Research Note
Diagonalization techniques applied to reversible programming languages

Armando B. Matos

December 2016

Abstract

It is shown that there exist recursive (total computable) functions that are not PR (primitive recursive) and whose co-domain is $\{0,1\}$. (However the *computation time* of recursive but not PR functions grows faster that any PR function.)

For reversible $\mathbb{Z}^n \to \mathbb{Z}^n \to$ transformations we show that,
– if L is a programming language whose programs are total and implement (only) bijections, and
– if the identity transformation is implemented by an infinity of programs of L, then
there are constructible computable bijections that is not implemented by any program of L.

Notes.
– File: `disgonaliz.tex`
– Reference in `bib.bib`: `matos-diag`

Research Note

SRL /ESRLlanguages: (i) are extra registers needed? (ii) programs with registers initialized

Armando B. Matos

December 2016

Abstract

Programs written in the SRL language with some input values fixed (that is, the initial values of some registers are constant) are considered. Moreover assume that when the computation is finished, all registers except one are discarded.

In this setting we try to answer the following questions: what class of unary (all inputs except one are fixed) functions that can be implemented using this method?

A complete answer to that question was not obtained. However, we show how to implement polynomials, some exponential functions and show that the following closure property holds: the sum and product of "implementable" functions is also "implementable".

Notes.
– File: `topic-extra-regs/extra-registers.tex`
– Reference in `bib.bib`: `matos-extra`

On SRL transformations: the maximum rate of growth and LOOP representation.

Armando Matos, Luca Roversi and Luca Paolini

Abstract

The purpose of this note is to
– prove that the asymptotic upper bounds of primitive recursive (PR) functions and of SRL transformations are essentially the same;
– represent the SRL constructs in the LOOP language;
– transcribe some related conversations with Luca Roversi and Luca Paolini.

The purpose of this note is to prove that the asymptotic upper bounds of primitive recursive (PR) functions and of SRL transformations are essentially the same.

Relatively to PR functions, we refer the reader to the LOOP programming language, see~\cite{MR1,MR2}. To establish upper bounds on SRL transformations, we adopt the following assumptions:

(a) Only non-negative values are considered, so that a direct comparison with PR functions is possible.
(b) The largest of the inputs of a SRL transformation, max(x1,...,xn) is used as a measure of the input size. A similar assumption is used for PR functions.
(c) The value of a selected output is used as the output size.
(d) The number of input registers of the SRL transformation is chosen conveniently.

Relatively to (b), and in order to simplify the proof, we "study" only the case in which all the inputs except one are fixed. This allows a direct comparison with PR unary functions.

Research Note

Can the "if" instruction be implemented in the SRL/ESRL programming languages?

Armando B. Matos
January 2017

Abstract

Let the instruction if $n(P)$ be a possible extension of the SRL language. Its meaning is "if $n \geq 0$ then P". Call it the "basic conditional instructions".

1) Suppose that the basic conditional instruction is added to the SRL language. Can we implement more elaborate conditional instructions, like "if $(n \geq 2) \vee (x < y)$ then P"? Which "extended" conditional instructions can be implemented with the aid of the basic conditional instruction?

2) Can the basic conditional instruction be implemented in (pure) SRL language?

Although we do not fully answer this questions, we consider the possibility of implementation (of the basic and extended conditional instruction) in very restricted SRL languages in which the number of registers and the number of loops (for instructions) is very small.

Notes.

– File: `if-instruction.tex`

– Reference in `bib.bib`: `matos-if-inst`

Contents

Preliminary Notes

SRL-like languages and Kolmogorov complexity

Armando B. Matos

May 2017

Abstract

Consider a configuration $C(t)$ of a reversible computation, where t is the time corresponding to the configuration. At first it may seem that the Kolmogorov complexity of $C(t)$ is constant, because $C(t)$ may be described by a minimum program for the tuple of input values $\overline{x}$ and reciprocally, $\overline{x}$ can be described by a minimum program for C(t).

However, a more detailed analysis shows that this is not always the case. Consider a SRL computation. An intermediate configuration $C(t)$ can be described by (i) a constant part (essentially the text of the program), (ii) a minimum program describing the input values, and (iii) the time t. If the part (iii) is negligible, we have the so called "input dominated complexity".

The Kolmogorov complexity of a configuration in this general case is studied. In particular, we describe situations in which the term (iii) dominates, "time dominated complexity": we exhibit SRL programs whose running time is so large that the terms (i) and (ii) are negligible. In such cases, an intermediate configuration has almost always a complexity $\log_2 t \approx K(t)$, where t is the configuration time. During the computation, and also when it finishes, the value of $K(t)$ may however be much smaller.

Notes.
– File: `kolgo.tex`
– Reference in `bib.bib`: `matos-kolgo`
– In this note the word "reversibilization" denotes a reversible simulation of an irreversible computation.

Contents

Preliminary Notes

Comparison of several variants of the SRL language

Armando B. Matos
January 2017

Abstract

Consider the SRL language with auxiliary registers. We classify the use of these registers: they may be initialized or not and may have or not the same final value. For instance, a 0-*reg* has the initial value 0 and a 00-*reg* has also the final value 0.

We describe functions that can only be implemented in SRL with the aid of (possibly initialised) auxiliary registers.

We also study the possibility of implement primitive recursive functions in SRL (with auxiliary registers). The work of Luca Roversi, Luca Paolini, and Mauro Piccolo is mentioned.

Notes.
– File: `topic-languages/languages.tex`
– Reference in `bib.bib`: `matos-languages`

Contents

Preliminary Notes

On the additional memory used to "reversibilize" computations

Armando B. Matos

January 2017

Abstract

First, we review several models of computation that either are reversible or can be "made reversible", namely logic gates, Prolog programs, reversible communicating systems (processes), Turing machines and SRL-like languages.

A non-reversible computation can be made reversible with the aid of additional memory, often organised as a stack. In the case of the Turing machine, an initially blank tape is used to implement the stack. In the case of a register machine a zero-initialised register can be used to code a stack.

We argue that, both under the mathematical and the physical points of view, this initialised memory "destroys" the reversible character of the whole computation: "how can we recover the contents of a tape before it is erased?", or "what was the contents of a zero-initialised register (before the computation)?". Initialisations correspond to the destruction of information / decrease of entropy.

Notes.
– File: `memory.tex`
– Reference in `bib.bib`: `matos-memory`
– In this note the word "reversibilization" denotes a reversible simulation of an irreversible computation.

Contents

"Reversibilizations" often look like this:

> *Every partial recursive function can be computed by a reversible Turing machine [...] Take the standard irreversible Turing machine computing that function. We modify it by adding an auxiliary storage tape [...]* (from [BTV01])

Research Note

ForSwap and SRL-like languages with 0-initialized registers: can the "if" instruction be implemented?

Armando B. Matos

September 2017

Abstract

A set $S \subseteq \mathbb{Z}$ is *periodic* if for some positive integer p and for every $n \in \mathbb{Z}$ we have $n \in S$ iff $n+p \in S$. In this report we describe the implementation in the SRL language of the instruction "if $a \in S$(then P)" where (i) a is a SRL register, (ii) S is a periodic set, and (iii) P is a SRL program.
Keywords: SRL language, ForSwap language, periodic set, if instruction.

Notes.

– File: `observations.tex`

– Reference in `bib.bib`: `matos-if-obs`, [Mat].

Contents

Integer reversible transformations

Programs that can run backwards

Armando B. Matos

Departamento de Ciência de Computadores
Universidade do Porto
May 2017

Research Note

On some decision problems related to a simple reversible language

Armando B. Matos, Luca Roversi, Luca Paolini

Latest version, October 12, 2017

Abstract

There are several interesting decision problems related with the "Simple Reversible Language" (SRL) and its variants. In this note the following results are proved.

1. The following decision problems are equivalent: (i) "are two given SRL programs equivalent?", (ii) "is a given SRL program equivalent to the identity (null) program?", (iii) "is the composition of two SRL programs equivalent to the identity program?". (The proof is easy.)
2. The decision problem "does a given SRL program have a fixed point?" is undecidable, and complete in the class Σ_0^1.
3. Several other fixed point SRL decision problems are also undecidable.
4. "Half-zero" problems are SRL decision problems in which half of the input registers have the initial value 0. Several natural "half-zero" problems are undecidable.

A summary of the results (and some open problems) can be seen in Figure 4, page 18.

Contents

Notes.
– File: `proofs.tex`
– Reference in `bib.bib`: `matos-proofs`

Decision problems related with SRL-like languages
Current status of knowledge

Luca Roversi, Luca Paolini, Armando Matos

June 2017

Preliminary

My notes on "subatomic proof systems"

Armando B. Matos
Started: April 2017

Abstract

Personal notes on the serial/parallel reorganisation expressed as an inference rule.

Notes.

– File: `splittable.tex`

– Reference in `bib.bib`: `matos-split`

Preliminary Notes
Groups and SRL -like languages

Armando B. Matos
June 2017

Abstract

In the context of the SRL language we have the language *monoid* and the $\mathbb{Z}^\omega$ *group* of transformations, each element of which is a function $t : \mathbb{Z}^\omega \to \mathbb{Z}^\omega$. To each program P corresponds a transformation $\tau(P)$, such that $\tau(P;Q) = \tau(Q) \cdot \tau(P)$. Among others, the following topics are included in the discussion:

1. Rotation groups implemented with swap's
2. SRL-like languages and the Post equivalence problem.

Notes.
– File: `srl-groups.tex`
– Reference in `bib.bib`: `matos-srl-groups`

Contents

Preliminary Note

For each positive integer k there is a SRL program $P_k(n, a, b)$
such that, after in the computation $P_k(n, 0, 0)$,
the final contents of a, of b and of n are asymptotically larger than

$$\underbrace{2 \uparrow (2 \uparrow \ldots (2 \uparrow n))}_{k \text{ 2's}}$$

Armando B. Matos[a]
May 2017

[a] Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto

Abstract

It is proved that for every positive integer k there is a SRL program with a register n and the other registers initialised with 0, such that the final value of any register exceeds an exponential tower of k 2's with the initial value of n at the top. In symbols,

$\forall x_i$: final value of $x_i > 2^{2^{\cdots^{2^n}}}$ where the number of 2's is k.

(In Knuth tower notation the exponential tower is denoted by $2 \overset{k}{\Uparrow} n$.)

For every k the corresponding program can be be easily exhibited and uses only 3 registers (one of them is n and the other two are initialised with 0.

Notes.
– File: `tower.tex`
– Reference in `bib.bib`: `matos-tower`

Contents