On the expressivity of total reversible programming languages

Luca Paolini[†] and Luca Roversi[†] and Armando Matos[‡]

[†] Università degli Studi di Torino
[‡] Universidade do Porto

July 2020

Outline

- Introduction: Motivations
  Primitive Recursive Functions
- Problem: Genesis of SRL
  Questions about SRL
- Solution: Test-For-Zero
  Representation of RPP
- Discussion: Conclusions
  Future Works

Motivations

The initial studies on the reversible computing are related to the interest for the thermodynamic of the computation.

Motivations

The initial studies on the reversible computing are related to the interest for the thermodynamic of the computation.

However, reversible computing is relevant for many other applications; as the following classic applications:

- Lossless compression procedures, many kinds of cryptographic procedures, and so on.
- A wide number of related cases arise when we use a backtracking mechanisms.
- Core of many computing model (e.g. quantum one).

Motivations

The initial studies on the reversible computing are related to the interest for the thermodynamic of the computation.

However, reversible computing is relevant for many other applications; as the following classic applications:

- Lossless compression procedures, many kinds of cryptographic procedures, and so on.
- A wide number of related cases arise when we use a backtracking mechanisms.
- Core of many computing model (e.g. quantum one).

A foundational theory of reversible computing should ease the development of all the above applications but not only.

Summary on Primitive Recursive Functions

Primitive recursive functions (PR) identify a total core of classic computing.

- PR include almost all common (total) functions (on natural numbers).
- PR are simple and endowed with a straightforward semantics.
- PR can be easily extended to grasp the class of all recursive functions.
- PR are sufficient to check if a (finite) computation is correct.

Summary on Primitive Recursive Functions

PR is the smallest class of functions on natural numbers including:

- the zero-function $\mathsf{Z}(x) = 0$,
- the successor $\mathsf{S}(x) := x + 1$
- projections $\pi_i^k(x_1, \ldots, x_k) := x_i$ for all $k \geq i \geq 1$,

and it is closed under:

- composition, viz. the schema that given $g_1, \ldots, g_m, h$ of suitable arities, produces $f(\overrightarrow{x}) := h(g_1(\overrightarrow{x}), \ldots, g_m(\overrightarrow{x}))$, and
- primitive recursion, viz. the function f which is defined from g and h by means of the schema $f(\overrightarrow{x}, 0) := g(\overrightarrow{x})$ and
$$f(\overrightarrow{x}, y + 1) := h(f(\overrightarrow{x}, y), \overrightarrow{x}, y).$$

Summary on Primitive Recursive Functions

Some negative results is known about reversible classes of total functions:

- PR bijections do not include all total computable reversible functions.
- PR bijections are not closed under inversion.
- The class of all computable bijections cannot be recursively enumerated.

SRL Genesis

- In 1968 Dennis Ritchie in his doctoral thesis "Program Structure and Computational Complexity" proposed the LOOP language (an old-fashion FOR language). LOOP is complete w.r.t. to primitive recursive functions.

- In this paper we focus our attention on SRL and its variants, namely a family of total reversible programming languages introduced in 2003 by Armando Matos conceived as a restriction of LOOP.

- The main difference between SRL languages and LOOP languages is that their registers store (positive and negative) integers.

$$\mathtt{P} \ ::= \mathsf{inc}\,\varkappa \mid \mathsf{dec}\,\varkappa \mid \underbrace{\mathsf{for}\,\varkappa\,(P)}_{\text{in SRL: } \varkappa \notin P} \mid P; P$$

Questions about SRL

Many questions have been posed about the expressivity of SRL.

1. Is the program equivalence of SRL decidable?
2. Is it decidable if a program of SRL behaves as the identity?
3. Is decidable whether a given program is an inverse of a second one?
4. Is SRL primitive-recursive complete?
5. Is SRL sufficiently expressive to represent RPP (or RPRF)?

In this work we answer to all them, by showing that:

> "a choice-operator can be implemented in SRL."

Test-for-Zero

- A Truth Values is represented by two-ordered registers r_t, r_f:
  - **true** is represented by $r_t, r_f \leftarrow 1, 0$;
  - **false** is represented by $r_t, r_t \leftarrow 0, 1$.

Test-for-Zero

- A Truth Values is represented by two-ordered registers r_t, r_f:
  - **true** is represented by $r_t, r_f \leftarrow 1, 0$;
  - **false** is represented by $r_t, r_t \leftarrow 0, 1$.

- $\texttt{if}\ (r_t == 1 \wedge r_f == 0)\ \texttt{then}\ P_0\ \texttt{else}\ P_1$
  can be simulated by for $r_t\ (P_0)$; for $r_f\ (P_1)$

Test-for-Zero

- A Truth Values is represented by two-ordered registers r_t, r_f:
  - **true** is represented by $r_t, r_f \leftarrow 1, 0$;
  - **false** is represented by $r_t, r_t \leftarrow 0, 1$.

- $\texttt{if } (r_t == 1 \wedge r_f == 0) \texttt{ then } P_0 \texttt{ else } P_1$
  can be simulated by for r_t (P_0); for r_f (P_1)

- Therefore, we need a test-for-zero.
  If R is a register and r_t, r_f form a truth-pair initialized to $0, 1$,
  then we are looking for an operator such that:
  - if $R \neq 0$ then all registers are unchanged after the test;
  - if $R = 0$ then all registers are unchanged, but r_t, r_f which are swapped.

Test-for-Zero

- To decide the parity is easy:

$$\begin{vmatrix} n \\ 1 \\ 0 \\ 0 \end{vmatrix} \ \mathsf{for}\, r_0(\mathsf{swap}(r_1, r_2); \mathsf{for}\, r_1(\mathsf{inc}\, r_3)); \ \begin{vmatrix} n \\ b_{\mathit{even}} \\ b_{\mathit{odd}} \\ n^{\bullet}/2 \end{vmatrix}$$

- The *Fundamental Theorem of Arithmetic* :

  each $n \neq 0$ admits a unique decomposition (up to the order of its factors)

$$(\pm 1)2^k p_1 p_2 \cdots p_m$$

  where $k \geq 0$ and, each p_i is a (positive) odd-prime number.

Test-for-Zero

Procedure `isLessThanOne`

Let r_2, r_3 and r_5, r_6 be truth-pairs initialized to true and let r_4 be a zero-ancilla. Let both r_0 and r_1 contain the value N. Then:

$$\mathsf{for}\, r_0 \left(\begin{array}{ll} \mathsf{for}\, r_5(\mathsf{for}\, r_1(\ \mathsf{swap}(r_2, r_3);\ \mathsf{for}\, r_2(\mathsf{inc}\, r_4)\ ));\quad & \texttt{/* SP0 */} \\ \mathsf{for}\, r_3(\mathsf{swap}(r_5, r_6)); & \texttt{/* SP1 */} \\ \mathsf{for}\, r_5(\ \mathsf{for}\, r_4(\mathsf{dec}\, r_1);\ \mathsf{for}\, r_1(\mathsf{dec}\, r_4)\ ); & \texttt{/* SP2 */} \\ \mathsf{for}\, r_6 \left(\begin{array}{l} \mathsf{for}\, r_1(\ \mathsf{for}\, r_2(\mathsf{dec}\, r_4);\ \mathsf{swap}(r_2, r_3)\ ); \\ \mathsf{for}\, r_1(\mathsf{inc}\, r_4);\ \mathsf{for}\, r_4(\mathsf{inc}\, r_1) \end{array} \right) & \texttt{/* SP3 */} \end{array} \right)$$

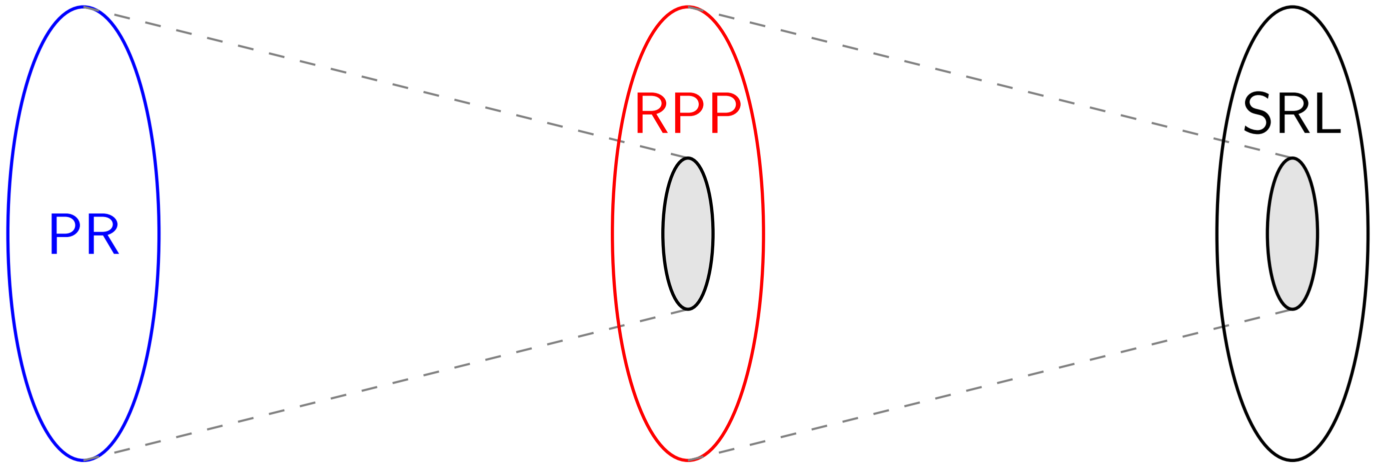
leaves true in the truth-pair r_5, r_6 if and only if N is strictly lower than 1.

RPP is a sub-class of endofunctions on $\mathbb{Z}^n$ for some $n \in \mathbb{N}$.

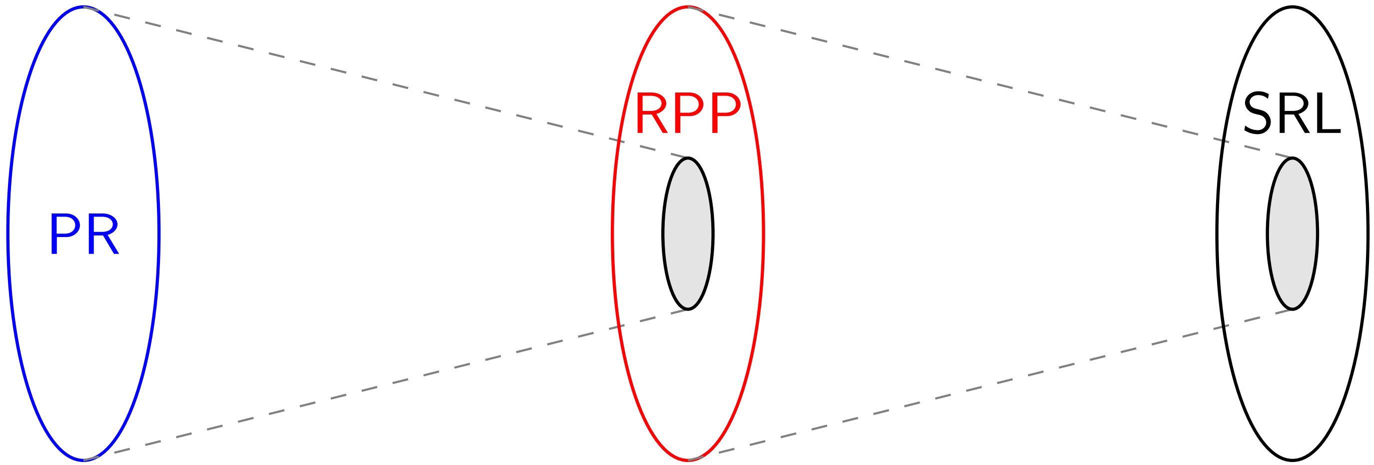
1. RPP^1 includes successor, predecessor negation
2. RPP^2 includes the swap
3. If $f, g \in \mathsf{RPP}^k$ then, RPP^k includes their series-composition
4. If $f \in \mathsf{RPP}^j$ and $g \in \mathsf{RPP}^k$, then RPP^{j+k} includes their parallel composition
5. If $f \in \mathsf{RPP}^k$, then the finite iteration $\mathsf{It}\,[f]$ belongs to RPP^{k+1}
6. Let $f, g, h \in \mathsf{RPP}^k$. The selection $\mathsf{If}\,[f, g, h]$ belongs to RPP^{k+1} and it is the function defined as:

$$\mathsf{If}\,[f, g, h]\,(\langle x_1, \ldots, x_k, z\rangle) := \begin{cases} (f \parallel \mathsf{Id})\,(\langle x_1, \ldots, x_k, z\rangle) & \text{if } z > 0\ , \\ (g \parallel \mathsf{Id})\,(\langle x_1, \ldots, x_k, z\rangle) & \text{if } z = 0\ , \\ (h \parallel \mathsf{Id})\,(\langle x_1, \ldots, x_k, z\rangle) & \text{if } z < 0\ . \end{cases}$$

Conclusions

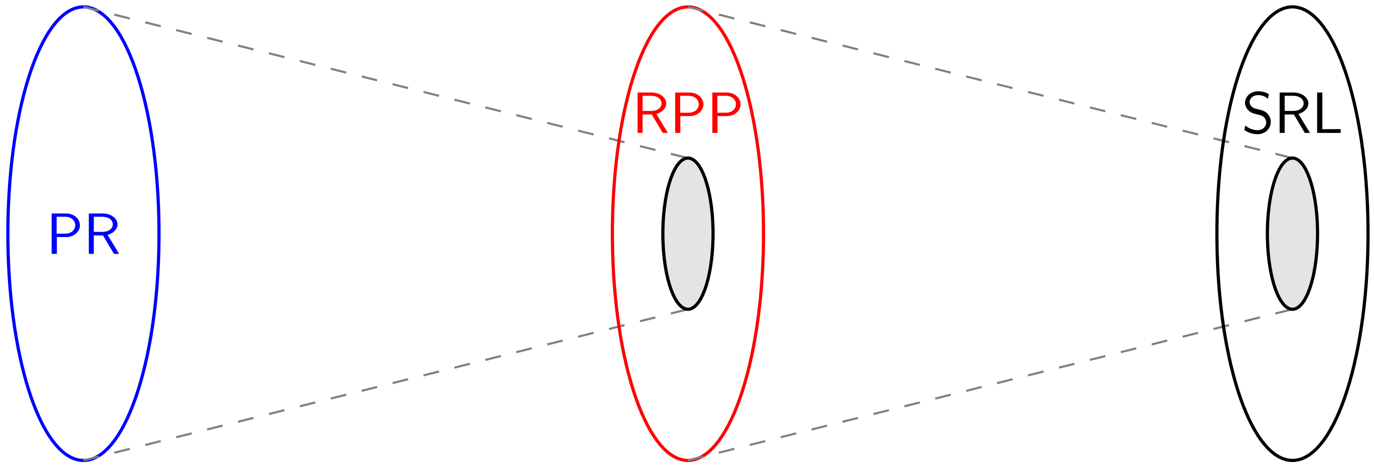


Conclusions



1. Is the program equivalence of SRL decidable?
2. Is it decidable if a program of SRL behaves as the identity?
3. Is decidable whether a given program is an inverse of a second one?
4. Is SRL primitive-recursive complete?
5. Is SRL sufficiently expressive to represent RPP (or RPRF)?

Conclusions



1. Is the program equivalence of SRL decidable? ☒
2. Is it decidable if a program of SRL behaves as the identity? ☒
3. Is decidable whether a given program is an inverse of a second one? ☒
4. Is SRL primitive-recursive complete? ☑
5. Is SRL sufficiently expressive to represent RPP (or RPRF)? ☑

Future Works

- ☐ Turing-complete extensions
- ☐ Kleene's theorems, ...
- ☐ Complexity Hierarchies, ...

End ...

thank you!