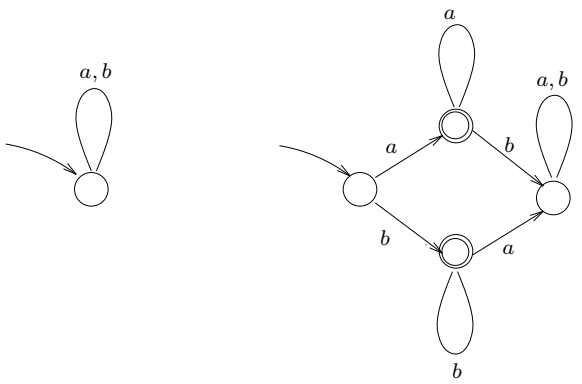
For (s,p) define $L_{(s,p)}$ to be the set of words admitting an (s,p)-factorization. A *live assignment* Λ associates to each s a subset Λ_s of progress states of $\precsim_s$. A FORC $(\sim, \precsim)$ together with a live assignment Λ is called an LFORC (for Language recognizing FORC) and denoted $\mathfrak{L} = (\sim, \precsim, \Lambda)$. The language *recognized* by $(\sim, \precsim, \Lambda)$ is the union of $L_{(s,p)}$ for $p \in \Lambda_s$ and is denoted $L(\sim, \precsim, \Lambda)$. Thus it consists of the words that allow some (s,p)-factorization with $p \in \Lambda_s$.

2.0.1. *Example.* An LFORC is perhaps best understood as a family of automata. The ω-regular language $\Sigma^*(a^\omega \cup b^\omega)$, where $\Sigma = \{a, b\}$, can be represented by an LFORC specified by three automata:



The first automaton defines the safety congruence $\sim$ as $x \sim y$ if and only if the last letter in x and in y are the same. The congruence $\precsim_1$ is specified by the second automaton. Each state is marked with the corresponding safety state according to the requirement (FORC). The states in Λ_1 are marked by an inner circle. The other progress congruence $\precsim_2$ is shown as the last automaton.

There is another LFORC representation of the same language with a simpler safety congruence and a more complicated progress congruence:



□

The *size* of an LFORC is the maximum index of its congruence relations. Thus the size of the first LFORC above is three and the size of the second one is four. (One could also have defined the size as the total number of classes, but this number is at most quadratically bigger.)

Given L and $(\sim, \stackrel{\circ}{\sim})$, define the *natural live assignment* Λ by letting Λ_s^L consist of the p such that some $\alpha \in L$ allows an (s,p)-factorization, i.e. such that $L \cap L_{(s,p)}$ is non-empty. Then $L \subseteq L(\sim, \stackrel{\circ}{\sim}, \Lambda^L)$.

A language L is *saturated* by $(\sim, \stackrel{\circ}{\sim})$ if for all α and β both admitting an (s,p)-factorization, it holds that $\alpha \in L$ if and only if $\beta \in L$ or, in other words, if for all (s,p), either $L_{(s,p)} \subseteq L$ or $L_{(s,p)} \cap L = \emptyset$. Thus for $\alpha \in L$, we may choose any factorization to determine whether $\alpha \in L$.

We can express the saturation property of [2] as follows.

Lemma 2. (Saturation)
L is recognized by $(\sim, \stackrel{\circ}{\sim}, \Lambda^L)$ if and only if L is saturated by $(\sim, \stackrel{\circ}{\sim})$.
Proof.

"⇒" Assume $L = L(\sim, \stackrel{\circ}{\sim}, \Lambda^L)$ and $L_{(s,p)} \bigcap L \neq \emptyset$. Then by definition of Λ_s^L, $p \in \Lambda_s^L$ and by definition of recognition, $L_{(s,p)} \subseteq L$.

"⇐" We just need to establish that $L(\sim, \stackrel{\circ}{\sim}, \Lambda^L) \subseteq L$. So assume that $(\sim, \stackrel{\circ}{\sim})$ saturates L and that α has an (s,p)-factorization with $p \in \Lambda_s^L$. Now $p \in \Lambda_s^L$ only since some other word β in L has an (s,p)-factorization. Thus by saturation, $\alpha \in L$.

□

3. Refinements and Retractions

We say that $\sim$ *refines* $\simeq$ if $x \sim y$ implies $x \simeq y$. Then for $\underline{s}$ an equivalence class of $\simeq$, $|\underline{s}|_\sim$ is the number of equivalence classes of $\sim$ contained in $\underline{s}$. Moreover, if s is an equivalence class of $\sim$, then $[s]_\simeq$ is the equivalence class of $\simeq$ that contains s.

In the following, we always assume that Λ is the natural live assignment.

Lemma 3. (Cyclicity) If $\sim$ refines $\simeq$ and $x \in \underline{s} = \underline{s} \cdot y$, then for some $i, j \leq |\underline{s}|_\sim$ and some $s \subseteq \underline{s}$, $x \cdot y^i \in s = s \cdot y^j$.

In particular, when

$$\alpha = \underbrace{u}_{\underline{s}} \underbrace{v^\omega}_{\underline{p}}$$

is a factorization in $(\simeq, \mathrel{\hat{\simeq}})$, then there is a factorization

$$\alpha = \underbrace{uv^i}_{s} \cdot (\underbrace{v^j}_{p})^\omega$$

in $(\sim, \mathrel{\hat{\sim}})$ with $s \in \underline{s}$ and $i, j \leq |\underline{s}|_\sim$. We say that the former factorization *induces* the latter.

Proof. Note that the $\sim$-states of x, $x \cdot y$, $x \cdot y^2$, ... are all among the $|\underline{s}|_\sim$ different $\sim$-states contained in $\underline{s}$. □

LFORC $\mathcal{L} = (\sim, \mathrel{\hat{\sim}}, \Lambda)$ *retracts* to LFORC $\underline{\mathcal{L}} = (\simeq, \mathrel{\hat{\simeq}}, \underline{\Lambda})$ if

(R-S) $x \sim y$ implies $x \simeq y$

(R-P) for all $\underline{s}$ of $\simeq$,
if for all s of $\sim$ contained in $\underline{s}$,
$x \mathrel{\hat{\sim}}_s y$
and
for all v and all $i \leq |\ \underline{s}\ |_\sim$,
$s(xv)^i = s$ implies $(xv)^i \mathrel{\hat{\sim}}_s (yv)^i$,
then $x \mathrel{\hat{\simeq}}_{\underline{s}} y$.

(R-Λ) for all $\underline{s}$ of $\simeq$, all x such that $\underline{s} = \underline{s}x$,
all s of $\sim$ contained in $\underline{s}$, and all $i \leq |\ \underline{s}\ |_\sim$,
if $s = sx^i$, then $[x^i]_s \in \Lambda_s$ iff $[x]_{\underline{s}} \in \underline{\Lambda}_{\underline{s}}$.

The condition (R-S) expresses that $\mathcal{L}$ *safety-refines* $\underline{\mathcal{L}}$, i.e. that the safety congruence of $\mathcal{L}$ refines the safety congruence of $\underline{\mathcal{L}}$.

Unfortunately, it is not sufficient to formulate a similarly simple requirement for the progress congruences. In fact, Example 2.0.1 shows that the minimum progress congruence may become more complicated as the safety congruence is refined! (Therefore, Theorem 2 of the technical report [3] is not correct.)

Instead, condition (R-P) expresses that the product of all $\mathrel{\hat{\sim}}_s$, where s is contained in $\underline{s}$, augmented with a condition about finite iterations, refines $\sim_{\underline{s}}$. The intuition is that when $\sim$ is collapsed to $\simeq$, an $\underline{s}$-cyclic word x in $\simeq$ may induce an s-cycle in $\sim$ only when repeated a number of times that is at most $|\underline{s}|_\sim$. Requirement (R-P) stipulates that if x and y are equivalent with respect of all such repetitions for s a subset of $\underline{s}$, then x and y are equivalent with respect to progress for $\underline{s}$.

Finally, condition (R-Λ) expresses that acceptance in $\mathfrak{L}$ is matched by acceptance in $\underline{\mathfrak{L}}$ in the following sense. Let x be an $\underline{s}$-cyclic word such that x^i is s-cyclic, where s is contained in $\underline{s}$. Then x^i is in a state of $\Lambda_{\underline{s}}$ if and only if x is in a state of Λ_s.

Note that in the case that $\sim = \simeq$, then (R-P) simply states that $x \approx_s y$ implies $x \approx_{\underline{s}} y$ and (R-Λ) states that $[x^i]_s \in \Lambda_s$ if and only if $[x]_{\underline{s}} \in \Lambda_{\underline{s}}$. Thus if $\approx_s$ and $\approx_{\underline{s}}$ are regarded as usual DFAs with final states Λ_s and $\Lambda_{\underline{s}}$, then (R-P) and (R-Λ) expresses that a usual automaton homomorphism exists from the former automaton to the latter.

Saturation by finite FORCs characterizes ω-regularity [3]. Similarly, we have

Lemma 4. The class of languages recognized by finite LFORCS is the class of ω-regular languages.

Proof. The acceptance criterion of an LFORC can easily be encoded by a nondeterministic Büchi automaton that guesses the the factorization.

Vice versa, it can be seen that any deterministic automaton with the Streett acceptance condition gives rise to an LFORC. Recall that a Streett acceptance condition consists of a list of pairs of subsets of states, called "red" and "green" states. A run is accepted if it holds for each pair that if green states of the pair occurs infinitely often, then red states of the pair occurs infinitely often. The progress information along a cycle in the automaton consists of recording which "red" and "green lights" have been seen since the beginning of the cycle. The acceptance condition for the progress automaton is that for each pair for which a green state has been encountered, also a red state has been encountered. □

Given a language L, Maler and Staiger define a *canonical* FORC $(\sim^L, \approx_s^L)$ and show that it saturates L. The corresponding *canonical* LFORC $\mathfrak{L}^L = (\sim^L, \approx_s^L, \Lambda^L)$ of any ω-regular language L is then defined as follows:

- $x \sim^L y$ if for all $\alpha, x\alpha \in L$ iff $y\alpha \in L$, and
- $\approx_s^L$ is the right congruence $\approx_s$ defined as $x \approx_s y$ iff
  ($\approx_s 1$) $sx = sy$, and
  ($\approx_s 2$) for all v, if $u \in s = sxv = syv$ then $u(xv)^\omega \in L$ iff $u(yv)^\omega \in L$
- Λ^L is the natural live assignment.

Lemma 5. $\mathfrak{L}^L$ recognizes L.

Proof. Since $L(\mathfrak{L}^L)$ is ω-regular it suffices to verify that each word of the form uv^ω is accepted if and only if it is in L.

So assume xy^ω is in L. By Lemma 1(b), xy^ω can be factorized as

$$\underbrace{xy^m}_{s}\underbrace{(y^n)^\omega}_{p}.$$

The progress state p is then in Λ_s by definition of the natural live assignment. It follows that $xy^\omega \in L(\mathfrak{L}^L)$.

Vice versa, if xy^ω is in $L(\mathfrak{L}^L)$, then it admits a factorization of the above form with p is in Λ_s. Thus there is some word uv^ω in L that also has a (s,p) factorization. By Lemma 1(c), uv^ω has a factorization

$$\underbrace{u'}_{s}\underbrace{v'}_{p}{}^{\omega}.$$

By definition of $(\sim^L, \approx_s^L)$, it follows that $xy^\omega \in L$ if and only if $u'v'^\omega \in L$. Thus $xy^\omega \in L$. $\square$

The canonicity of $\mathfrak{L}^L$ is explained by the following result.

Theorem 1. (Canonicity) Any $\mathfrak{L}$ recognizing L retracts to $\mathfrak{L}^L$.

Proof. In this proof, states of $\mathcal{L}^L$ are denoted by underlined letters.

First, we prove (R-S) (along the lines of [2] and [3]). Assume $x \sim y$. Since L is ω-regular, we just need to show that for all $uv^\omega, \alpha = xuv^\omega \in L$ iff $\beta = yuv^\omega \in L$. By saturation of $(\sim, \approx)$ it suffices to show that α and β admit a common factorization. But α has an (s,p)-factorization

$$\alpha = \underbrace{xuv^m}_{s}(\underbrace{v^n}_{p})^\omega$$

according to $(\sim, \approx_s)$ and since $\sim$ is a right congruence and $x \sim y$, we infer $xuv^m \sim yuv^m \in s$. Thus,

$$\beta = \underbrace{yuv^m}_{s}(\underbrace{v^n}_{p})^\omega$$

is an (s,p)-factorization according to $(\sim, \approx)$.

Second, to show that (R-P) holds, pick $\underline{s}$ of $\sim^L$ and assume

$$\forall s \subseteq \underline{s} : x \approx_s y \wedge (\forall v, \forall i \leq |\ \underline{s}\ |_\sim, s(xv)^i = s \Rightarrow (xv)^i \approx_s (yv)^i) \tag{1}$$

We must prove

$$\underline{s}x = \underline{s}y \tag{2}$$

and

$$\forall u, v : u \in \underline{s} = \underline{s}xv = \underline{s}yv \Rightarrow (u(xv)^\omega \in L \Leftrightarrow u(yv)^\omega \in L) \tag{3}$$

To show (2), pick some $s \subseteq \underline{s}$. Then by (1), $x \approx_s y$ and thus by (FORC), $sx = sy$. Now since $\sim$ refines $\sim^L$, $\underline{s}x = [sx]_{\sim^L} = [sy]_{\sim^L} = \underline{s}y$.

To show (3), pick some u and v such that $u \in \underline{s} = \underline{s}xv$. We wish to show that $u \cdot (xv)^\omega$ and $u \cdot (yv)^\omega$ have a common factorization in $\mathfrak{L}$. Now, $u \cdot (xv)^\omega$ has an $\mathfrak{L}^{\mathrm{L}}$ factorization

$$\underbrace{u}_{\underline{s}}(\underbrace{xv}_{\underline{p}})^\omega$$

for some $\underline{p}$ of $\sim^L_{\underline{s}}$. Thus by Lemma (Cyclicity) for some s in $\underline{s}$ and some $i, j \leq |\underline{s}|_\sim$

$$\underbrace{u \cdot (xv)^i}_{s}\underbrace{((xv)^j)^\omega}_{p}$$

is a factorization in $\mathfrak{F}$ for some p of $\approx_s$. But we infer from (1) that

$$(xv)^j \approx_s (yv)^j$$

Thus

$$\underbrace{u \cdot (yv)^i}_{s}\underbrace{\left((yv)^j\right)^\omega}_{p}$$

is an (s,p)-factorization. Thus by saturation of $\mathfrak{L}$, $u \cdot (xv)^\omega \in L$ iff $u \cdot (yv)^\omega \in L$.

Third, to show (R-L), we fix $\underline{s}$ of $\sim^L$, $s \subseteq \underline{s}$, $i \leq |\ \underline{s}\ |_\sim$, and x such that $\underline{s}x = \underline{s}$ and $s = sx^i$. Let u be such that $[u]_\sim = s$. Then $[u]_{\sim^L} = \underline{s}$ and

$$\begin{aligned}
[x^i]_s \in \Lambda_s \quad &\text{iff} \quad \begin{cases}\text{by saturation of } \mathfrak{F} \text{ and} \\ \text{assumption } s = sx^i\end{cases} \\
u(x^i)^\omega \in L \quad &\text{iff} \\
ux^\omega \in L \quad &\text{iff} \quad \begin{cases}\text{by definition of } \Lambda^L \text{ and} \\ \text{since } [u]_{\sim^L} = \underline{s} = \underline{s}x\end{cases} \\
[x]_{\underline{s}} \in \Lambda^L_{\underline{s}} &
\end{aligned}$$

□

Theorem 2. (Language Preservation) If $\mathfrak{L}$ recognizes L and $\mathfrak{L}$ retracts to $\underline{\mathfrak{L}}$, then $\underline{\mathfrak{L}}$ recognizes L.

Proof. Let

$$\alpha = \underbrace{u}_{\underline{s}}\underbrace{v^\omega}_{\underline{p}}$$

be a word admitting an $(\underline{s}, \underline{p})$–factorization in $\underline{\mathfrak{L}}$. Then this factorization retracts to an (s,p)–factorization

$$\alpha = \underbrace{uv^i}_{s}(\underbrace{v^j}_{p})^\omega$$

in $\mathfrak{L}$, where $s \subseteq \underline{s}$, $j \leq |\ \underline{s}\ |_\sim$, and $sv^j = s$. Thus by (R-L), $p \in \Lambda_s$ iff $\underline{p} \in \Lambda_{\underline{p}}$. Thus $\alpha \in L(\mathfrak{L})$ iff $\alpha \in L(\underline{\mathfrak{L}})$. □

$\mathfrak{F} = (\sim, \approx)$ is $\sim$-*canonical* for L if $(\sim, \approx)$ saturates L and any other FORC with safety congruence $\sim$ and saturating L retracts to $\mathfrak{F}$.

Proposition 1. ($\sim$-Canonicity) If $\sim$ refines $\sim_L$, then a $\sim$-canonical $\mathfrak{F}$ exists.

Proof. Define $\overset{\circ}{\sim}_s$ by

(4) $x \overset{\circ}{\sim}_s y$ if $sx \sim sy$ and

for all $v, s = sxv$ implies $u(xv)^\omega \in L$ iff $u(yv)^\omega \in L$, where $u \in s$.

Since $\sim$ refines $\sim_L$, the choice of u in (4) is immaterial and it can be seen that $\mathfrak{F}$ saturates L. Also, it is not difficult to see that any other FORC with $\sim$ as safety congruence retracts to $\mathfrak{F}$. □

We noted before that the progress and live state conditions for a retraction involving LFORCs with the same safety congruence essentially express DFA homomorphisms. Thus for a fixed safety congruence $\sim$, DFA minimization that respects requirement (FORC) can be applied to obtain the $\sim$-canonical LFORC.

We shall next show that when the safety congruence becomes simpler, the progress congruences get more complicated but they still essentially contain the simpler progress congruences if these are minimum with respect to their safety congruence.

Assume $\mathfrak{F}$ retracts to $\underline{\mathfrak{F}}$. Then $\underline{\mathfrak{F}}$ *progress-refines* $\mathfrak{F}$ if for $s \in \underline{s}$,

$$x \overset{\circ}{\simeq}_{\underline{s}} y \text{ and } sx \sim sy \text{ implies } x \overset{\circ}{\sim}_s y$$

Thus the product of $\overset{\circ}{\simeq}$ and $\sim$ refines $\overset{\circ}{\sim}$.

Theorem 3. (Progress Refinement) If $\mathfrak{F}$ is $\sim$-canonical and $\mathfrak{F}$ retracts to $\underline{\mathfrak{F}}$, then $\underline{\mathfrak{F}}$ progress-refines $\mathfrak{F}$.

Proof. Assume $x \overset{\circ}{\simeq}_{\underline{s}} y$ and $sx \sim sy$. To prove that $x \overset{\circ}{\sim}_s y$, it suffices to prove that $\alpha = u(xv)^\omega \in L$ iff $\beta = u(yv)^\omega \in L$ whenever $u \in s = sxv$. Now since $xv \simeq_{\underline{s}} yv$ and $u \in \underline{s} = \underline{s}xv$, both α and β admit a $(\underline{s}, \underline{p})$-factorization in $\underline{\mathfrak{F}}$, where $\underline{p} = [xv]_{\underline{s}} = [yv]_{\underline{s}}$. Since $\mathfrak{F}$ retracts to $\underline{\mathfrak{F}}$ and $\mathfrak{F}$ saturates L, also $\underline{\mathfrak{F}}$ saturates L. Thus $\alpha \in L$ iff $\beta \in L$. □

4. Collapsed LFORCS

Let $\mathfrak{L} = (\sim, \overset{\circ}{\sim}, \Lambda)$ be an LFORC and $\simeq$ be a refinement of $\mathfrak{L}$. Define the *collapsed* LFORC $\underline{\mathfrak{L}} = (\simeq, \overset{\circ}{\simeq}, \underline{\Lambda})$ by

(5) $x \overset{\circ}{\simeq}_{\underline{s}} y$ iff for all $s \subseteq \underline{s}$, $x \overset{\circ}{\sim}_s y$ and
for all v and $i \leq |\underline{s}|_\sim$,
$\underline{s}xv = s$ implies $(xv)^i \overset{\circ}{\sim}_s (yv)^i$

(6) $\underline{p} \in \Lambda_{\underline{s}}$ iff $[x^i]_s \in \Lambda_s$, where $x \in \underline{p}$,
$\underline{s}x = \underline{s}$, $sx^i = s$, and $s \subseteq \underline{s}$

It can be seen that (5) always defines a right congruence. The definition (6) may not always make sense. The *Consistency Requirement* is that for all $\underline{s}$ the membership of a progress state $\underline{p}$ in $\Lambda_{\underline{s}}$ is determined unambiguously by (6) for any choice of x, i, and s.

Lemma 6. The collapsed LFORC $\underline{\mathfrak{L}}$ recognizes $L(\mathfrak{L})$ if the Consistency Requirement holds. If the Consistency Requirement does not hold, then $\simeq$ does not refine $\sim^L$.

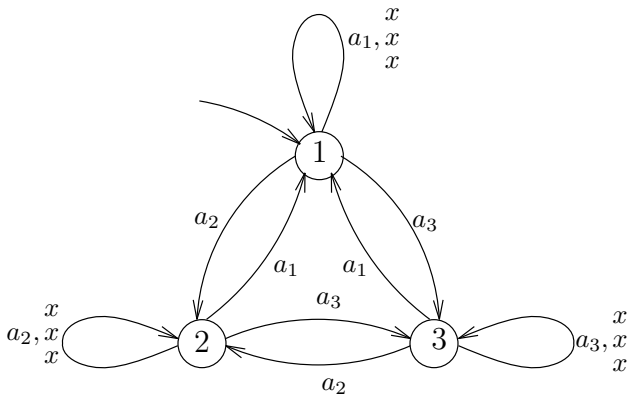
Proof. If the Consistency Requirement holds then $\underline{\mathfrak{L}}$ is a refinement of $\mathfrak{L}$.

If the Consistency Requirement does not hold, then there are $\underline{s}$, $\underline{p}$ of $\overset{\circ}{\sim}_s$, s, s' of $\sim$ contained in $\underline{s}$, $x \overset{\circ}{\sim}_{\underline{s}} y$, where $x, y \in \underline{p}$, and $i, j \leq |\underline{s}|_\sim$ with $\underline{s}x = \underline{s} = \underline{s}y$, $sx^i = s$, $s' = s'y^j$. such that $[x^i]_s \in \Lambda_s$ and $[y^j]_{s'} \notin \Lambda_{s'}$. Thus if $u \in s$ and $v \in s'$, then $ux^\omega \in L$ and $vy^\omega \notin L$. By (5) and since $x \overset{\circ}{\sim}_{\underline{s}} y$, $x^i \sim_s y^i$. Thus $uy^\omega \in L$. But then u and v are not equivalent with respect to $\sim^L$. □

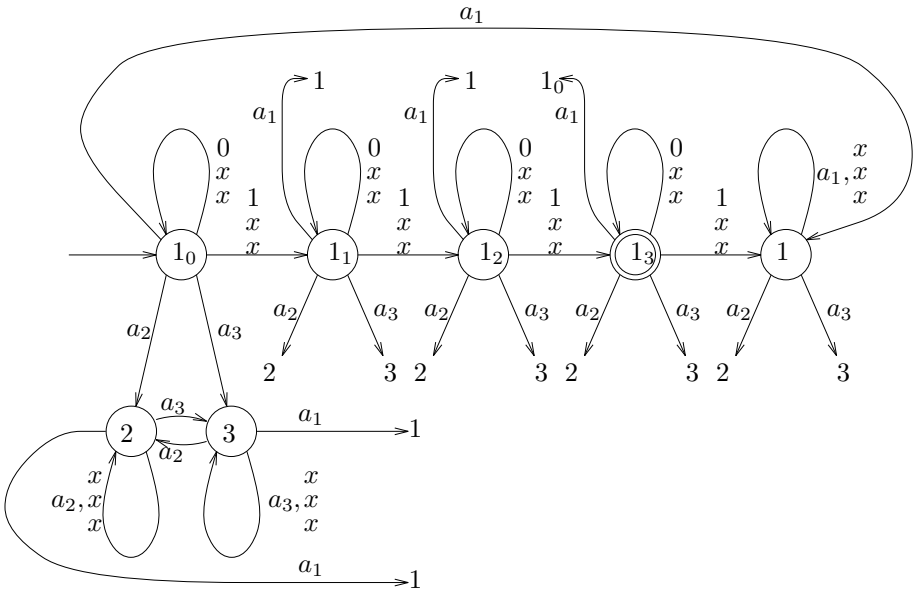
5. Lower Bound

We establish an exponential lower bound for minimization, that is, there is an infinite family L_n of languages that can be represented by LFORCs of size $O(n)$ but whose canonical LFORCs contain a progress automaton with n^n states.

We let Σ_n consist of n *proper* letters $a_1, \ldots, a_n$ and of the 2^n bit vectors in $\mathbb{B}^n$. A word α is in L_n if from some point on there is a proper letter a_i such that a_i occurs infinitely often and no other proper letter occurs; also, the number of 1s in track i between two consecutive letters a_i must be exactly n. There is certainly only one safety class in this language since a word is recognized by properties of its tail. Define a linearly big LFORC recognizing L_n by using the safety congruence represented by the automaton $\mathfrak{S}_n$, which is depicted for $n = 3$ as



Thus each state s_i is a sink for the letter a_i, and a bit vector does not change the state. Continuing our example for $n = 3$, we can represent the progress congruence for safety state 1 by the automaton



This congruence respects the safety congruence and any factorization of α according to s, and a progress state of this automaton yields the right answer as to whether α is in L_n.

Thus it can be seen that for any n, the progress automaton has $2n+1$ states. We conclude that L_n can be represented by an LFORC of size $O(n)$.

For the lower bound, assume that the progress congruence of $\mathfrak{F}^{L_n}$ has less that $(n+1)^n$ states. If we say that the *#-signature* of a word over $\mathbb{B}^n$ is determined by the number of 1s in each component, then there are $(n+1)^n$ signatures, where all components have at most n 1s. Thus there are two words x and y over $\mathbb{B}^n$ with different such signatures that lead to the same progress state. For some component i, x and y contains a different number of 1s and it is then possible to find a word $u \in \mathbb{B}^k \cdot a_i$ such that $(xu)^\omega \in L_n$ and $(xy)^\omega \notin L_n$. This contradicts that $\mathfrak{F}^{L_n}$ accepts L_n. Thus $\mathfrak{F}^{L_n}$ has more than $(n+1)^n$ states.

Proposition 2. There is an infinite family of LFORCS $\mathfrak{L}^n$ of size $O(n)$ whose corresponding canonical LFORCS have size at least n^n.

6. The Automata-theoretic View and Upper Bound

As indicated in Example 2.0.1, an LFORC $\mathfrak{L}$ can be represented by a family of automata. The safety relation $\sim$ is represented by a *safety automaton* $\mathfrak{S} = (S, s^0, \delta)$ with state space S, initial state s^0, and deterministic transition function $\delta : \Sigma \to S \to S$ such that

$$x \sim y \text{ iff } \mathfrak{S}(x) = \mathfrak{S}(y),$$

where $\mathfrak{S}(x)$ denotes the state of $\mathfrak{S}$ upon reading x. Thus we may continue to identify each safety class s with a state $s \in S$. For each such s, we represent $\overset{\circ}{\sim}_s$ and Λ_s as

an automaton $\mathfrak{P}_s = (P_s, p_s^0, \delta_s, P_s^F)$, where P_s^F is a set of final states. Here each p represents a progress state such that $x \stackrel{\circ}{\sim}_s y$ if only if $\mathfrak{P}_s(x) = \mathfrak{P}_s(y)$ and $[x]_s \in \Lambda_s$ if and only if $\mathfrak{P}_s(x) \in P_s^F$. The *family of automata* or *FDFA* so defined is denoted $(\mathfrak{S}, \mathfrak{P})$.

To formulate a retraction as a homomorphism, we need an operation IC that transforms $\mathfrak{P}_s$ into an automaton $IC^j(\mathfrak{P}_s)$ that represents the iterative condition of (R-P) as follows:

$$IC(\mathfrak{P}_s)(x) = IC(\mathfrak{P}_s)(y) \tag{7}$$

iff

$$x \stackrel{\circ}{\sim}_s y \text{ and for all } v \text{ and } i \leq |f^{-1}(\underline{s})|,\ s(xv)^i = s \Rightarrow (xv)^i \stackrel{\circ}{\sim}_s (yv)^i$$

Lemma 7. For s and j, an automaton $IC(\mathfrak{P}_s)$ exists such that (7) holds. The automaton is at most exponential in size of $\mathfrak{P}_s$.

Proof. The proof consists of defining an automaton $\mathfrak{A}$ that is able to distinguish words according to or even more strictly than the criterion (7). The automaton $IC(\mathfrak{P}_s)$ is then a coarsest refinement of $\mathfrak{A}$.

Note that a transition relation $\delta : \Sigma \to P \to P$ can be extended to a function, also denoted δ, of type $\Sigma^* \to P \to P$ by defining $\delta(a_0 \cdots a_n) = \delta(a_n) \circ \cdots \circ \delta(a_0)$. By the standard technique for obtaining syntactic monoids, let $\mathfrak{A}$ constructed from $\mathfrak{P}_s$ be an automaton whose state space consists of the functions $\delta_s(x)$ such that $q = \mathfrak{A}(x)$ is the function $\delta_s(x)$. This automaton is exponential in size of $\mathfrak{P}_s$.

Each q determines a function $q : P \to P$, where $q(p)$ is the only p' such that the entry (p, p') is 1.

Since each q also determines the state $p = q(p^0)$ reached from the initial state p^0, we may define an operation $q \cdot v$, which denotes a state in P, namely, $q \cdot v = \delta_s(v)p$. Thus if $\mathfrak{A}(x) = q$, then $q \cdot v$ is simply $\delta_s(xv)$. Moreover, we may even define an operation $(q \cdot v)^i$ so that if $\mathfrak{A}(x) = q$, then $(q \cdot v)^i$ is $\delta_s((xv)^i)$. This is done by letting $(q \cdot v)^i$ be $\delta_s(v) \circ q \circ \cdots \circ \delta_s(v) \circ q \circ \delta_s(v)(q(p^0))$, where $\delta_s(v) \circ q$ is repeated $i-1$ times.

$\mathfrak{A}$ does not quite calculate what is needed in (7), but satisfied the weaker requirement:

$$\mathfrak{A}(x) = \mathfrak{A}(y) \tag{8}$$

implies

$$x \stackrel{\circ}{\sim}_s y \text{ and for all } v \text{ and } i \leq |\underline{s}|,\ s(xv)^i = s \Rightarrow (xv)^i \stackrel{\circ}{\sim}_s (yv)^i$$

To see this, assume $\mathfrak{A}(x) = \mathfrak{A}(y)$. Then $\mathfrak{P}_s(x) = \mathfrak{P}_s(y) = q(p^0)$, whence $x \stackrel{\circ}{\sim}_s y$. Moreover, $\delta_s((xv)^i) = (q \cdot v)^i = \delta_s((yv)^i)$. Thus in particular, it holds that if $i \leq |f^{-1}(\underline{s})|$, then $s(xv)^i = s \Rightarrow (xv)^i \stackrel{\circ}{\sim}_s (yv)^i$.

Note that by (FORC), there is a subset P_s^s of progress states such that $s = sx$ if and only if $\mathfrak{P}_s(x) \in P_s^s$.

To make the other direction of (7) hold, we will shrink $\mathfrak{A}$ according to the following characterization of states:

$$\chi(q) = (q(p^0), \{(p, i, L) \mid i \leq |f^{-1}(\underline{s})| \text{ and } L = \{v \mid (q \cdot v)^i = p \in P^s_s\})$$

The Ls are all regular languages and so χ can be computed by operations on usual finite-state automata. The function χ induces a partition of $\mathfrak{A}$. Let $IC(\mathfrak{P}_s)$ be the automaton corresponding to the coarsest partition of $\mathfrak{A}$ that refines the one induced by χ. We also use qs to denote the states of this automaton. If $q = IC(\mathfrak{P}_s)(x)$, then $\chi(q) = (\mathfrak{P}_s(x),\ \{(p, i, L) \mid i \leq j \text{ and } L = \{v \mid (x \cdot v)^i = p \in P^s_s\})$.
Thus (7) is satisfied. □

An *FDFA* homomorphism $h : (\mathfrak{S}, \mathfrak{P}) \to (\underline{\mathfrak{S}}, \underline{\mathfrak{P}})$ consists of

- a transition system homomorphism (i.e. a mapping respecting right concatenation) $f : \mathfrak{S} \to \underline{\mathfrak{S}}$; and
- for each $\underline{s} \in \underline{S}$, a transition system homomorphism

$$g : \bigotimes_{s \in \underline{s} : f(s) = \underline{s}} IC(\mathfrak{P}_s) \to \underline{\mathfrak{P}}_{\underline{s}},$$

where $\otimes$ denotes the transition system cross product, such that for all $s \in S$ with $f(s) = \underline{s}$ and all $i \leq |f^{-1}(\underline{s})|$,

$$\begin{aligned} \sqrt[i]{L(\mathfrak{P}_s)} \cap L &= L(\underline{\mathfrak{P}}_{\underline{s}}) \cap L, \text{ where} \\ L &= L_{\underline{s}}(\underline{\mathfrak{S}}) \cap \sqrt[i]{L_s(\mathfrak{S})} \end{aligned}$$

and $L_s(\mathfrak{S}) = \{x \mid \delta(x)(s) = s, \text{i.e. } sx = s\}$ and $\sqrt[i]{L}$ denotes the language $\{x \mid x^i \in L\}$.

Lemma 8. Let $(\mathfrak{S}, \mathfrak{P})$ be the automata representation of $\mathfrak{L}$ and let $(\underline{\mathfrak{S}}, \underline{\mathfrak{P}})$ be the automata representation of $\underline{\mathfrak{L}}$. Then $\mathfrak{L}$ retracts to $\underline{\mathfrak{L}}$ if and only if there is an *FDFA* homomorphism from $(\mathfrak{S}, \mathfrak{P})$ to $(\underline{\mathfrak{S}}, \underline{\mathfrak{P}})$.

Proof. Requirement (R-S) and (R-P) correspond to the existence of f and g. Requirement (R-Λ) is then encoded correctly as shown above since $L = L_{\underline{s}}(\underline{\mathfrak{S}}) \cap \sqrt[i]{L_s(\mathfrak{S})}$ defines the x such that $\underline{s} = \underline{s}x$ and $s = sx^i$. □

Let $(\mathfrak{S}^L, \mathfrak{P}^{\mathrm{L}})$ be the FDFA representation of $\mathfrak{L}^L$. We can now restate Theorem 1 and Theorem 2 as theorems about FDFAs and their homomorphisms.

Theorem 1′. (Canonicity) Any FDFA recognizing L allows a homomorphism to $(\mathfrak{S}^L, \mathfrak{P}^{\mathrm{L}})$.

Theorem 2′. (Language Preservation) If $(\mathfrak{S}, \mathfrak{P})$ recognizes L and there is a homomorphism from $(\mathfrak{S}, \mathfrak{P})$ to $(\underline{\mathfrak{S}}, \underline{\mathfrak{P}})$, then $(\underline{\mathfrak{S}}, \underline{\mathfrak{P}})$ recognizes L.

Proposition 3. (Upper Bound) If $\mathfrak{L}$ has size n and retracts to $\mathfrak{L}^L$, then the size of $\mathfrak{L}^L$ is at most n^{n^2}.

Proof. Clearly the safety congruence can only shrink. For the progress part, use the FDFA representation $(\mathfrak{S}, \mathfrak{P})$ of $\mathfrak{L}$ and assume that the canonical safety state $\underline{s}$ is refined by the s of $\mathfrak{S}$ such that $f(s) = \underline{s}$. Observe that $\bigotimes_{s \in S: f(s) = \underline{s}} IC(\mathfrak{P}_s)$ refines the canonical progress automaton or congruence for $\underline{s}$. Thus the canonical congruence is of size at most $(n^n)^n$, which is n^{n^2}. □

Acknowledgements. Thanks to Dexter Kozen, Ludwig Staiger, and Thomas Wilke for discussions about ω-regular representations.

References

1. B. Alpern and F.B. Schneider. Defining liveness. *Information Processing Letters*, 21:181–185, Oct. 1985.
2. A. Arnold. A syntactic congruence for rational ω-languages. *Theoretical Computer Science*, 39:333–335, 1985.
3. O. Maler and L. Staiger. On syntactic congruences for ω-languages. Technical Report 93-13, Aachener Informatik–Berichte, 1993. A preliminary version appeared in: Proc. STACS 93, LNCS **665**, Springer–Verlag, Berlin 1993, pp. 586–594.).
4. T. Wilke. An Eilenberg theorem for ∞-languages. In *Proc. 18th Inter. Coll. on Automata, Languages, and Programming, LNCS 510*, pages 588–599. Springer Verlag, 1991.

Recent Publications in the BRICS Report Series

RS-94-1 **Glynn Winskel.** ***Semantics, Algorithmics and Logic: Basic Research in Computer Science. BRICS Inaugural Talk*****. February 1994, 8 pp.**

RS-94-2 **Alexander E. Andreev.** ***Complexity of Nondeterministic Functions*****. February 1994, 47 pp.**

RS-94-3 **Uffe H. Engberg and Glynn Winskel.** ***Linear Logic on Petri Nets*****. February 1994, 54 pp. Appear in:** ***Proceedings of REX '93*** **(eds. J. W. de Bakker et al.), LNCS 803, 1994.**

RS-94-4 **Nils Klarlund and Michael I. Schwartzbach.** ***Graphs and Decidable Transductions based on Edge Constraints*****. February 1994, 19 pp. Appears in:** ***Trees in Algebra and Programming CAAP '94*** **(ed. S. Tison), LNCS 787, 1994.**

RS-94-5 **Peter D. Mosses.** ***Unified Algebras and Abstract Syntax*****. March 1994, 21 pp. To appear in:** ***Recent Trends in Data Type Specification*** **(ed. F. Orejas), LNCS 785, 1994.**

RS-94-6 **Mogens Nielsen and Christian Clausen.** ***Bisimulations, Games and Logic*****. April 1994, 37 pp. Full version of paper appearing in:** ***New Results and Trends in Computer Science*****, pages 289–305, LNCS 812, 1994.**

RS-94-7 **André Joyal, Mogens Nielsen, and Glynn Winskel.** ***Bisimulation from Open Maps*****. May 1994, 42 pp. Journal version of LICS '93 paper.**

RS-94-8 **Javier Esparza and Mogens Nielsen.** ***Decidability Issues for Petri Nets*****. May 1994, 23 pp. Appears in EATCS Bulletin 52, pages 245–262, 1994.**

RS-94-9 **Gordon Plotkin and Glynn Winskel.** ***Bistructures, Bidomains and Linear Logic*****. May 1994, 16 pp. To appear in the proceedings of ICALP '94, LNCS, 1994.**

RS-94-10 **Jakob Jensen, Michael Jørgensen, and Nils Klarlund.** ***Monadic Second-order Logic for Parameterized Verification*****. May 1994, 14 pp.**

RS-94-11 **Nils Klarlund.** ***A Homomorphism Concept for ω-Regularity*****. May 1994, 16 pp.**