Computationally Complete Spiking Neural P Systems Without Delay: Two Types of Neurons Are Enough

Rudolf Freund[1], Marian Kogler[1,2]

[1] Faculty of Informatics, Vienna University of Technology, Austria
[2] Institute of Computer Science, Martin Luther University Halle-Wittenberg, Germany
Email: rudi@emcc.at, marian@emcc.at, kogler@informatik.uni-halle.de

CMC11

Overview

Preliminaries

Results

Suggestions for Future Work

Register Machines

Definition (Register Machine)

A register machine is a construct

$$M = (n, B, p_0, p_h, I)$$

where

1. n, $n \geq 1$, is the number of registers,
2. B is the set of instruction labels,
3. p_0 is the start label,
4. p_h is the halting label (only used for the `HALT` instruction) and
5. I is a set of (labeled) instructions.

Register Machines

Definition (Register Machine (ctd.))

Instructions are of the following forms:

- $p_i : (\texttt{ADD}(r), p_j, p_k)$ increments the value in register r and continues with one of the instructions labeled by p_j and p_k, chosen in a nondeterministic way,
- $p_i : (\texttt{SUB}(r), p_j, p_k)$ tries to decrement the value in register r; if the register was non-empty before the instruction, the computation continues with the instruction labeled with p_j, if not, it continues with the instruction p_k;
- $p_h : \texttt{HALT}$ halts the machine.

Register Machines

Definition (Register Machine (ctd.))

- Deterministic register machines can be constructed by imposing the condition $p_j = p_k$ on `ADD`-instructions.
- We will be using nondeterministic register machines as generators and deterministic register machines as acceptors.
- Every recursively enumerable set of (vectors of) natural numbers with k components can be generated with only $k + 2$ registers, where the first k registers are never decremented (Minsky, 1967).

Spiking Neural P Systems (Without Delays)

Definition (Spiking neural P system)

A spiking neural P system (without delays) is a construct

$$\Pi = (O, \rho_1, ..., \rho_n, syn, in, out)$$

where

1. $O = \{a\}$ is the (unary) set of objects (the object a is called *spike*),
2. $\rho_1, ..., \rho_n$ are the neurons, where $\rho_i = (d_i, R_i)$ for $1 \leq i \leq n$, with d_i being the initial configuration of the neuron i and R_i being the set of rules,
3. *in* is the input neuron (with the only function to spike once in generating spiking neural P systems in order to start a computation), and
4. *out* is the output neuron (no function in accepting spiking neural P systems).

Spiking Neural P Systems (Without Delays)

Definition (Spiking neural P system (ctd.))

Possible forms for the rules:

- $E/a^i \rightarrow a^j$, where E is a regular expression over O and $i, j \geq 1$ (*firing rules*) or
- $a^i \rightarrow \lambda$, where $i \geq 1$ (*forgetting rules*). There must not be any rule $a^i \rightarrow \lambda$ such that $a^i \in L(E)$ for some E of a firing rule.
- $syn \subseteq \{1, ..., n\} \times \{1, ..., n\}$ are the synapses, where $(i, j) \in syn$ indicates a synapse from i to j,

Spiking Neural P Systems (Without Delays)

Definition (Spiking neural P system (ctd.))

- A spiking neural P system inputs and outputs numbers via a spike train. A spike train starts with a spike given in step t_1 and ends with a spike given in step t_2. The number is specified by $t_2 - t_1 - 1$, i.e., the number of steps that elapse between the two spikes. It accepts an input by a series of configurations, starting from the initial configuration and ending in a halting configuration.
- Rules of the form $E/a^i \rightarrow a^j$ where $L(E)$ is finite (infinite) are called *bounded (unbounded) rules*.
- Two neurons ρ_i and ρ_j are of the same type if and only if $R_i = R_j$, $d_i = d_j$ and $|\{(i,k) \in sym \mid k \in \{1,...,n\}\}| = |\{(j,k) \in sym \mid k \in \{1,...,n\}\}|$.

Results

- Accepting spiking neural P systems (without delays) with only two types of neurons are computationally complete.
- Generating spiking neural P systems without delays with only two types of neurons are computationally complete.
- Corollary: Spiking neural P systems without delays with only three neurons with unbounded rules are computationally complete.

Results

- Accepting spiking neural P systems (without delays) with only two types of neurons are computationally complete.
- Generating spiking neural P systems without delays with only two types of neurons are computationally complete.
- Corollary: Spiking neural P systems without delays with only three neurons with unbounded rules are computationally complete.

The Two Types

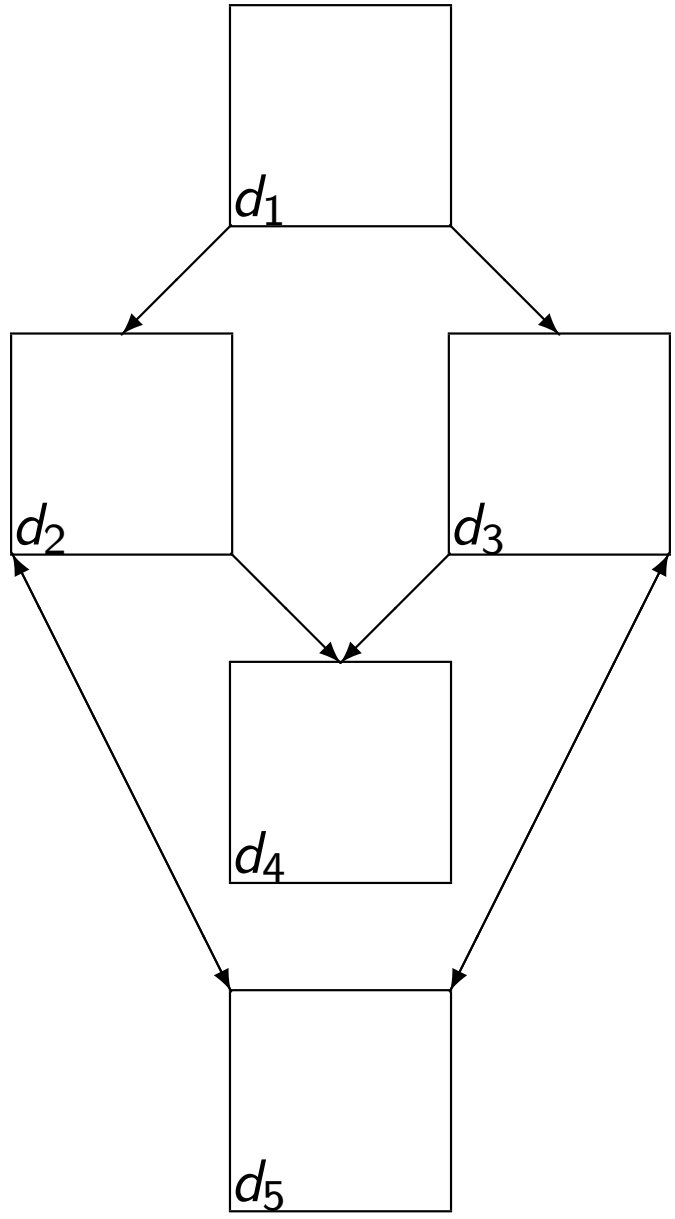
$$\lambda$$
$$a/a \longrightarrow a$$
$$a^2 \longrightarrow \lambda$$
$$a^3/a^3 \longrightarrow a$$
$$a^4/a^4 \longrightarrow a$$
$$a^5 \longrightarrow \lambda$$

Type 1

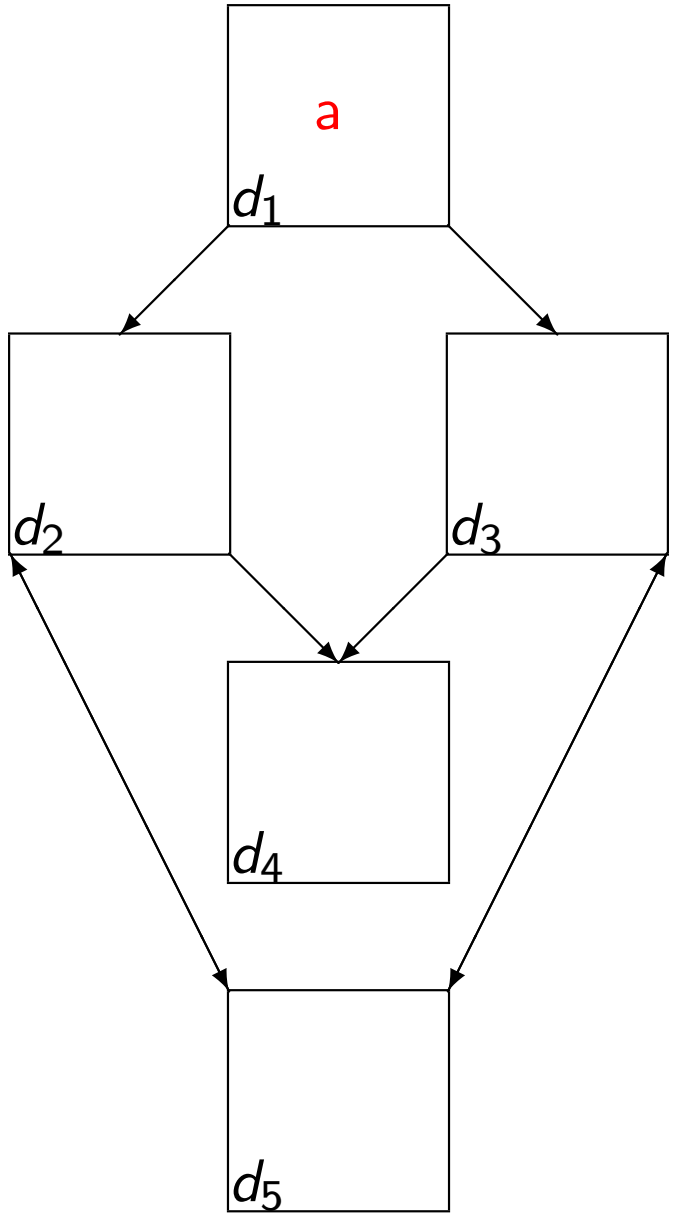
$$\lambda$$
$$a \longrightarrow \lambda$$
$$a^3(a^2)^*/a^3 \longrightarrow a$$

Type 2

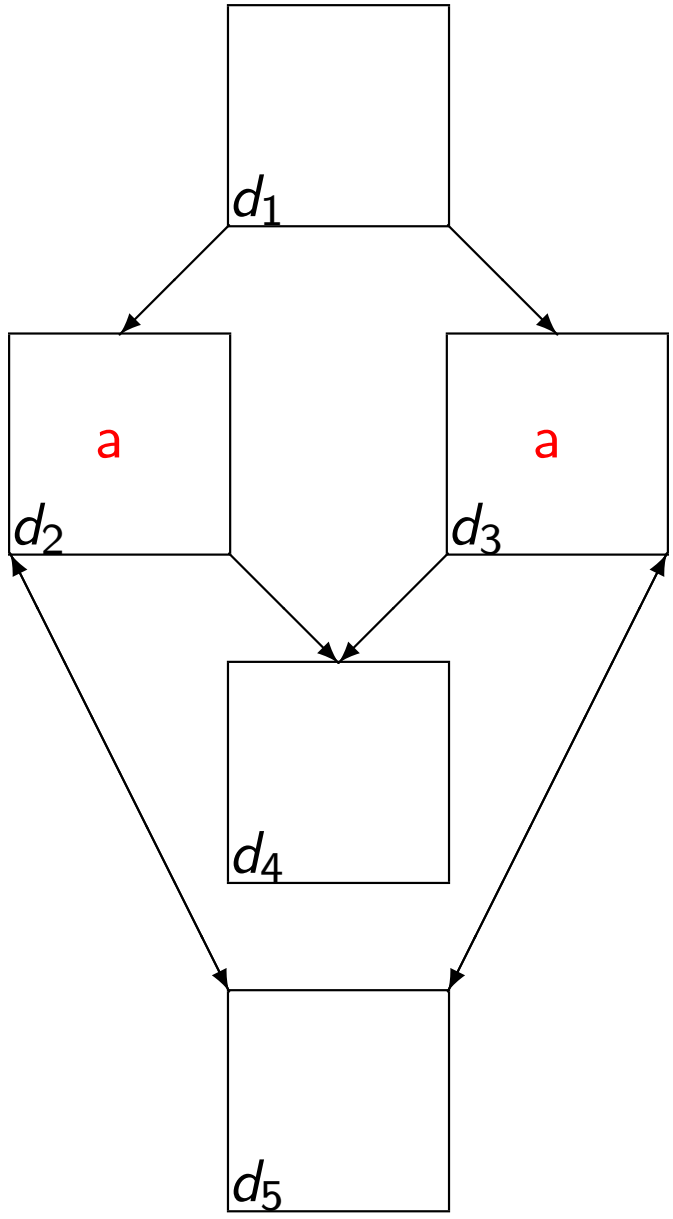
A Dummy Structure



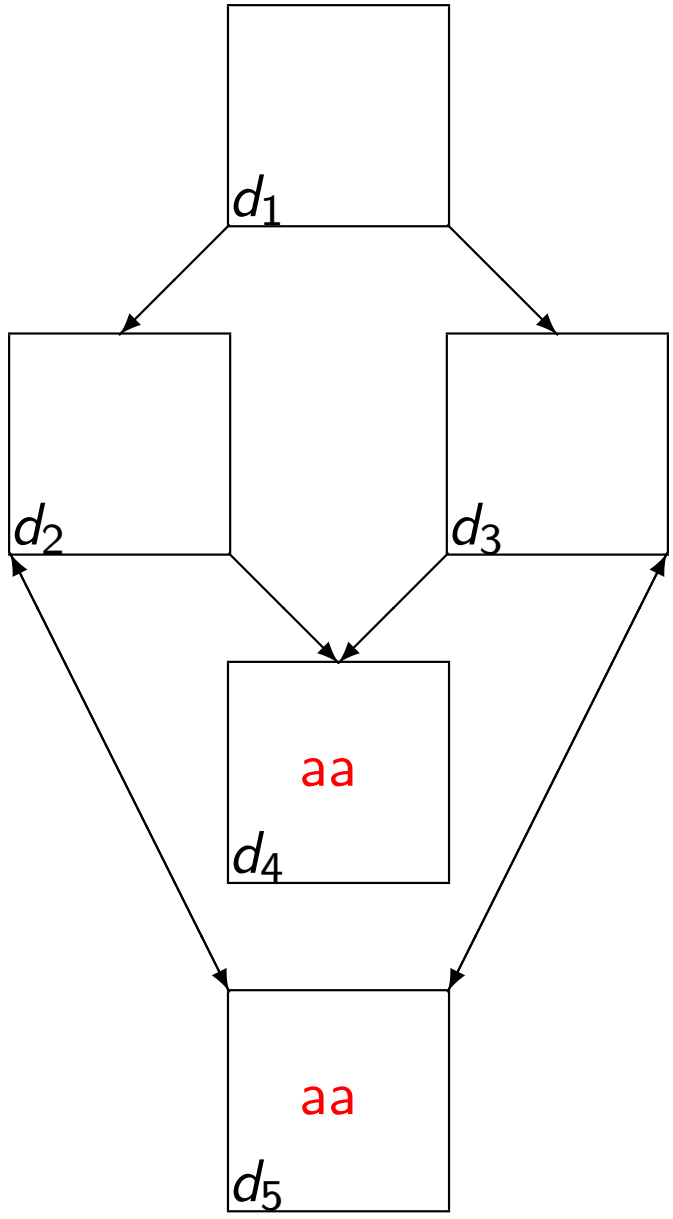
A Dummy Structure



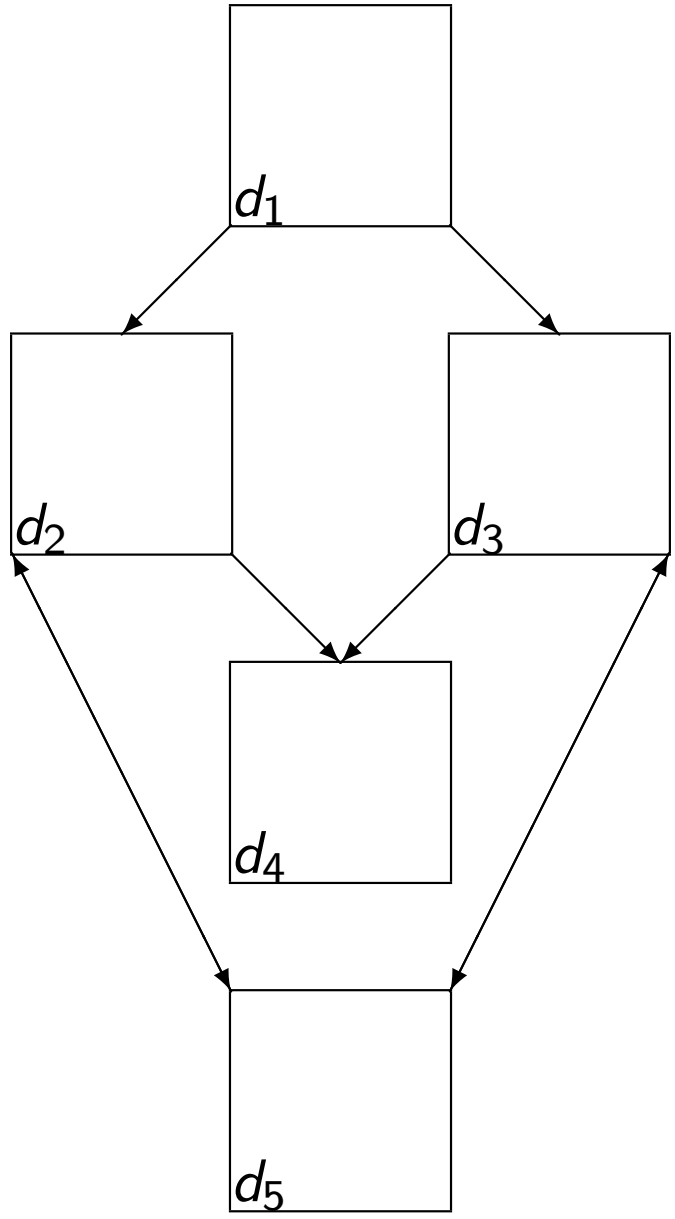
A Dummy Structure



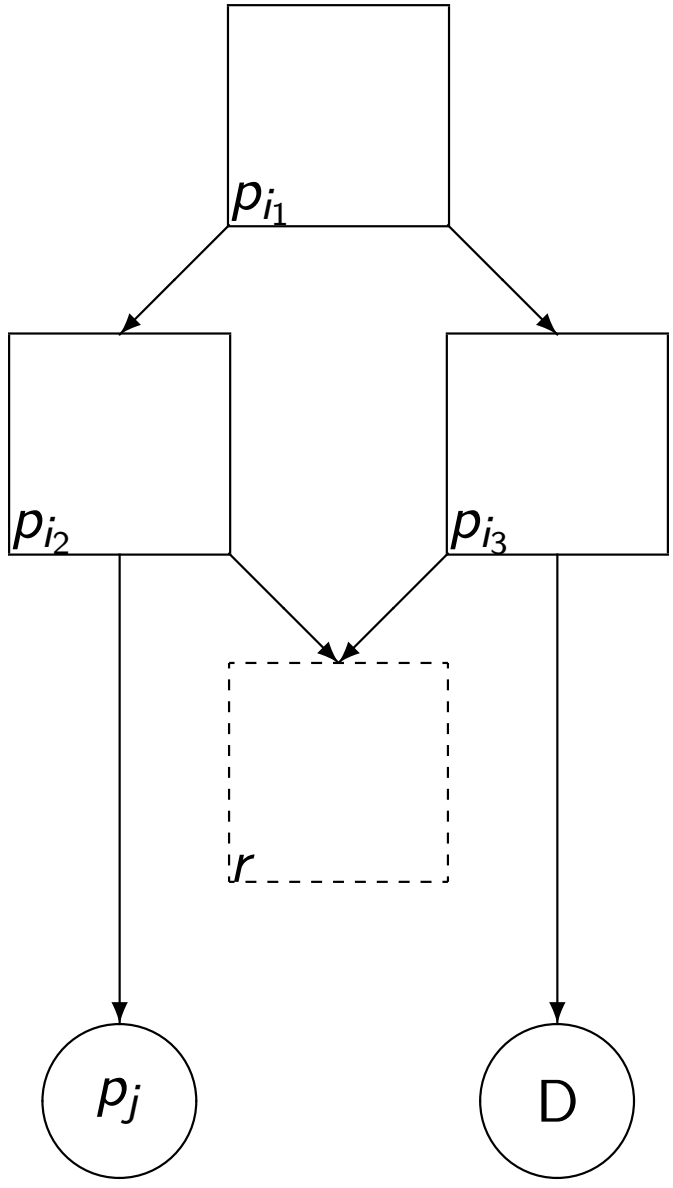
A Dummy Structure



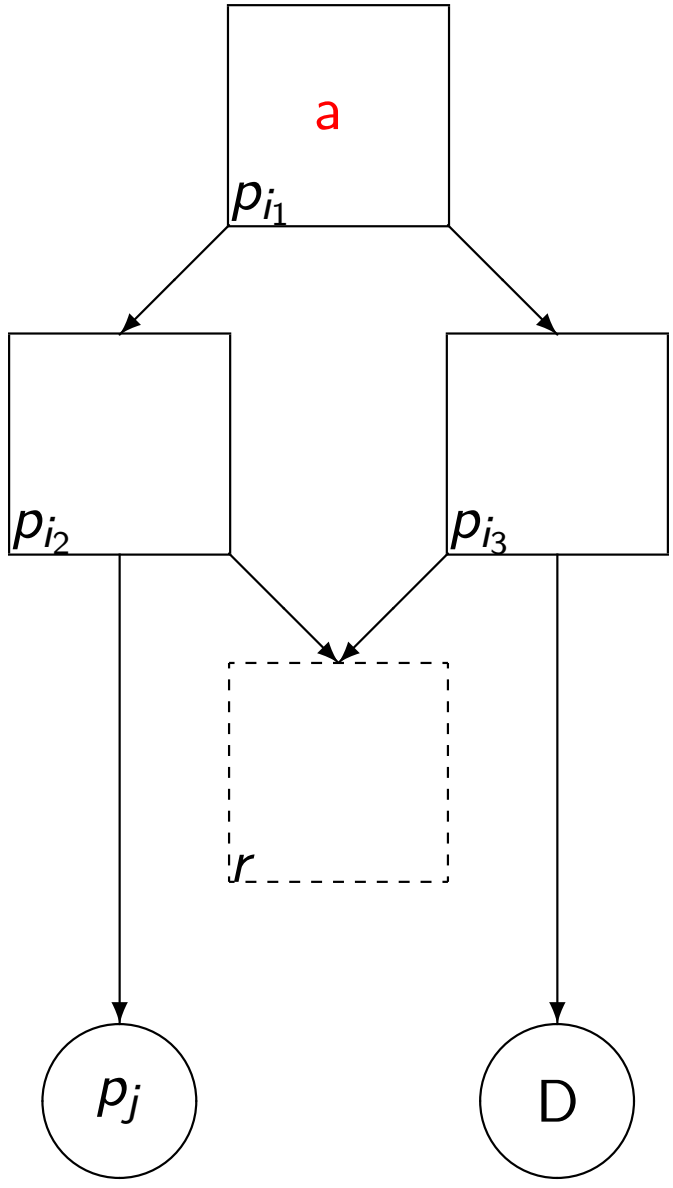
A Dummy Structure



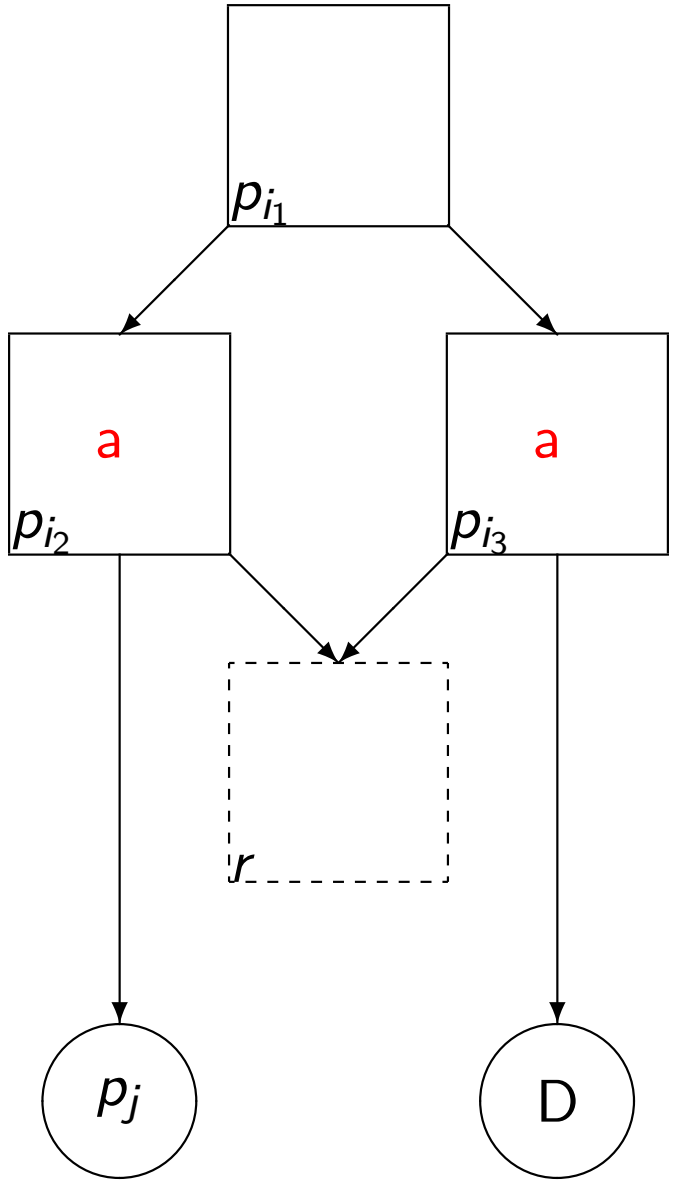
Simulating an ADD-instruction



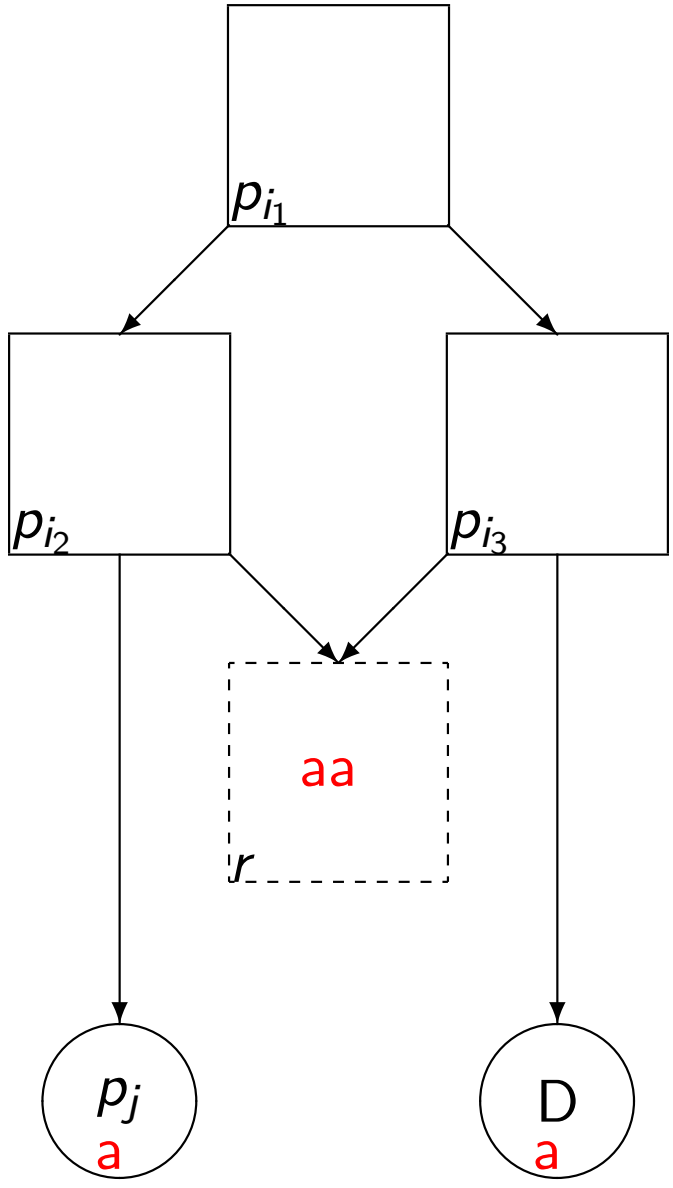
Simulating an ADD-instruction



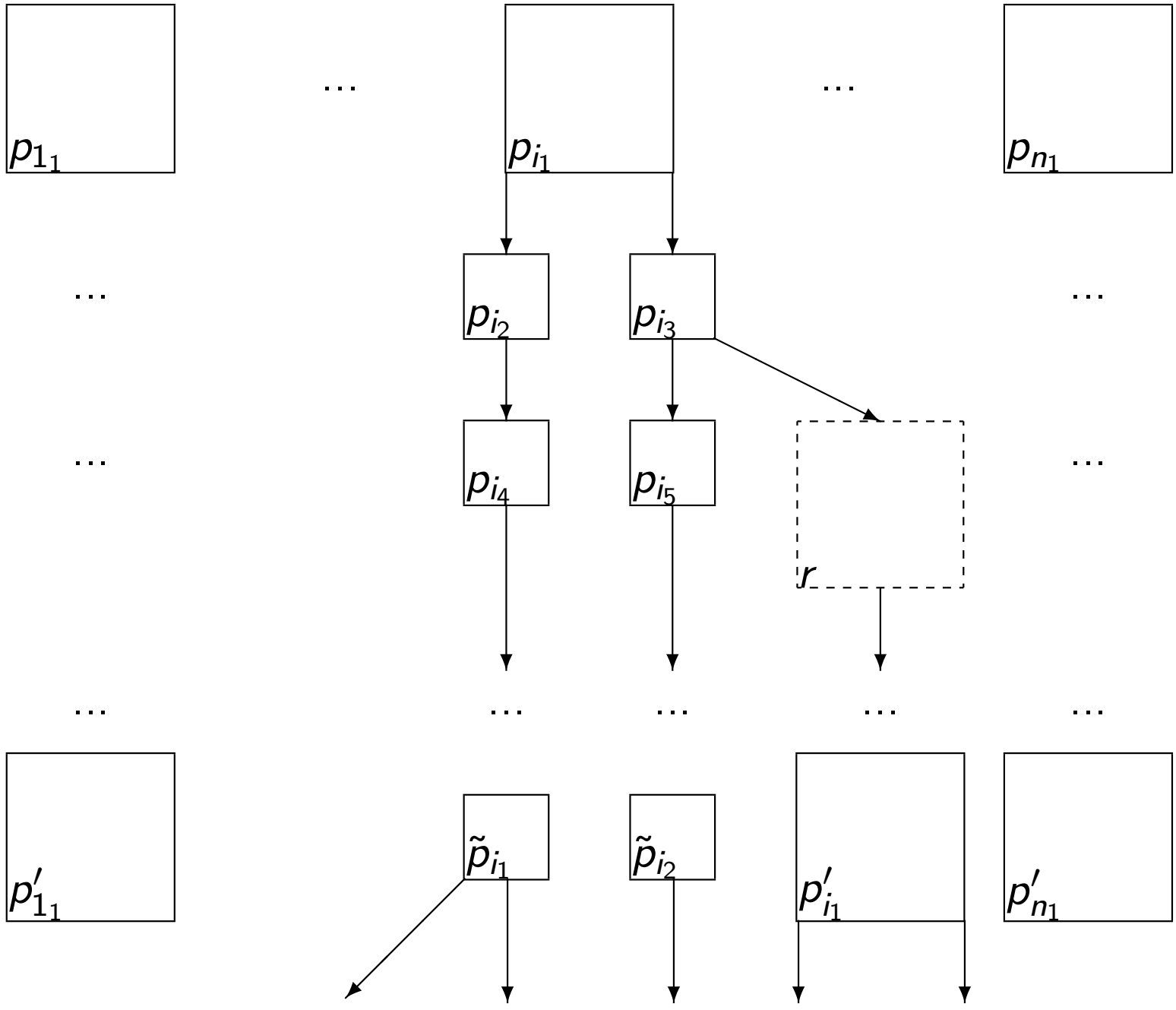
Simulating an ADD-instruction



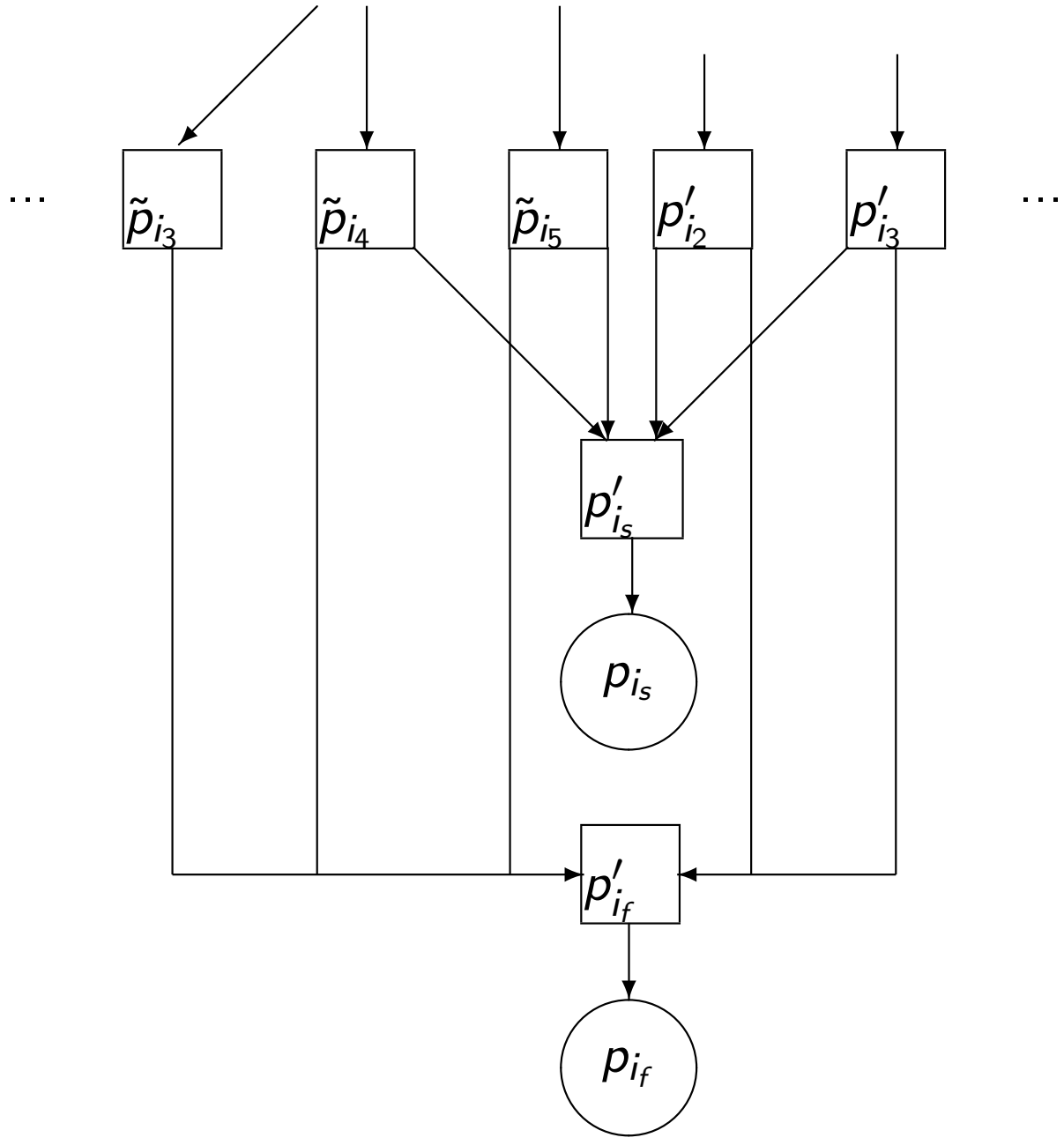
Simulating an ADD-instruction



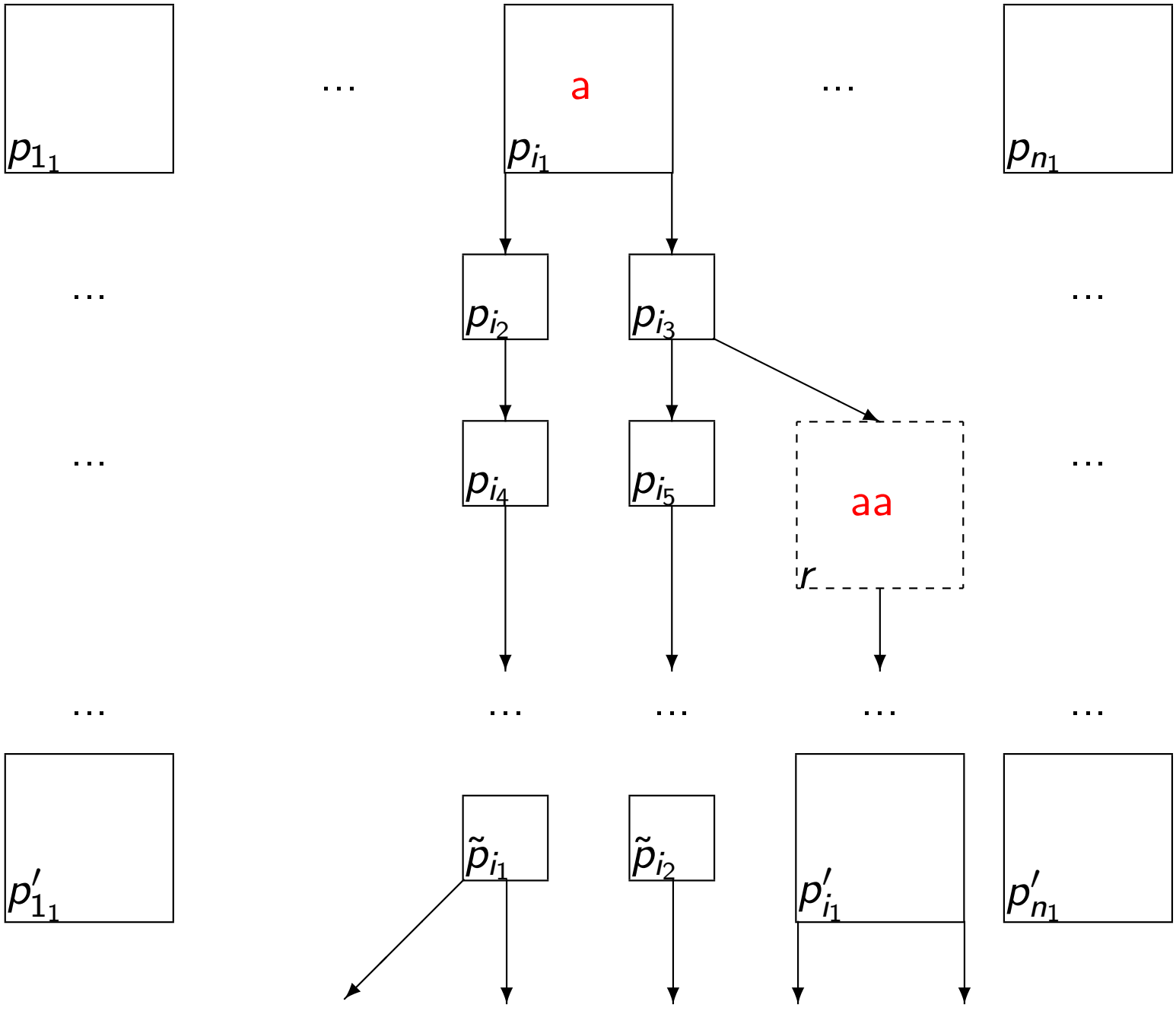
Simulating a SUB-instruction



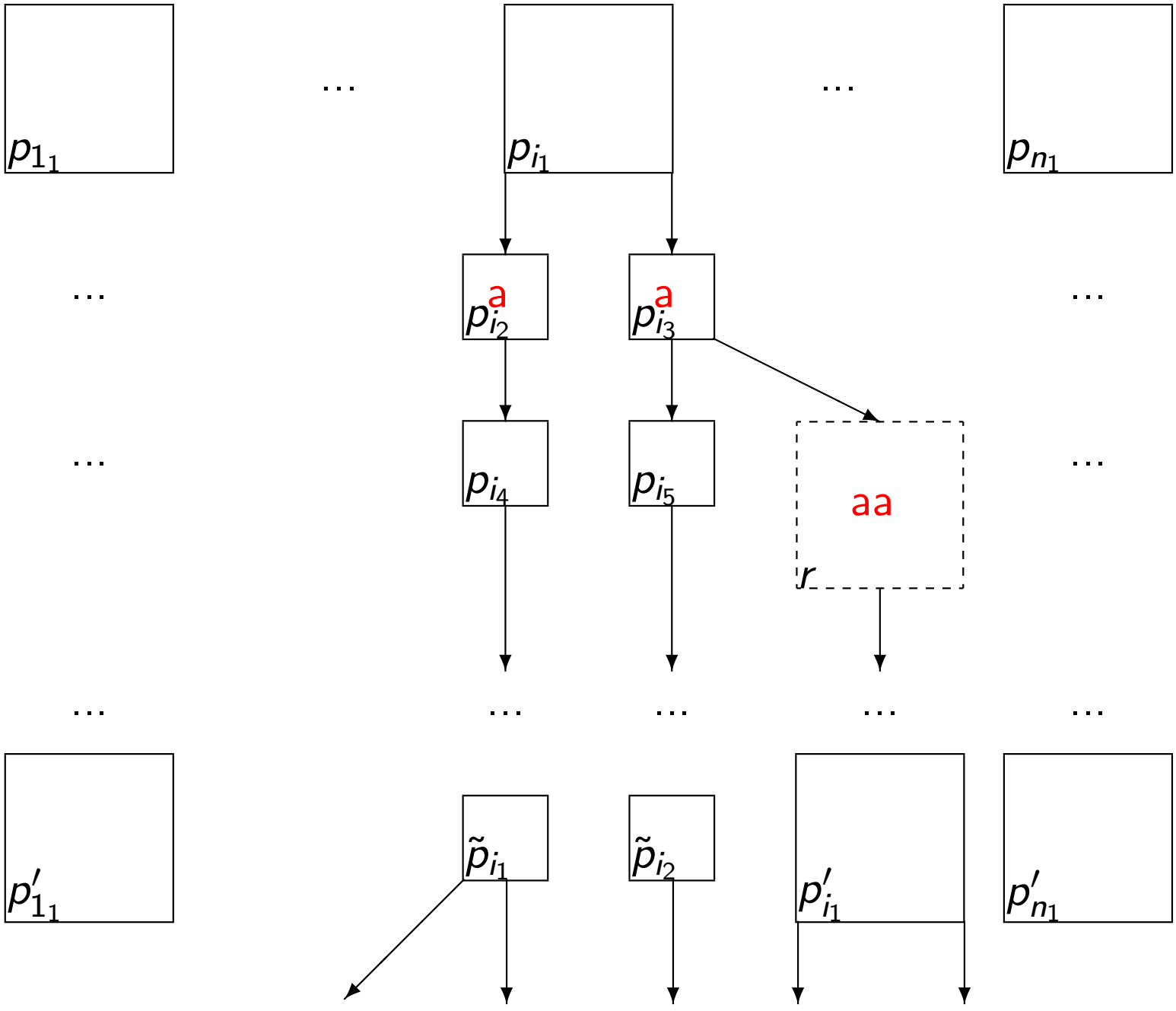
Simulating a SUB-instruction



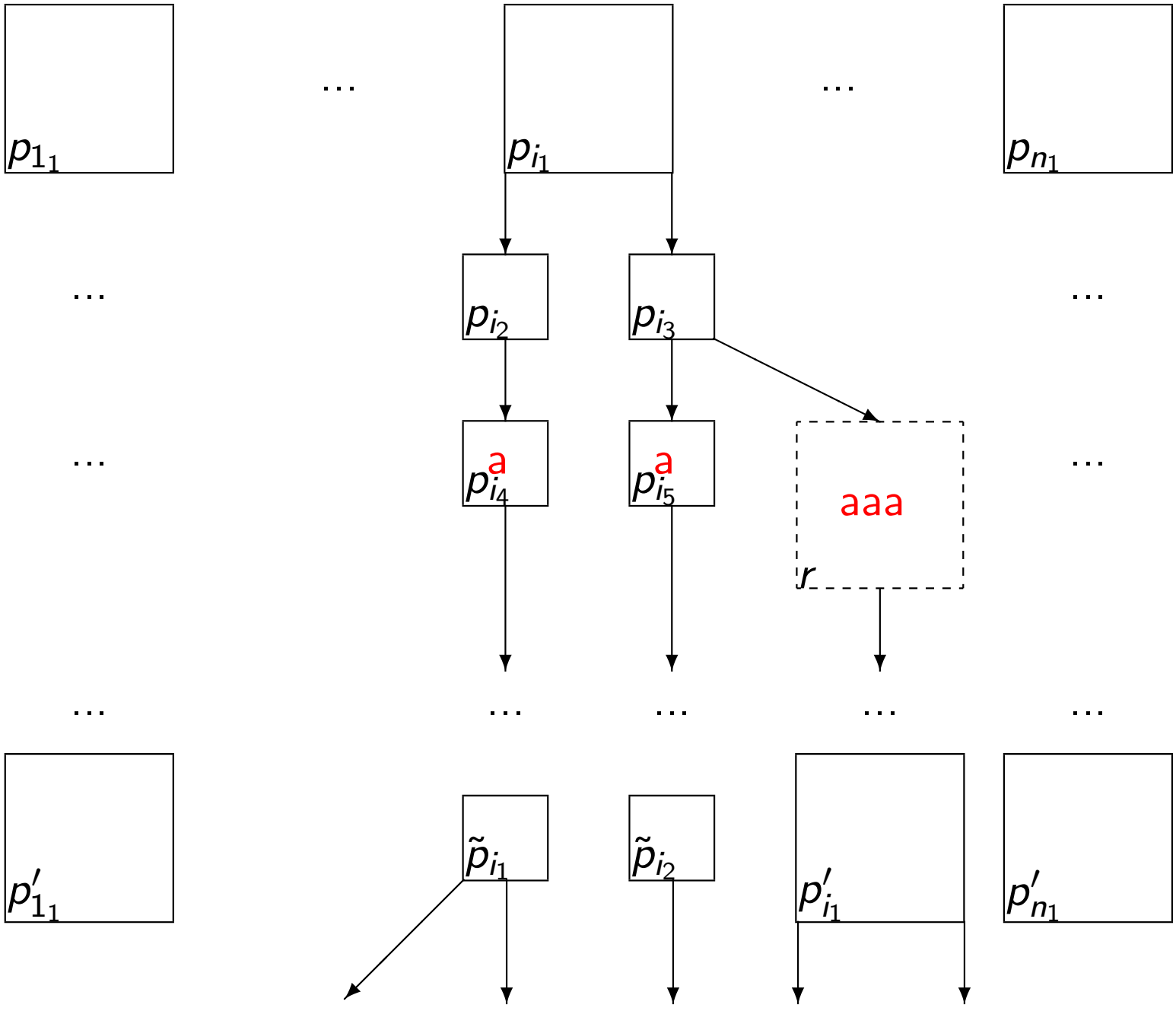
Simulating a SUB-instruction



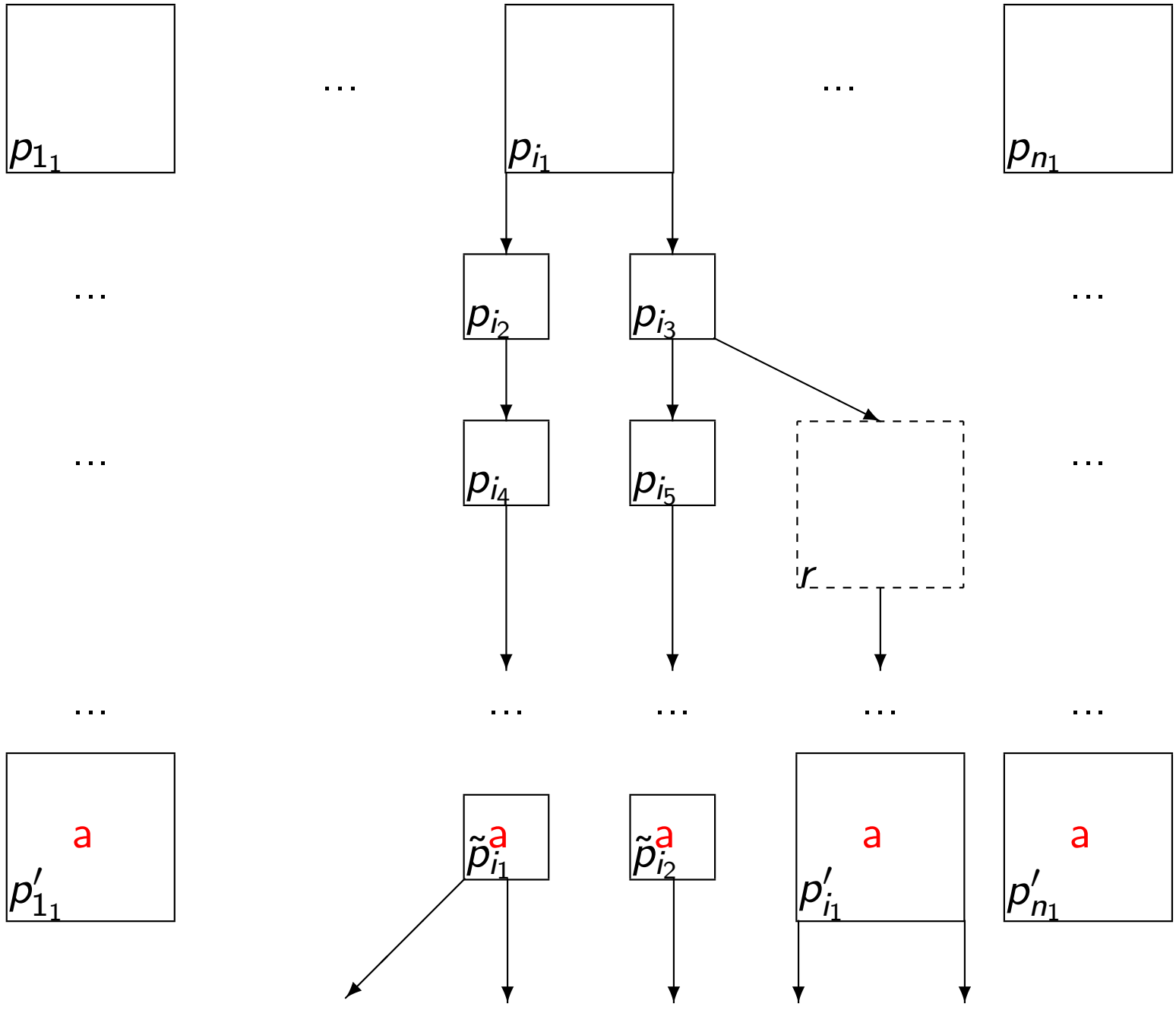
Simulating a SUB-instruction



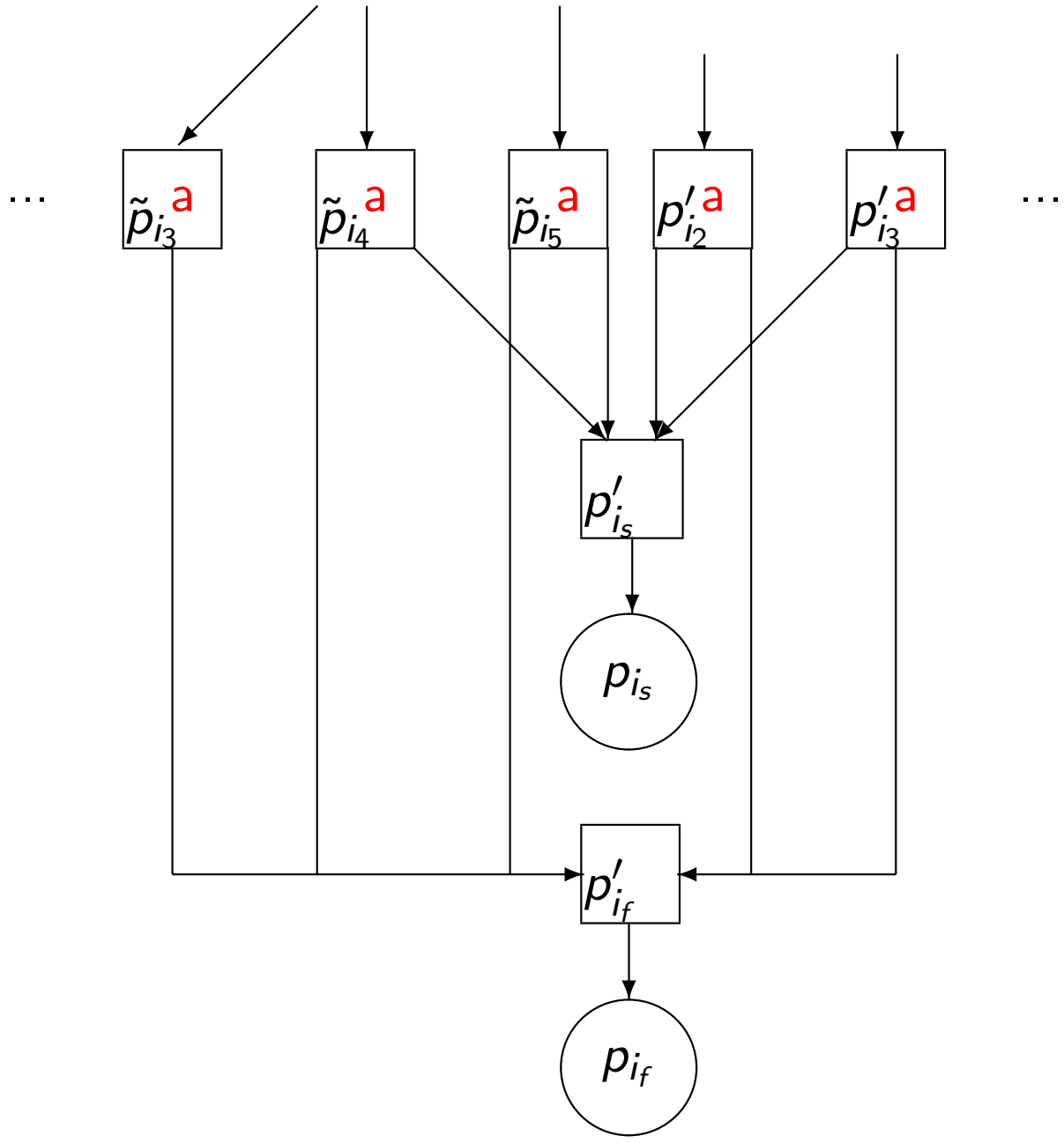
Simulating a SUB-instruction



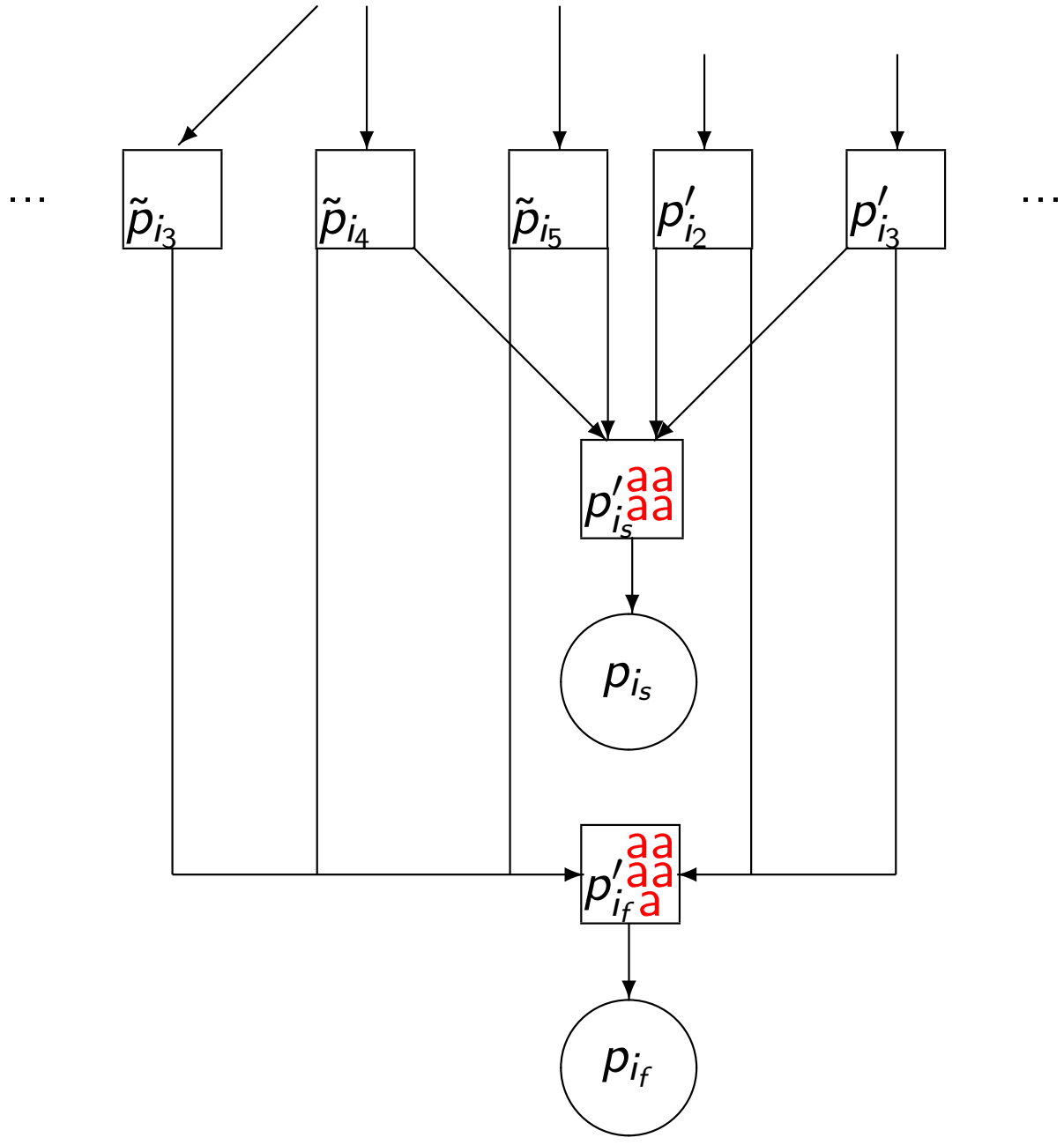
Simulating a SUB-instruction



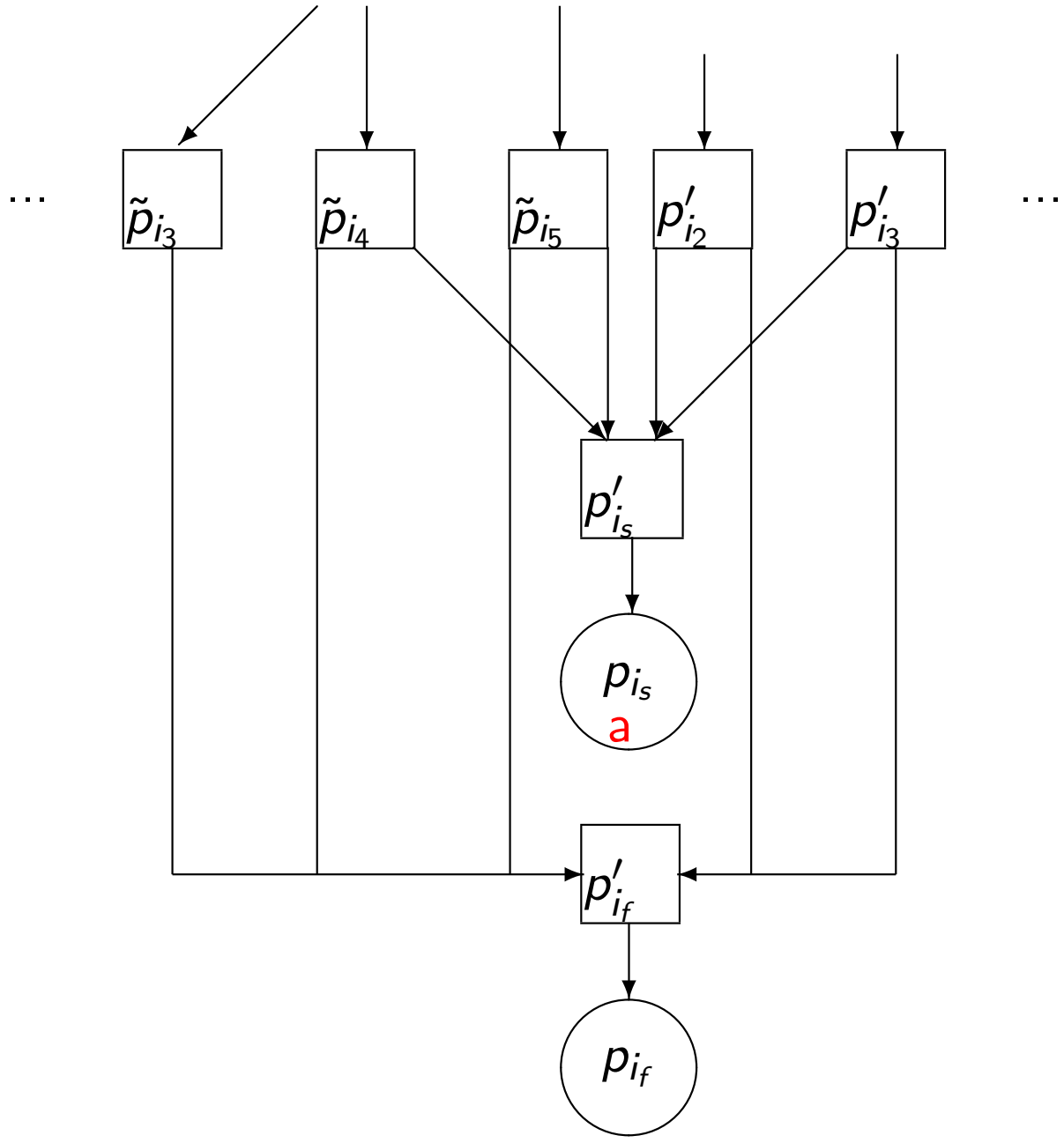
Simulating a SUB-instruction



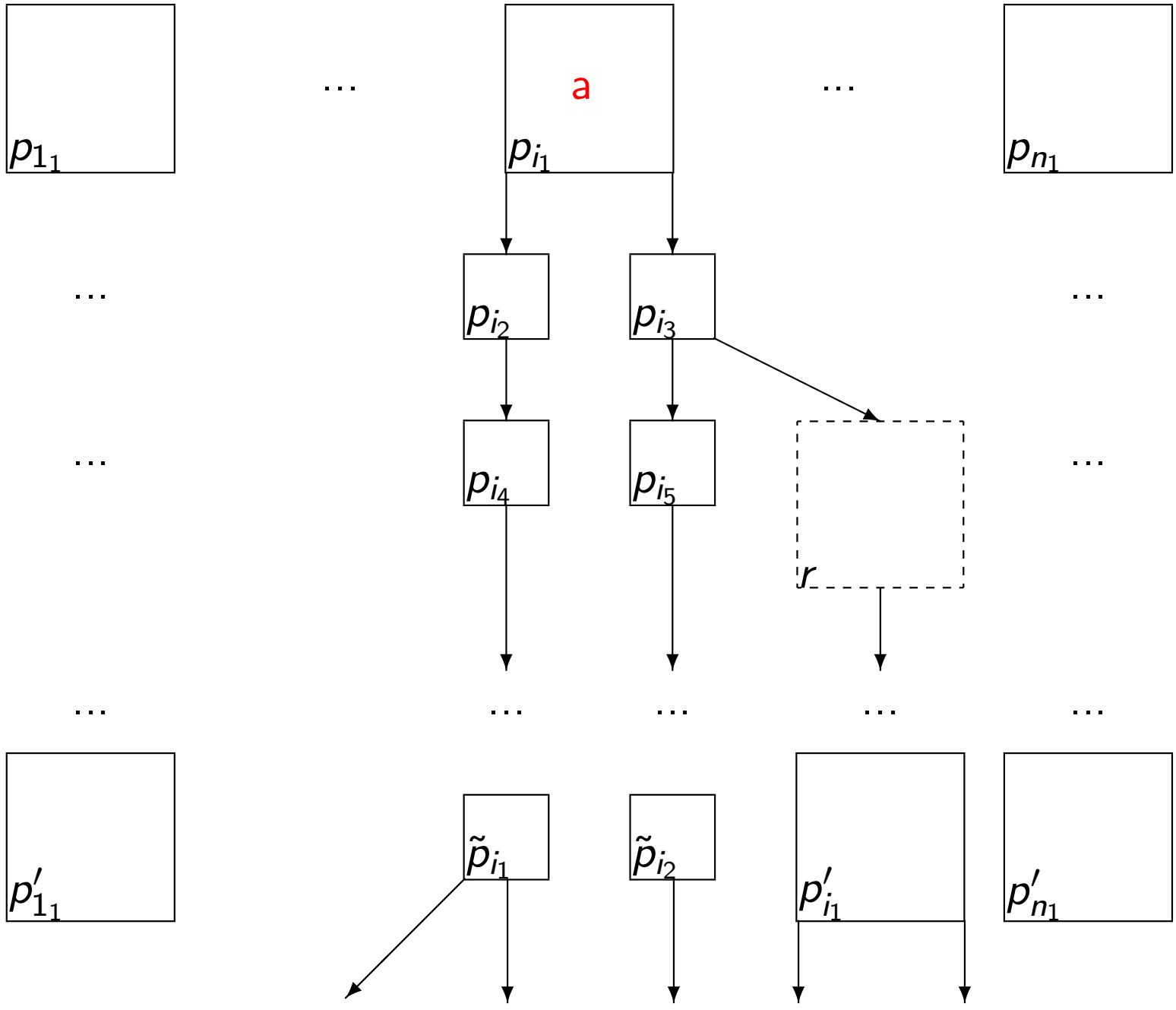
Simulating a SUB-instruction



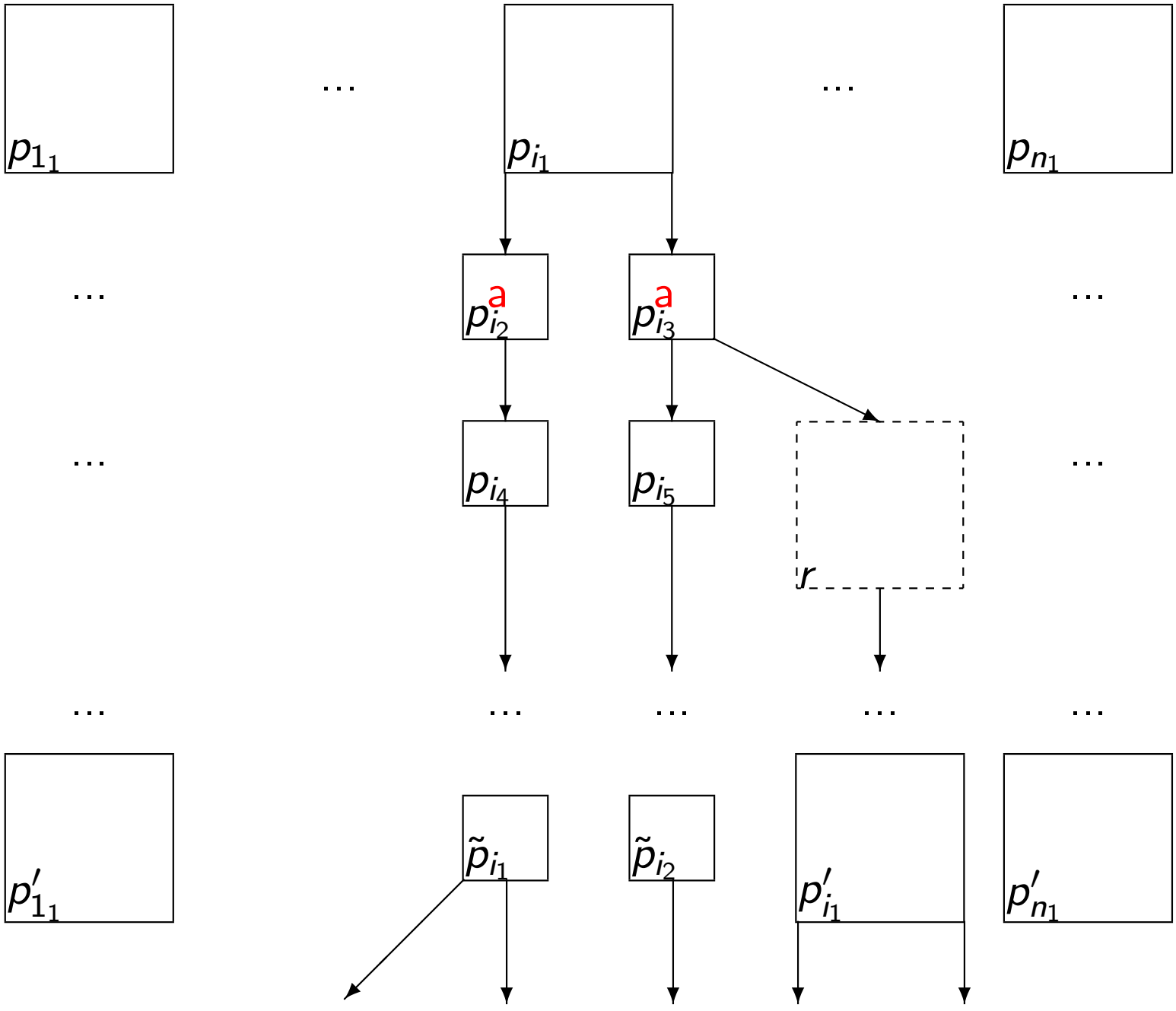
Simulating a SUB-instruction



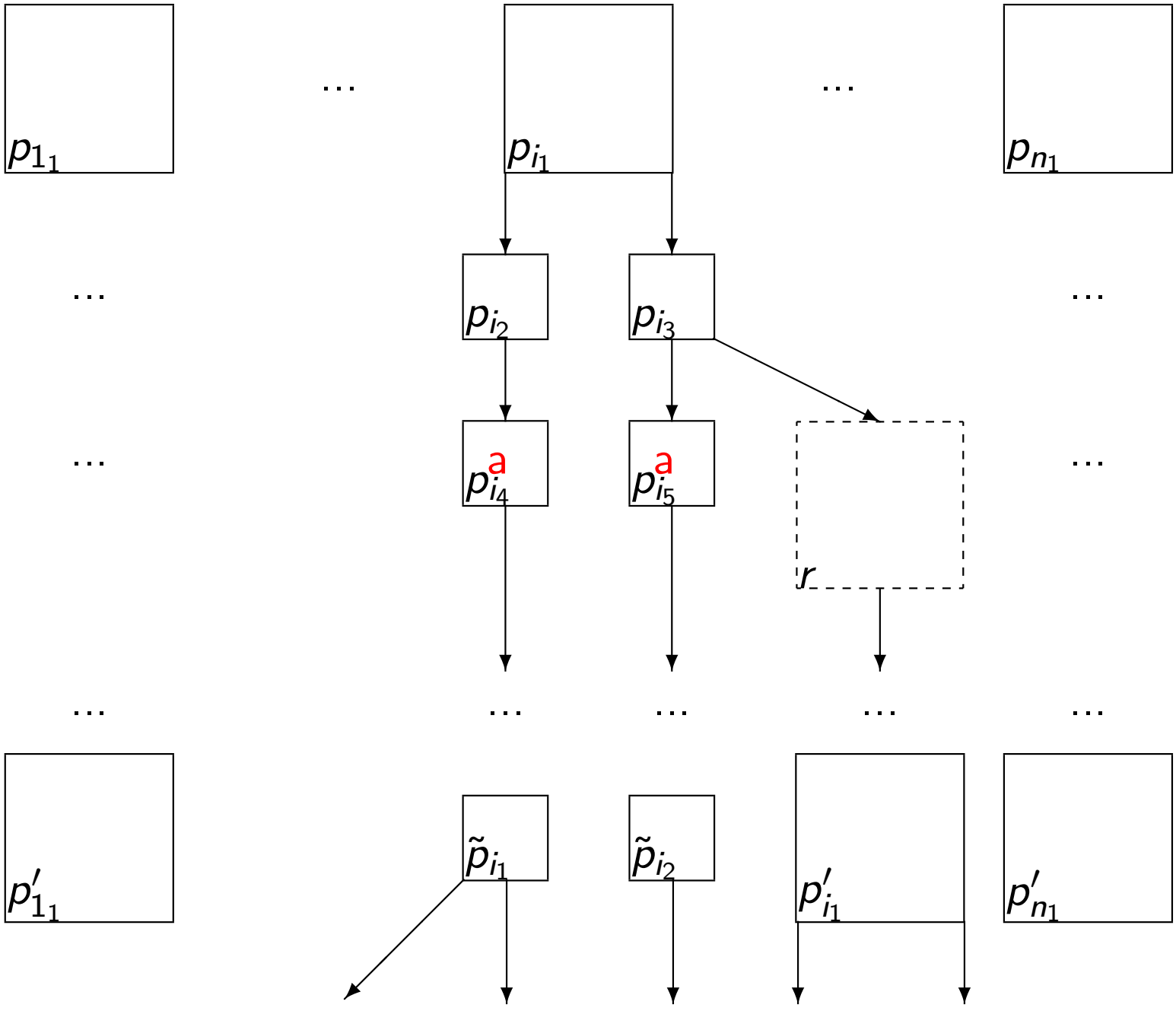
Simulating a SUB-instruction



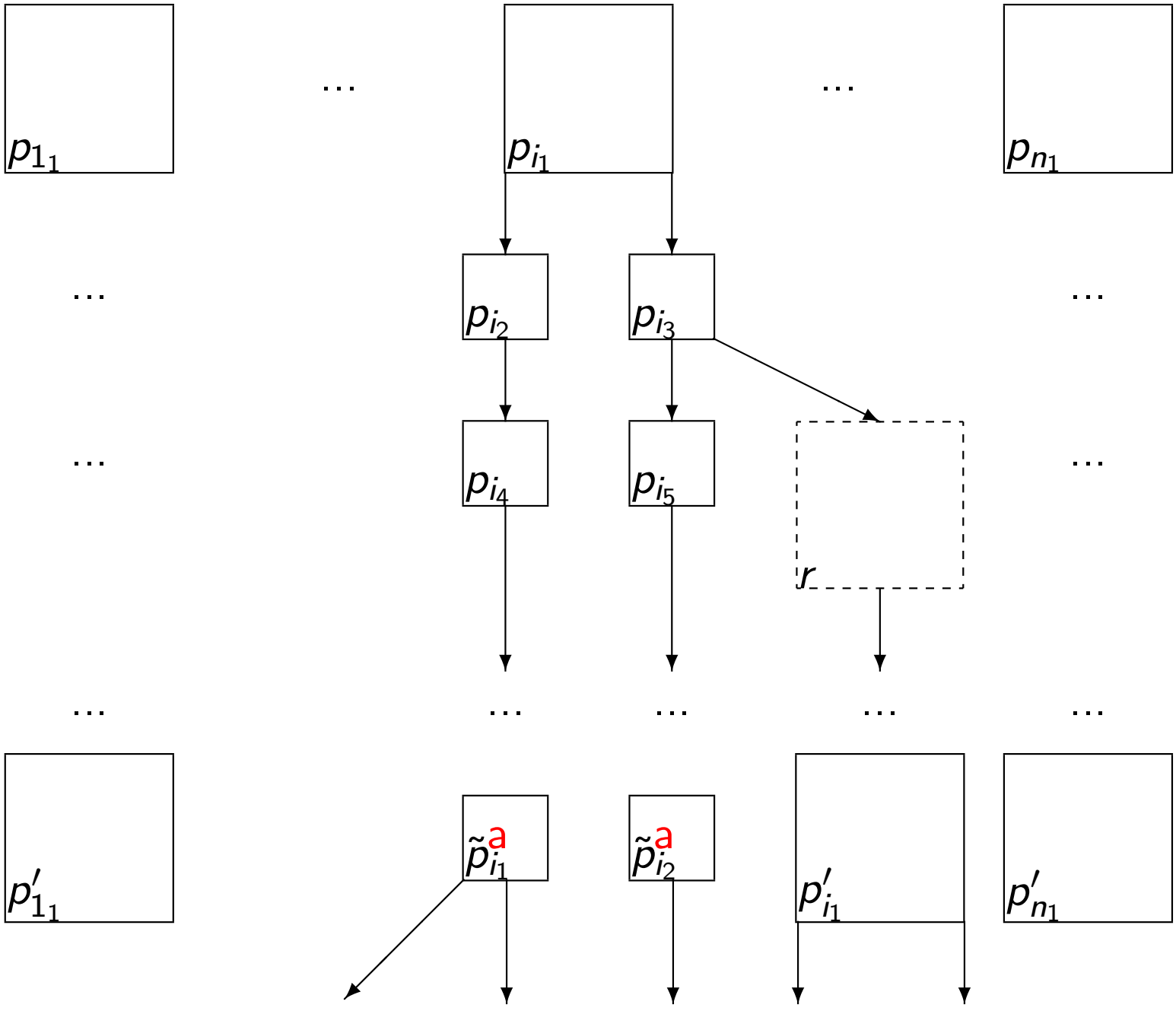
Simulating a SUB-instruction



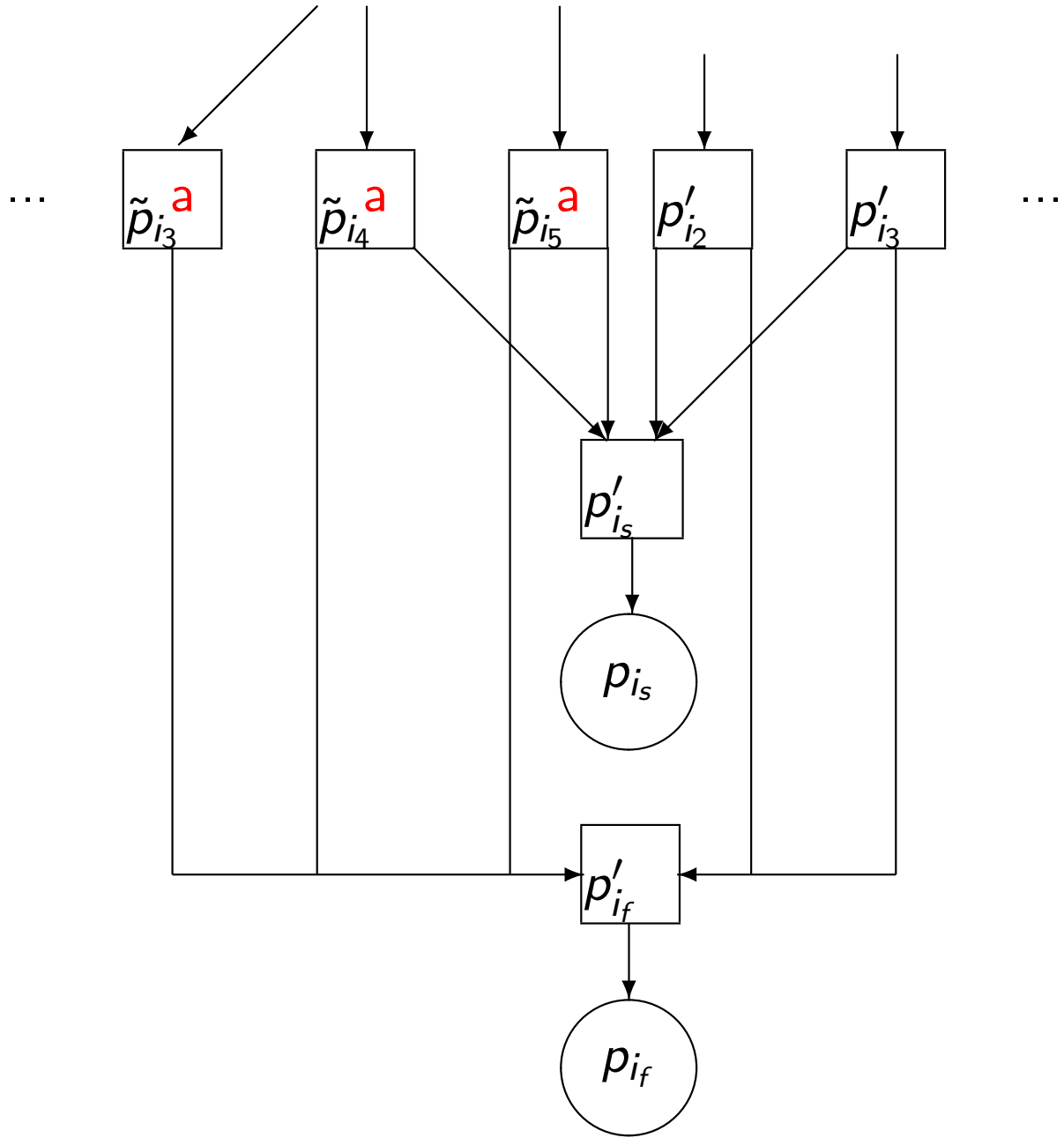
Simulating a SUB-instruction



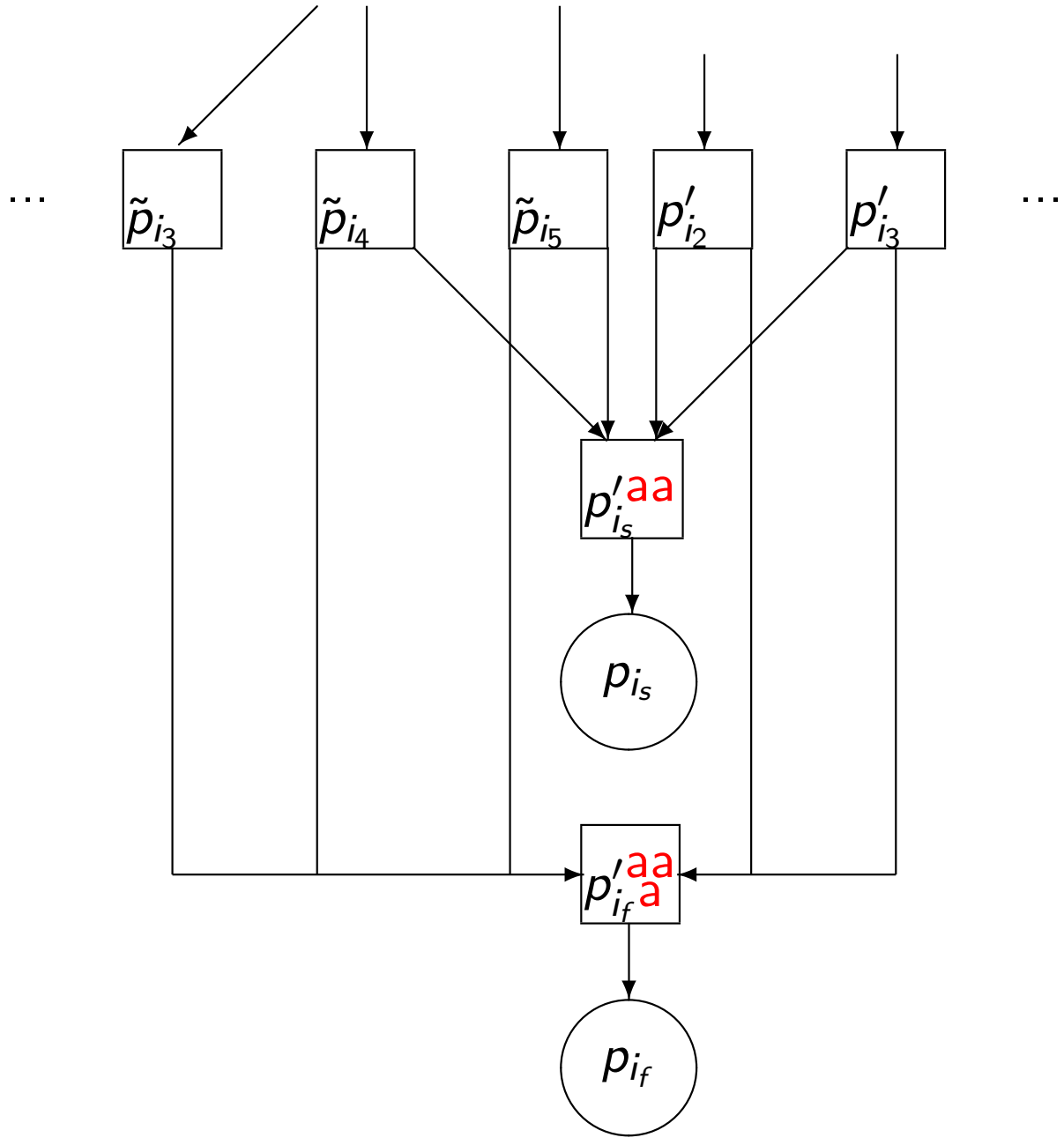
Simulating a SUB-instruction



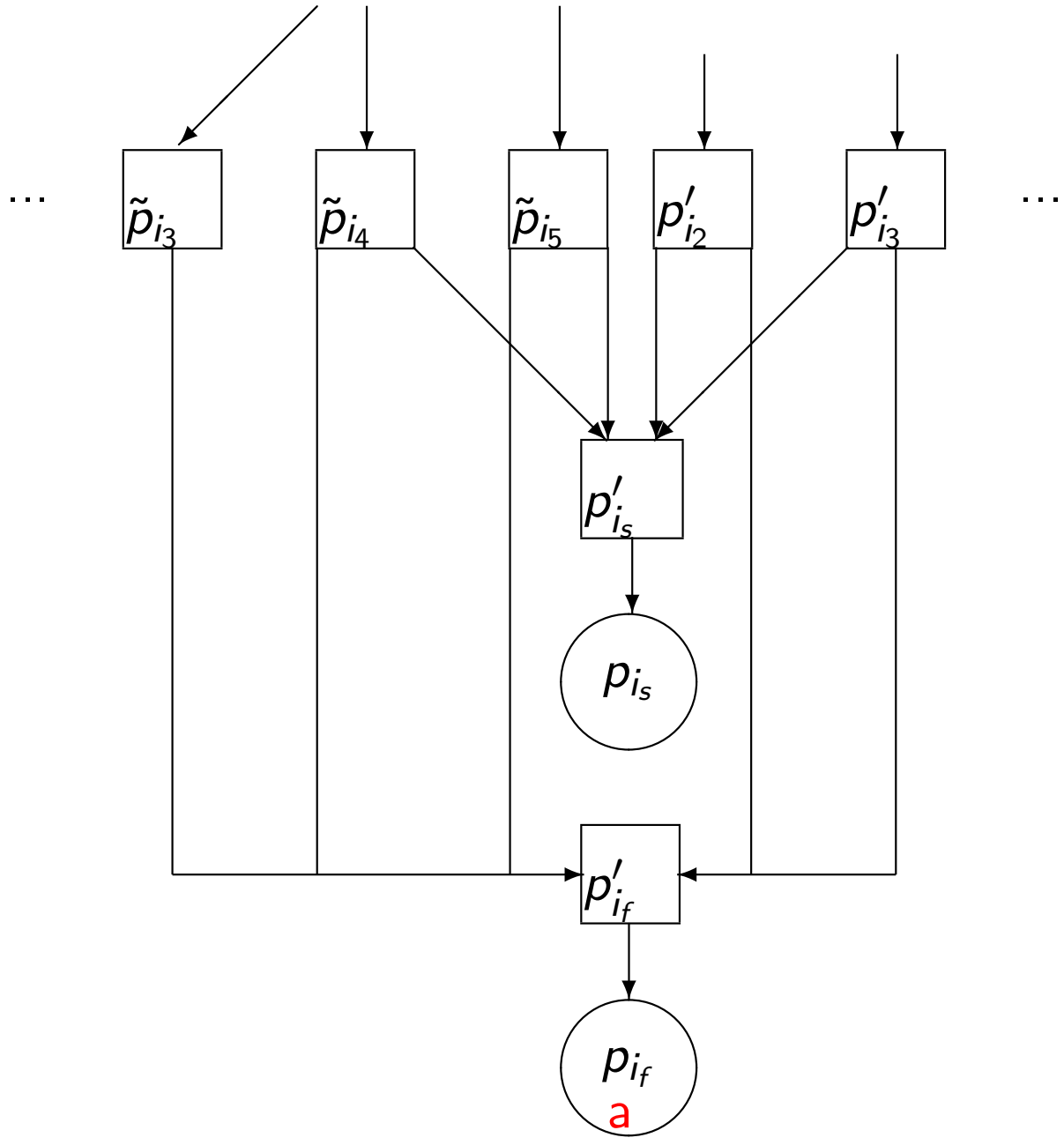
Simulating a SUB-instruction



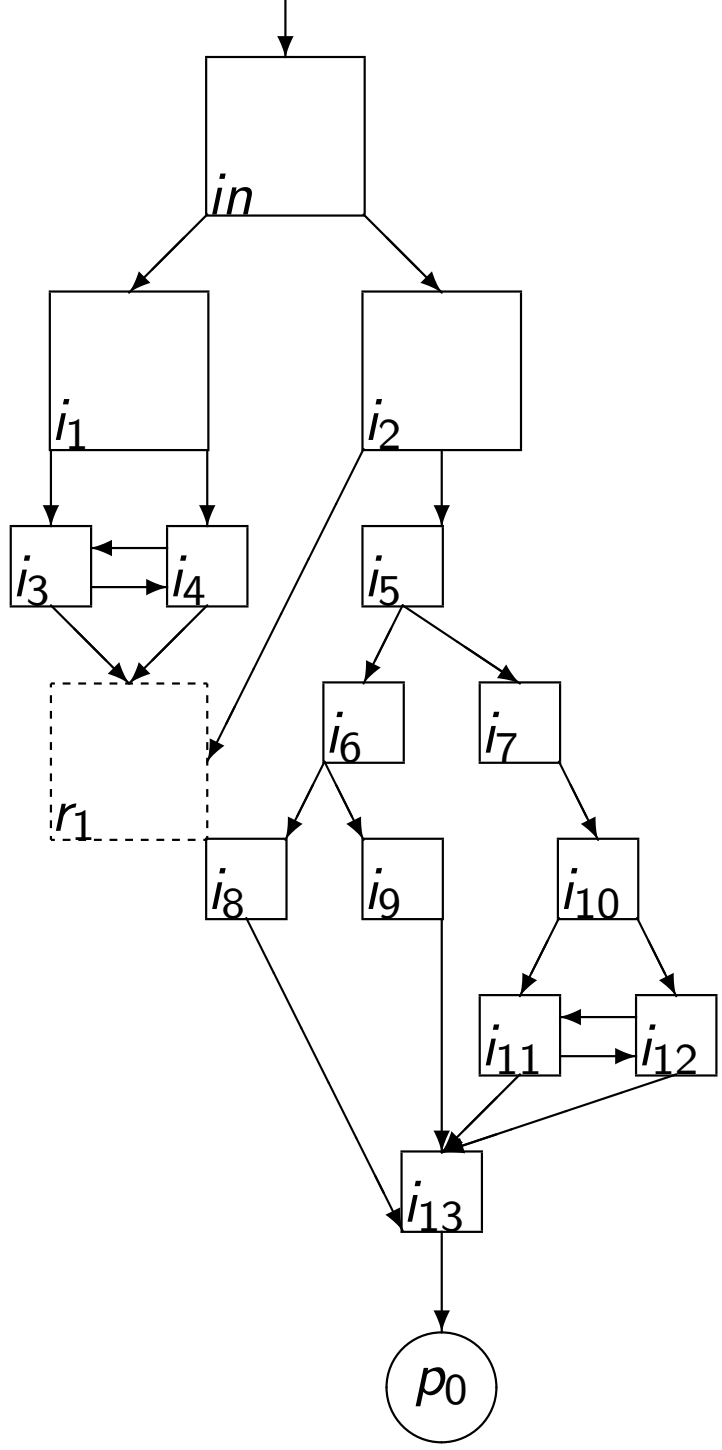
Simulating a SUB-instruction



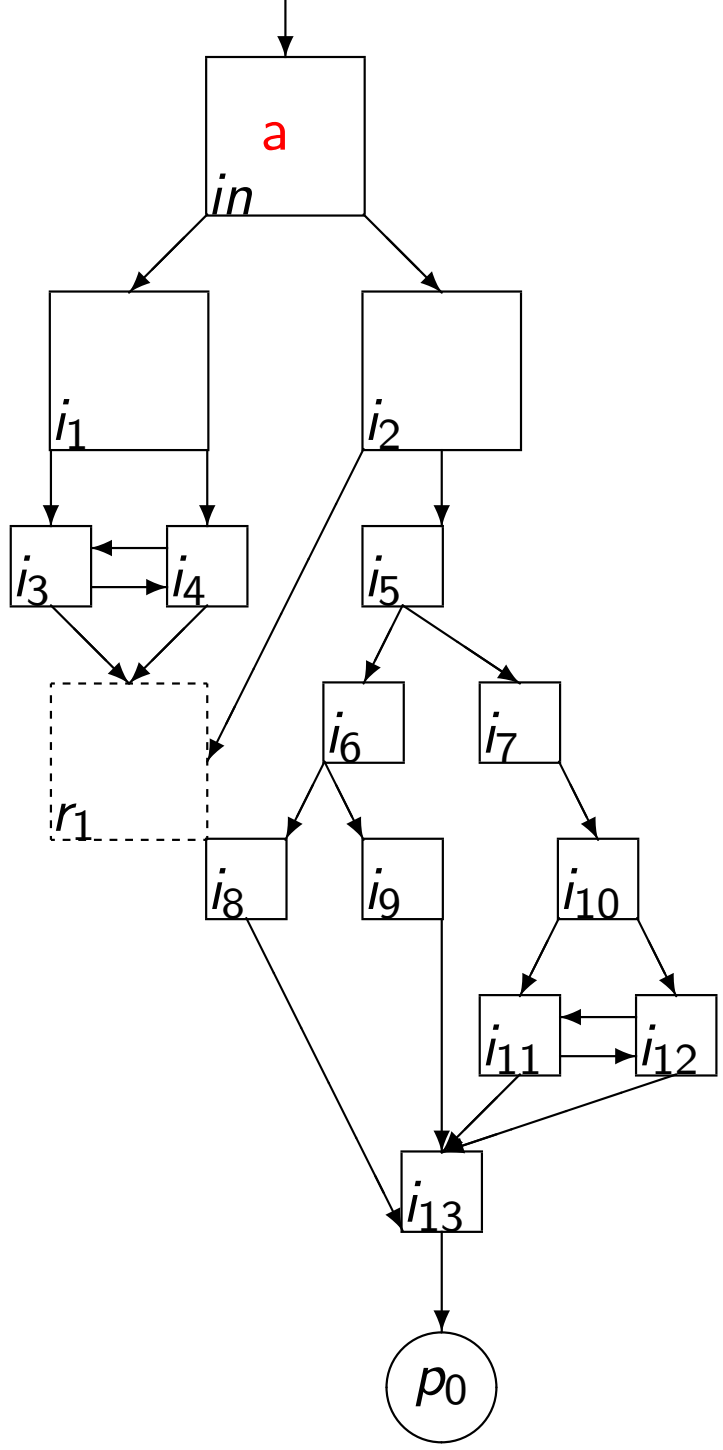
Simulating a SUB-instruction



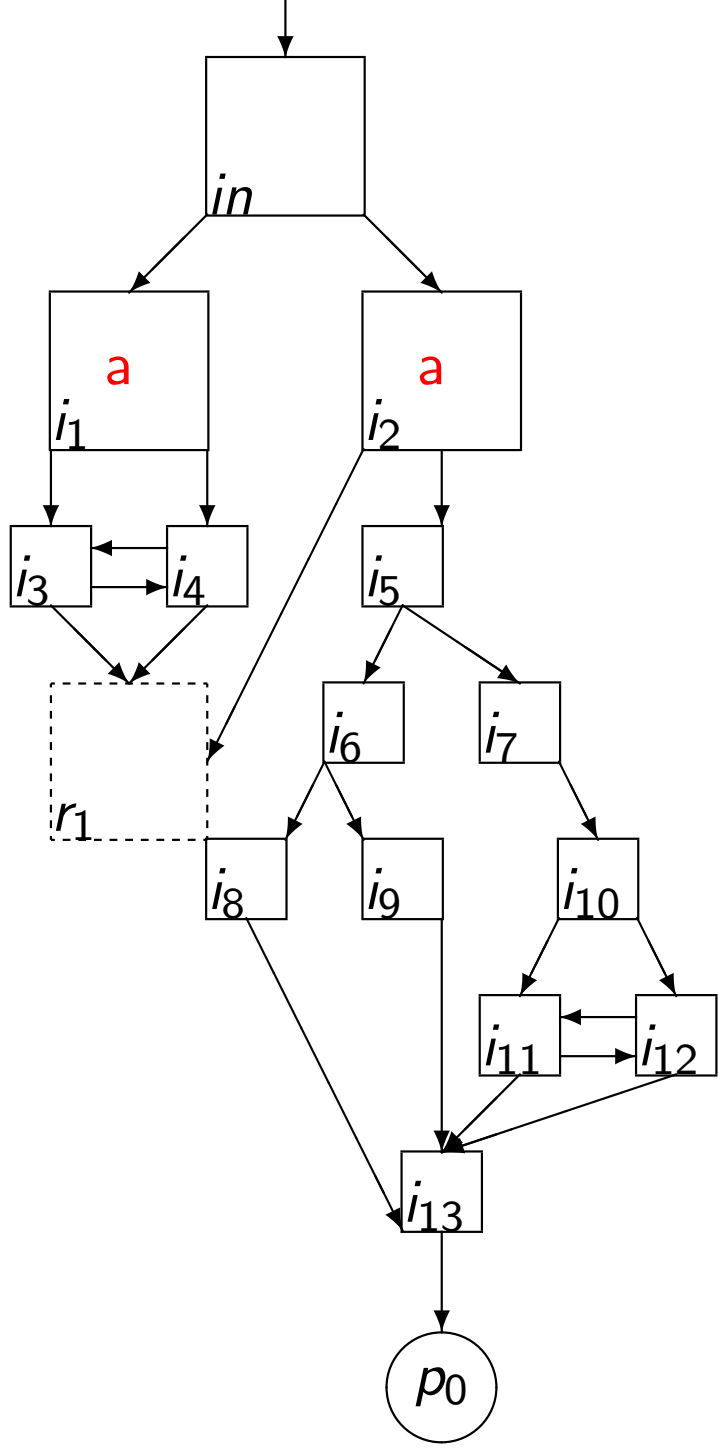
Initialization



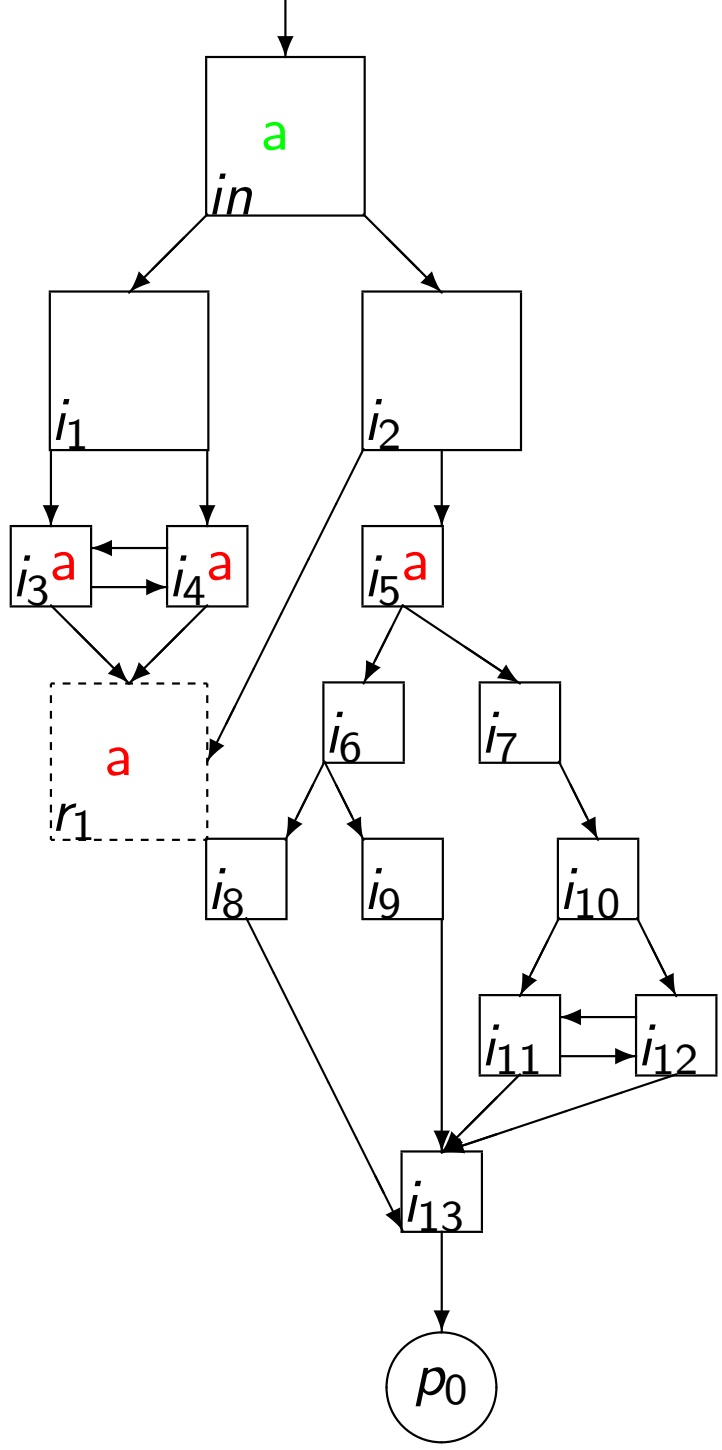
Initialization



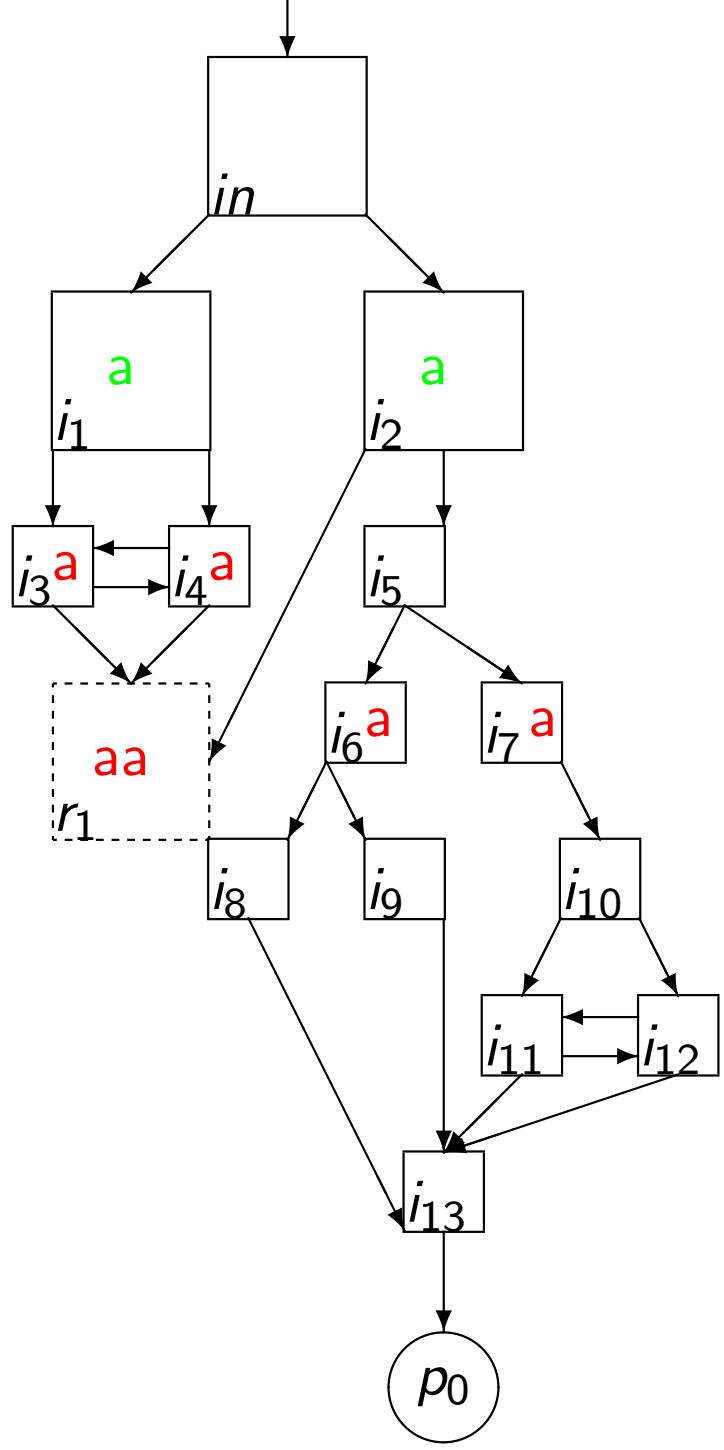
Initialization



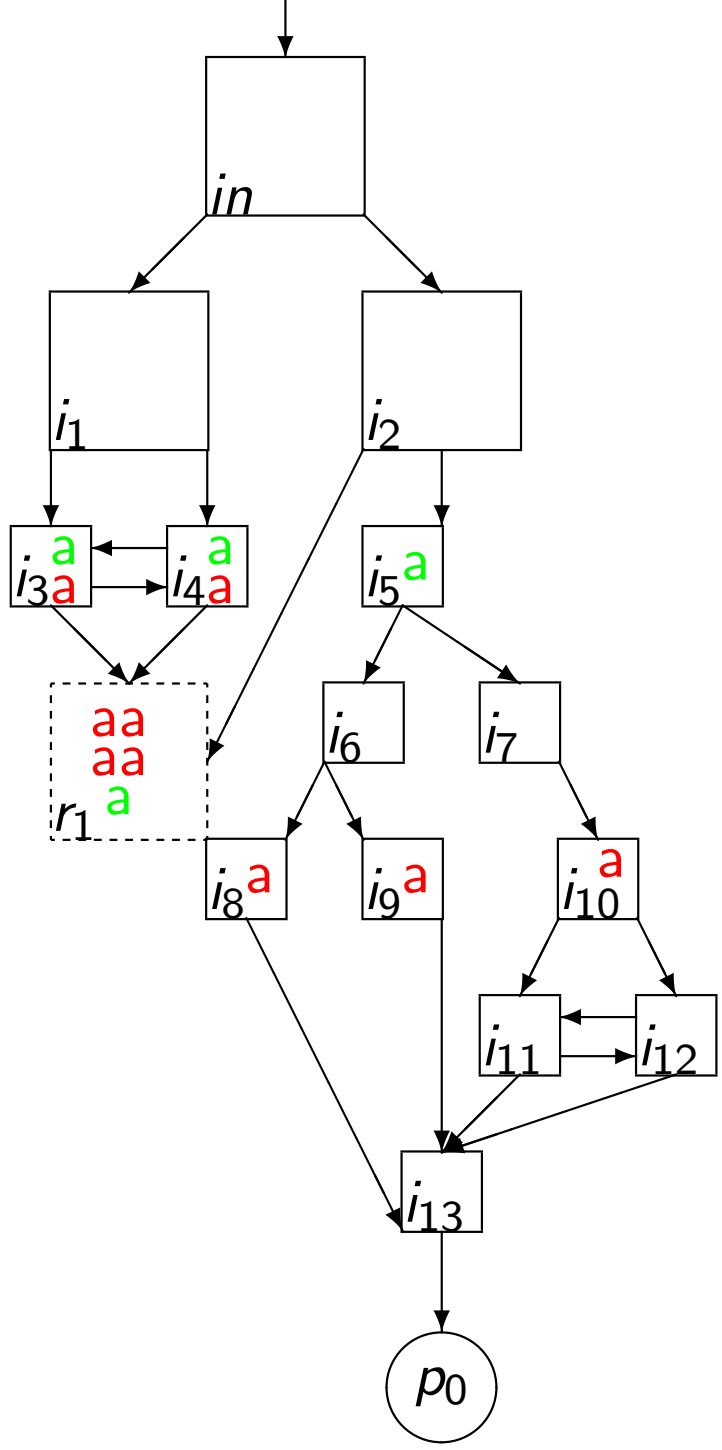
Initialization



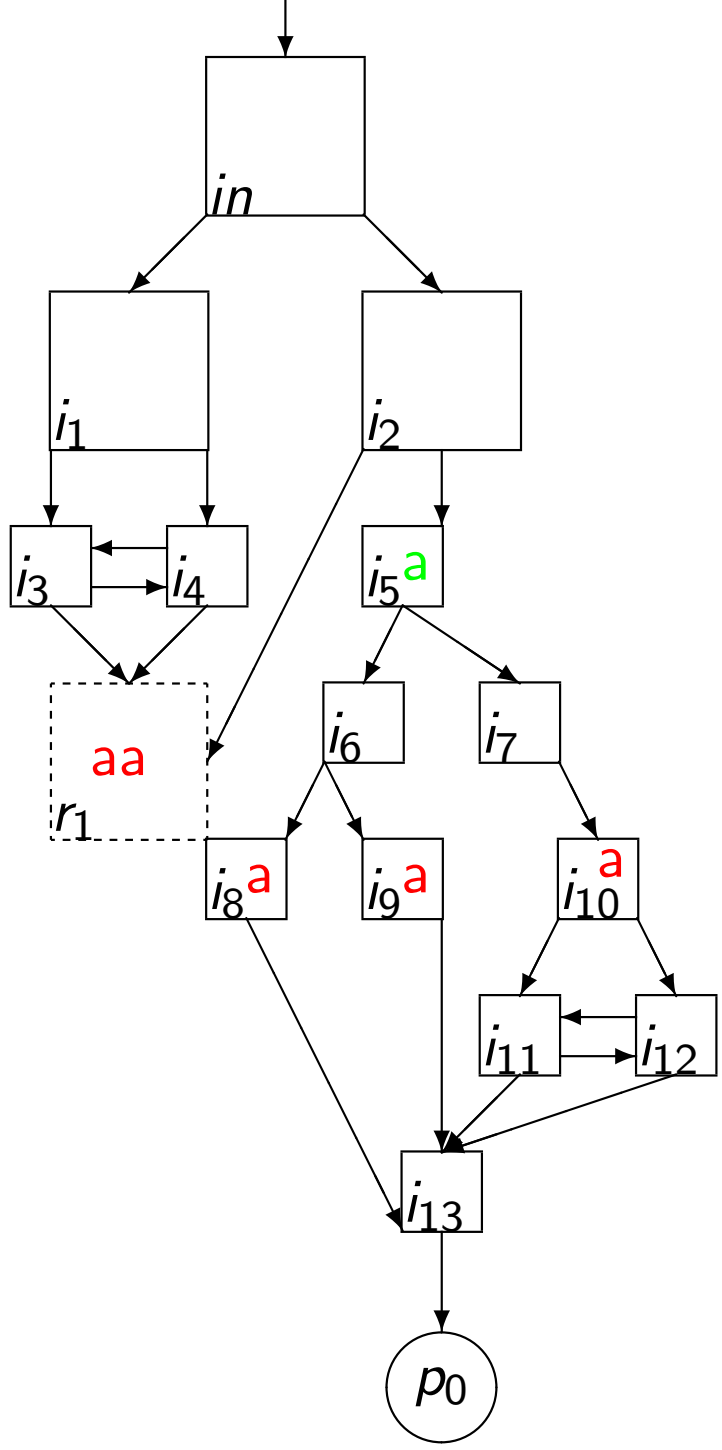
Initialization



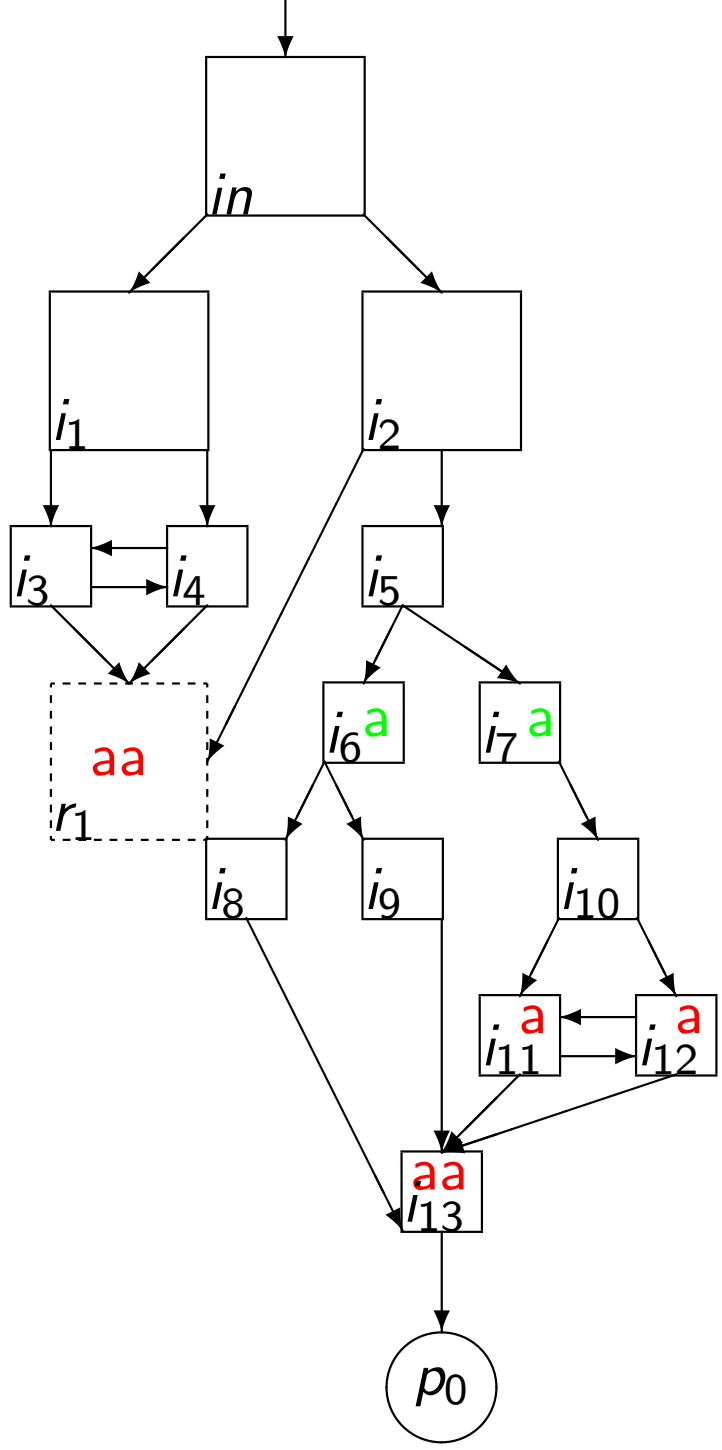
Initialization



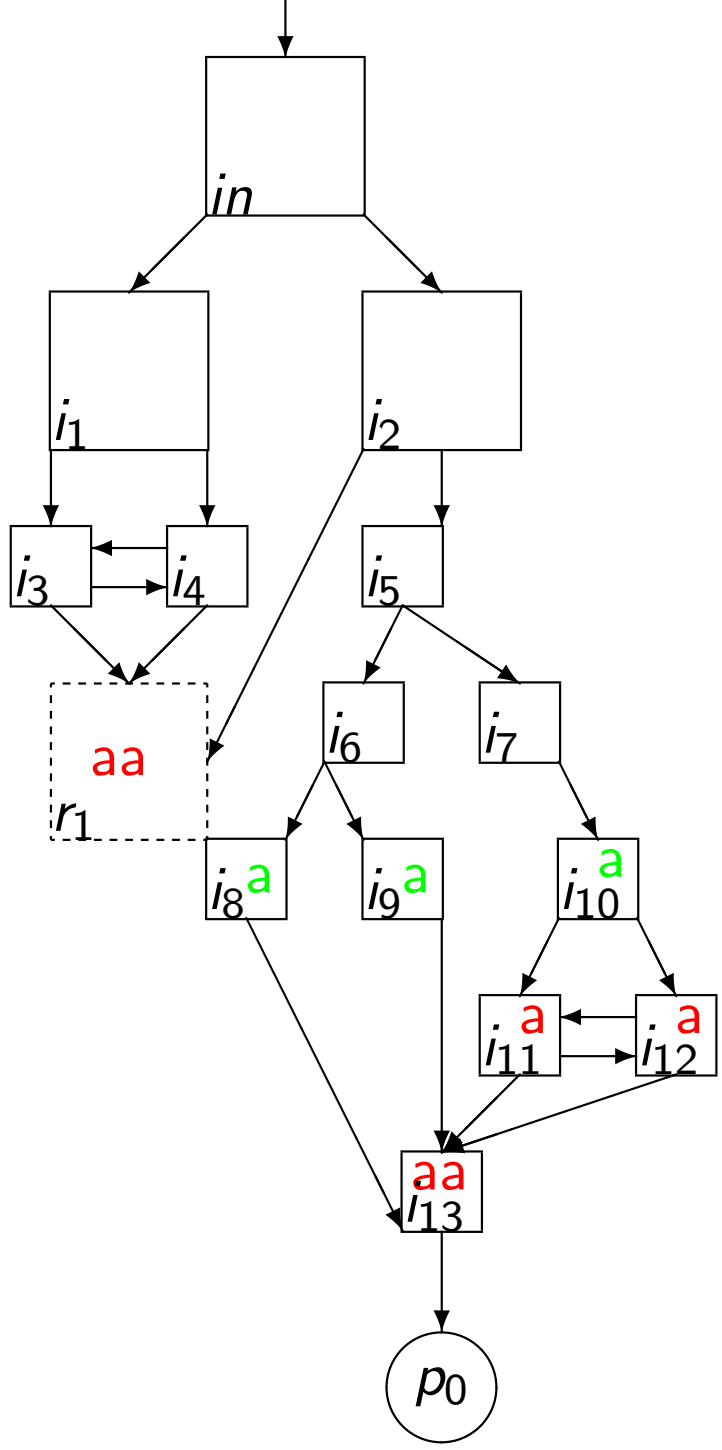
Initialization



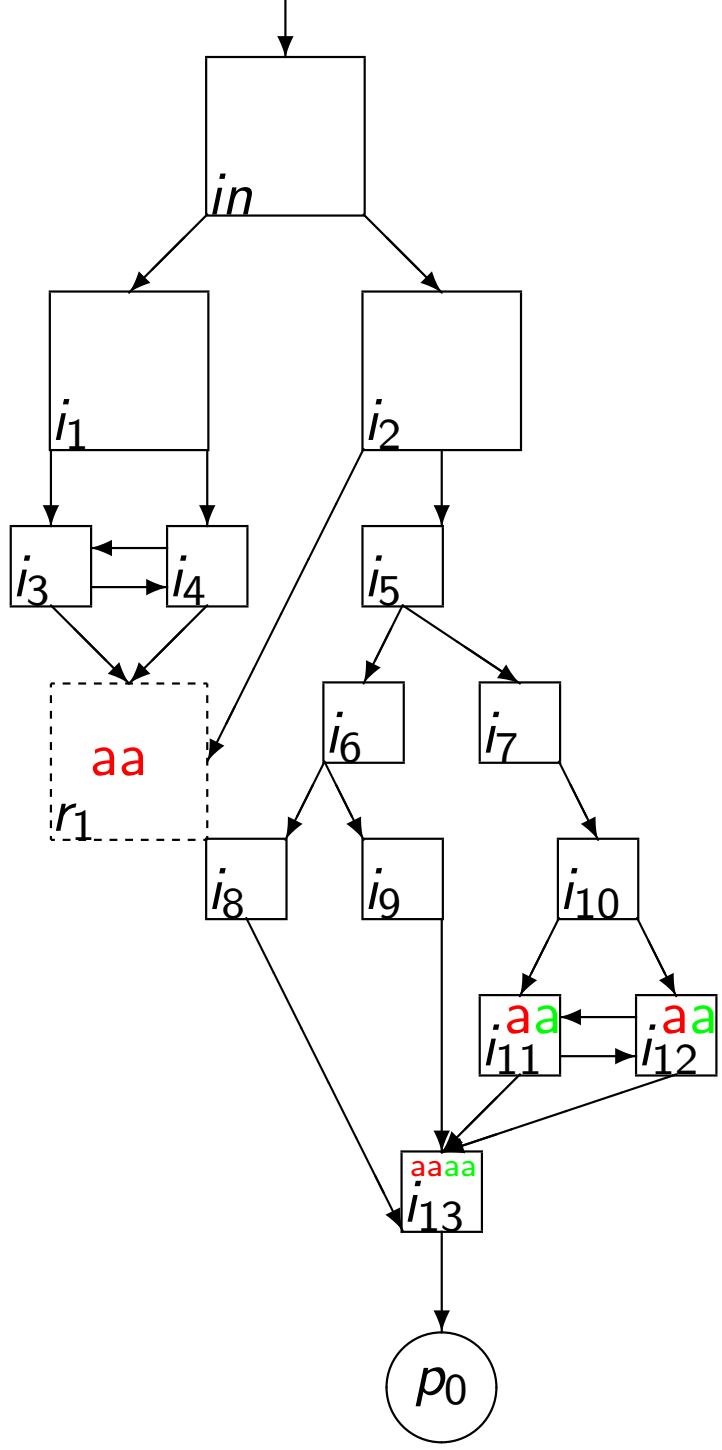
Initialization



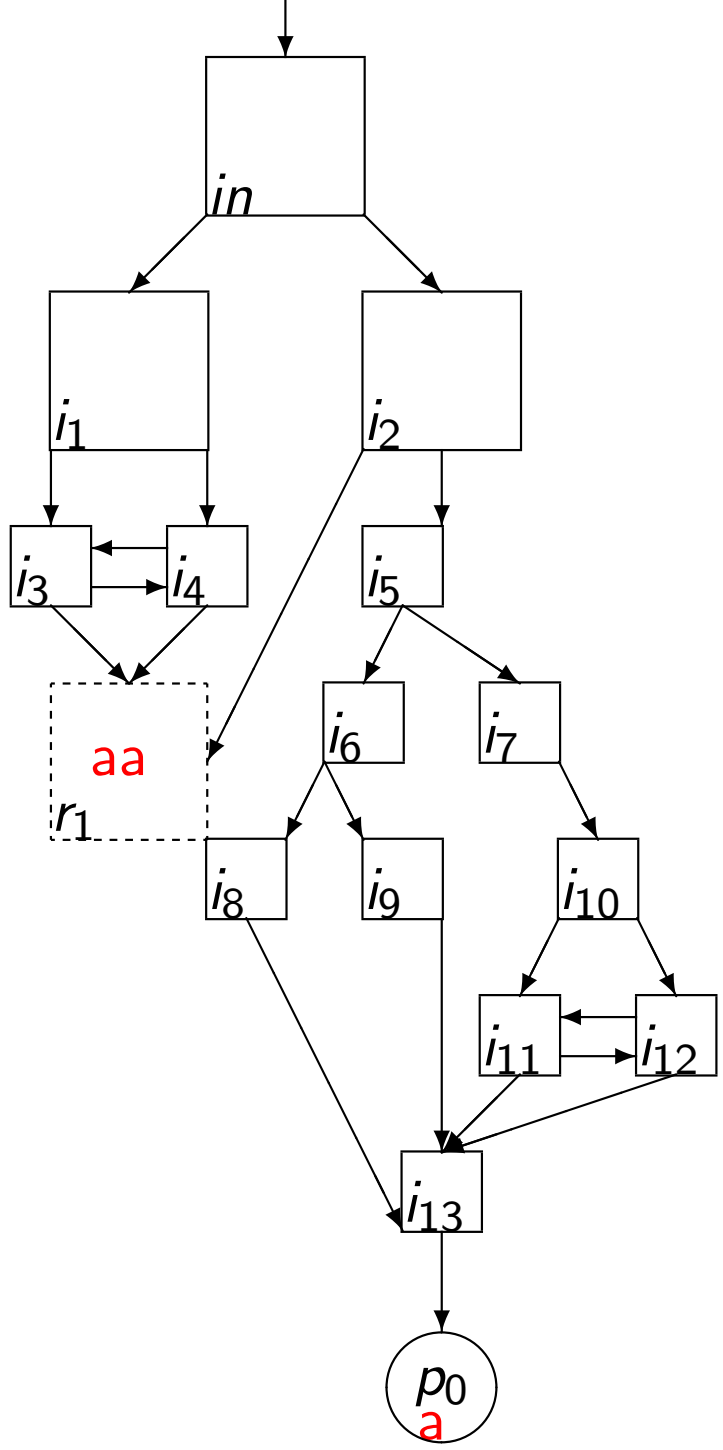
Initialization



Initialization



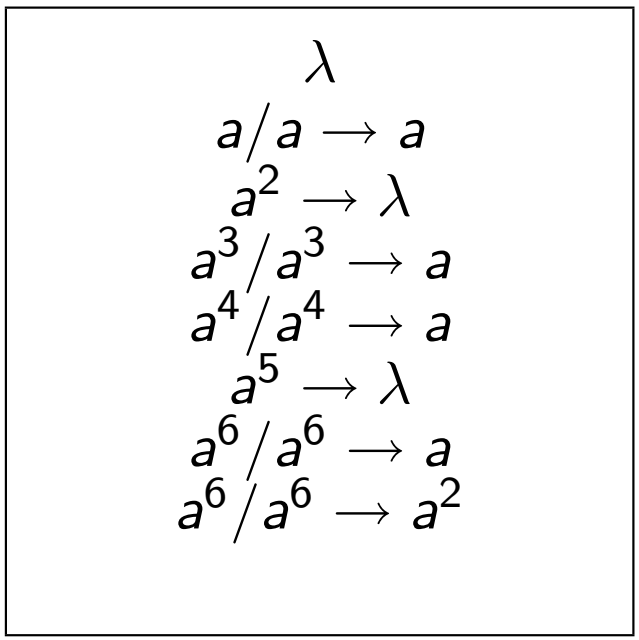
Initialization



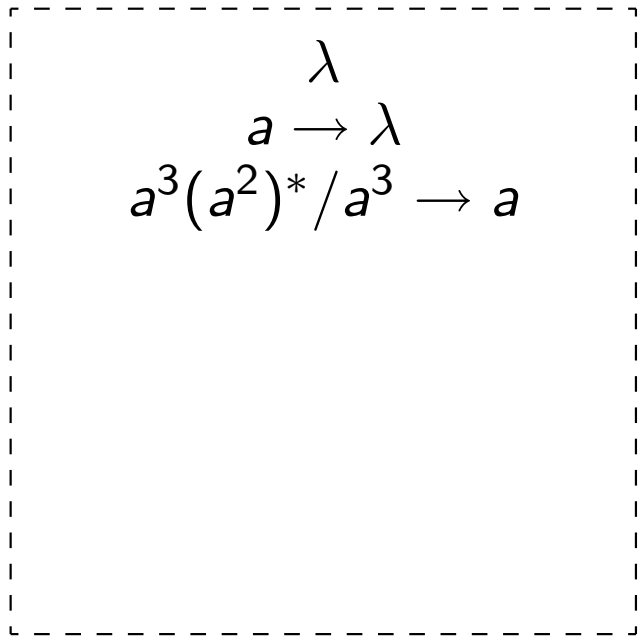
Results

- Accepting spiking neural P systems (without delays) with only two types of neurons are computationally complete.
- Generating spiking neural P systems without delays with only two types of neurons are computationally complete.
- Corollary: Spiking neural P systems without delays with only three neurons with unbounded rules are computationally complete.

The Two Types (Modified for Nondeterminism)

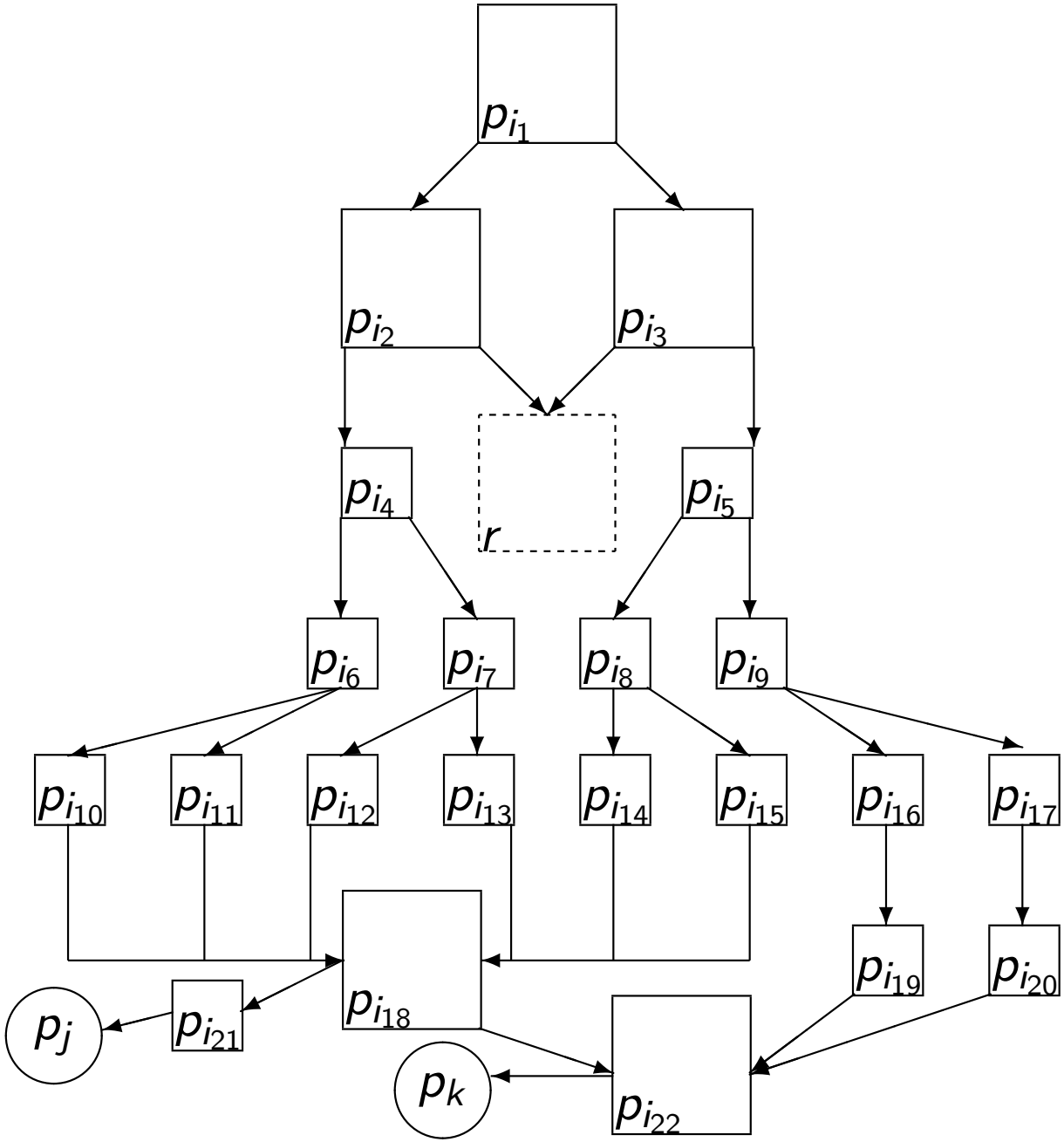


Type 1

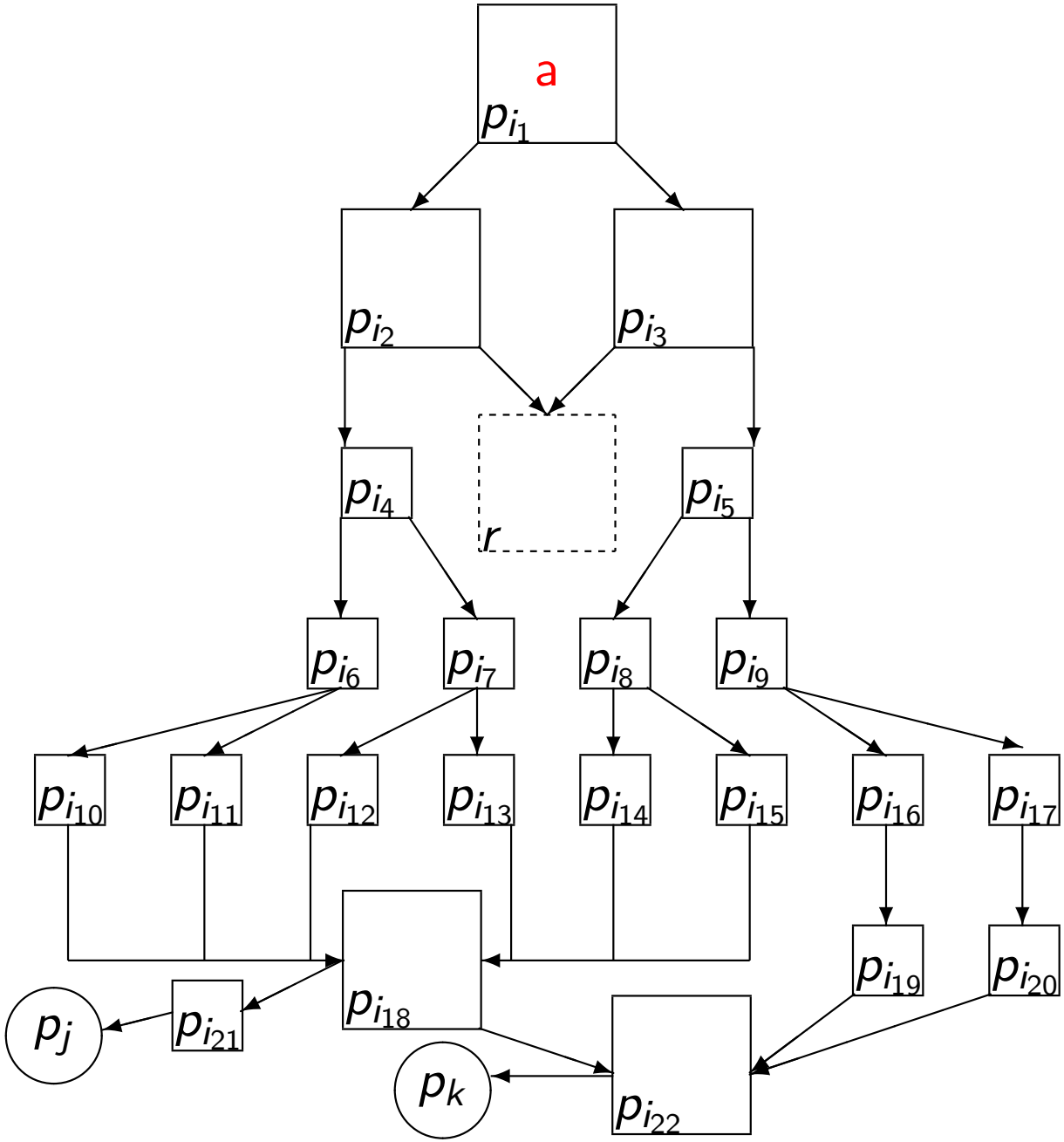


Type 2

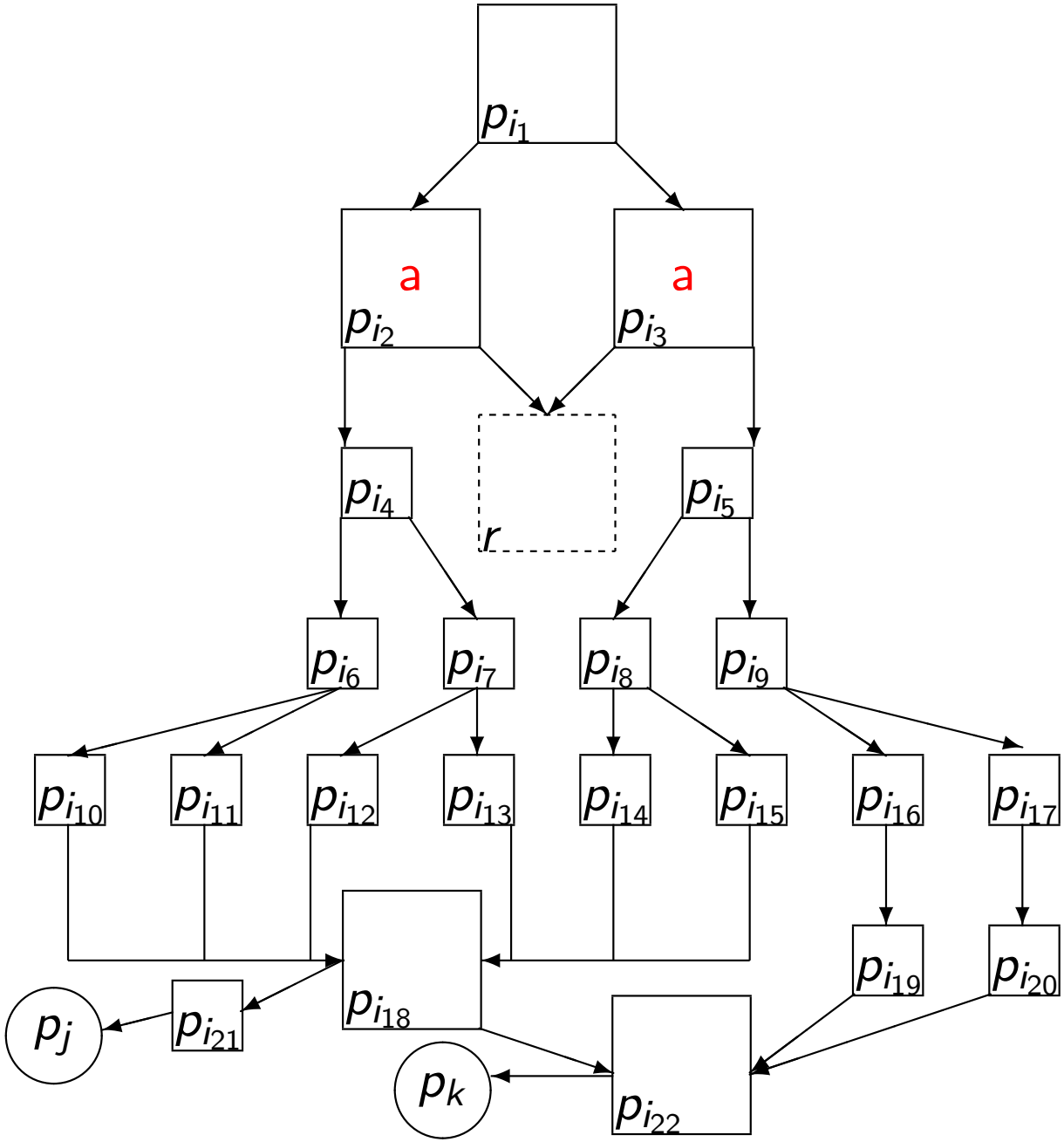
Simulating a non-deterministic ADD-instruction



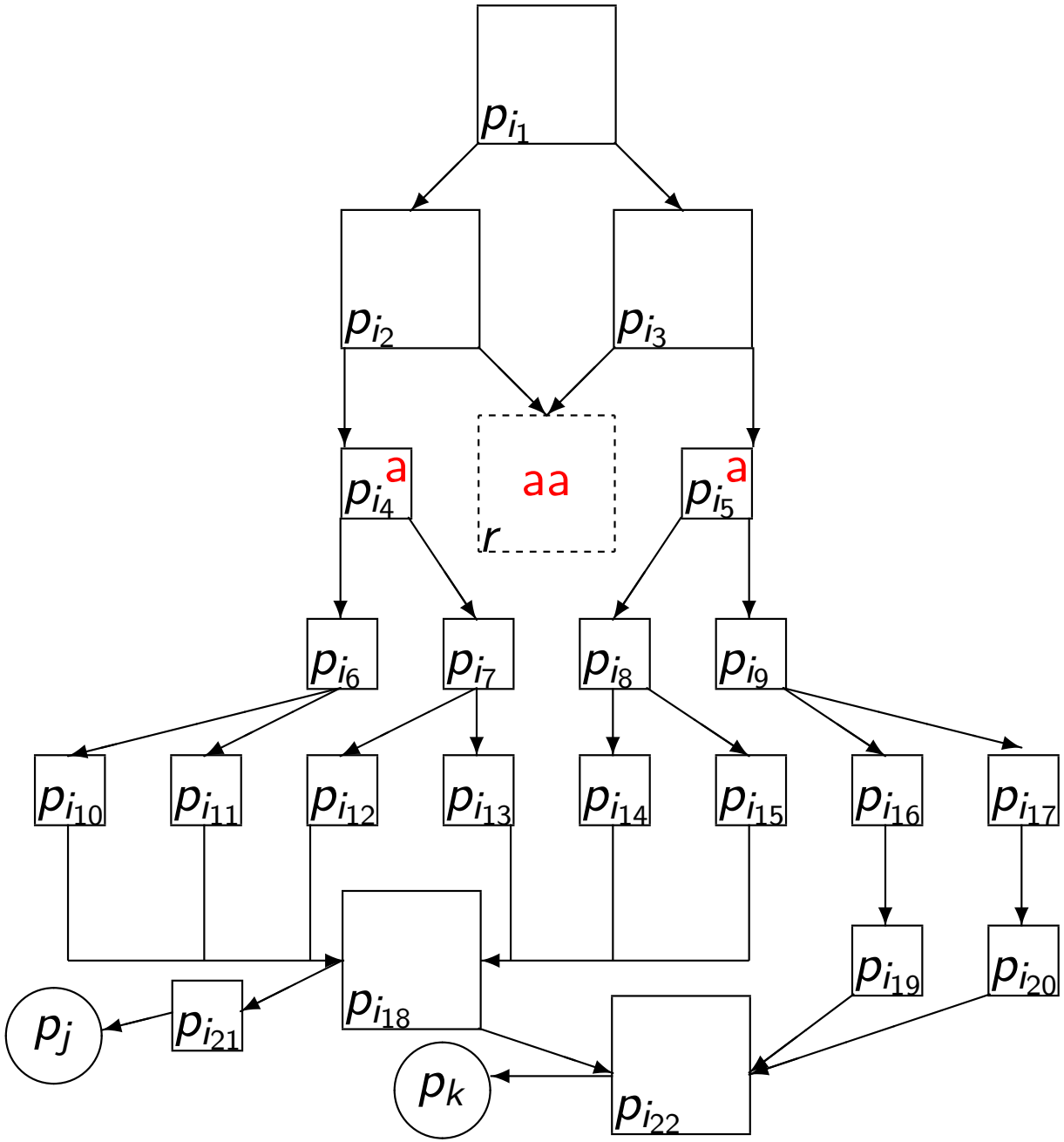
Simulating a non-deterministic ADD-instruction



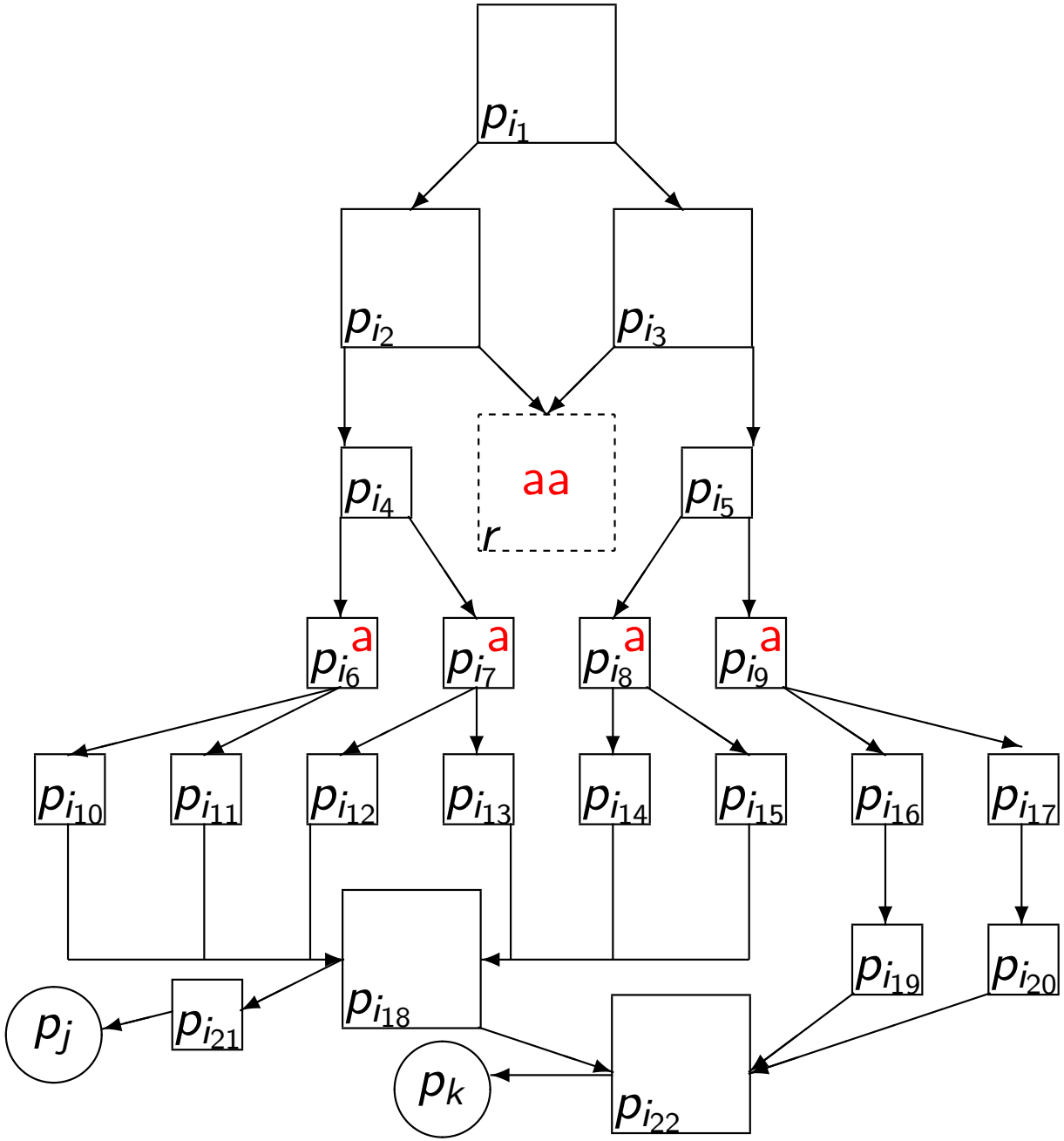
Simulating a non-deterministic ADD-instruction



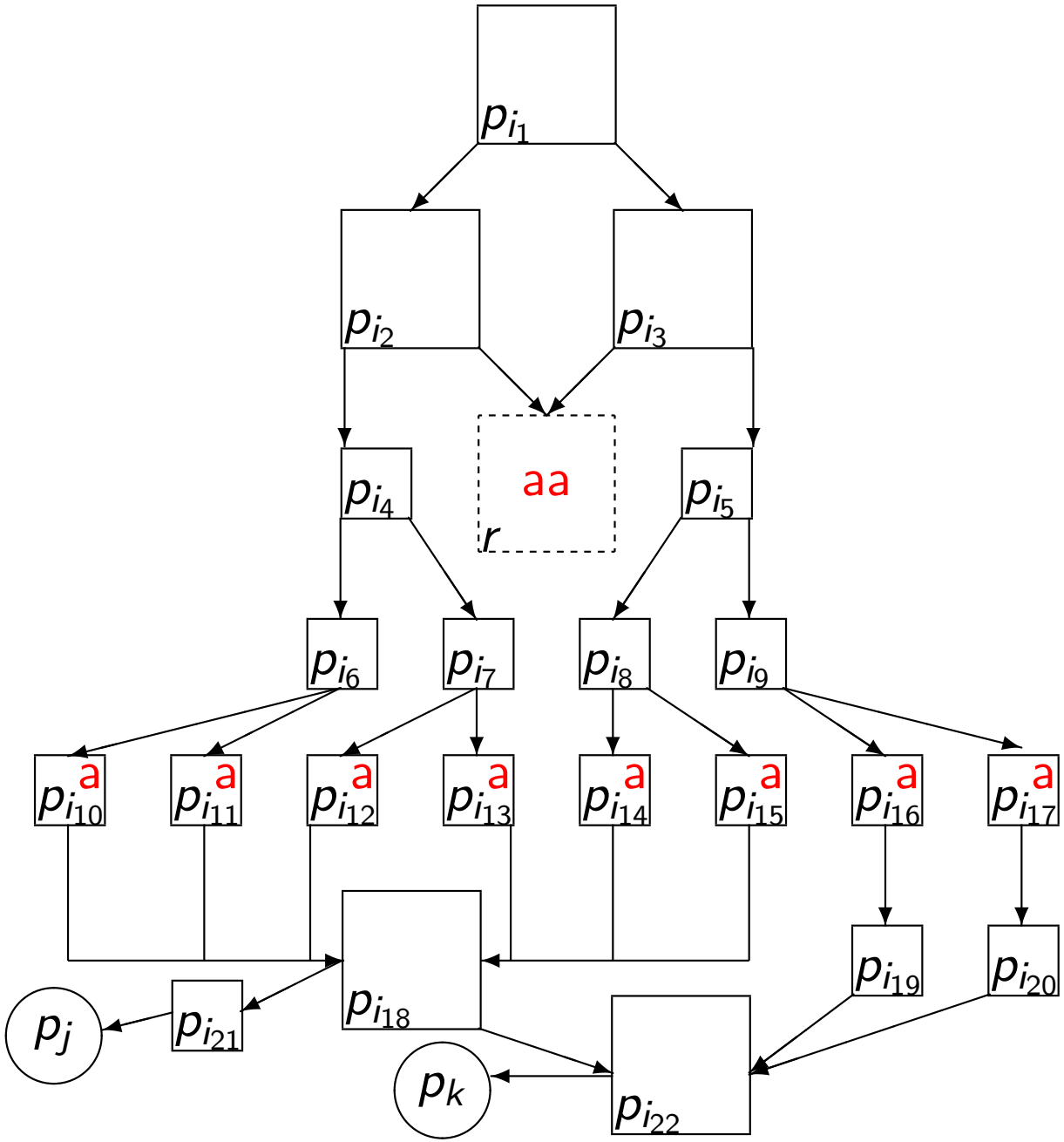
Simulating a non-deterministic ADD-instruction



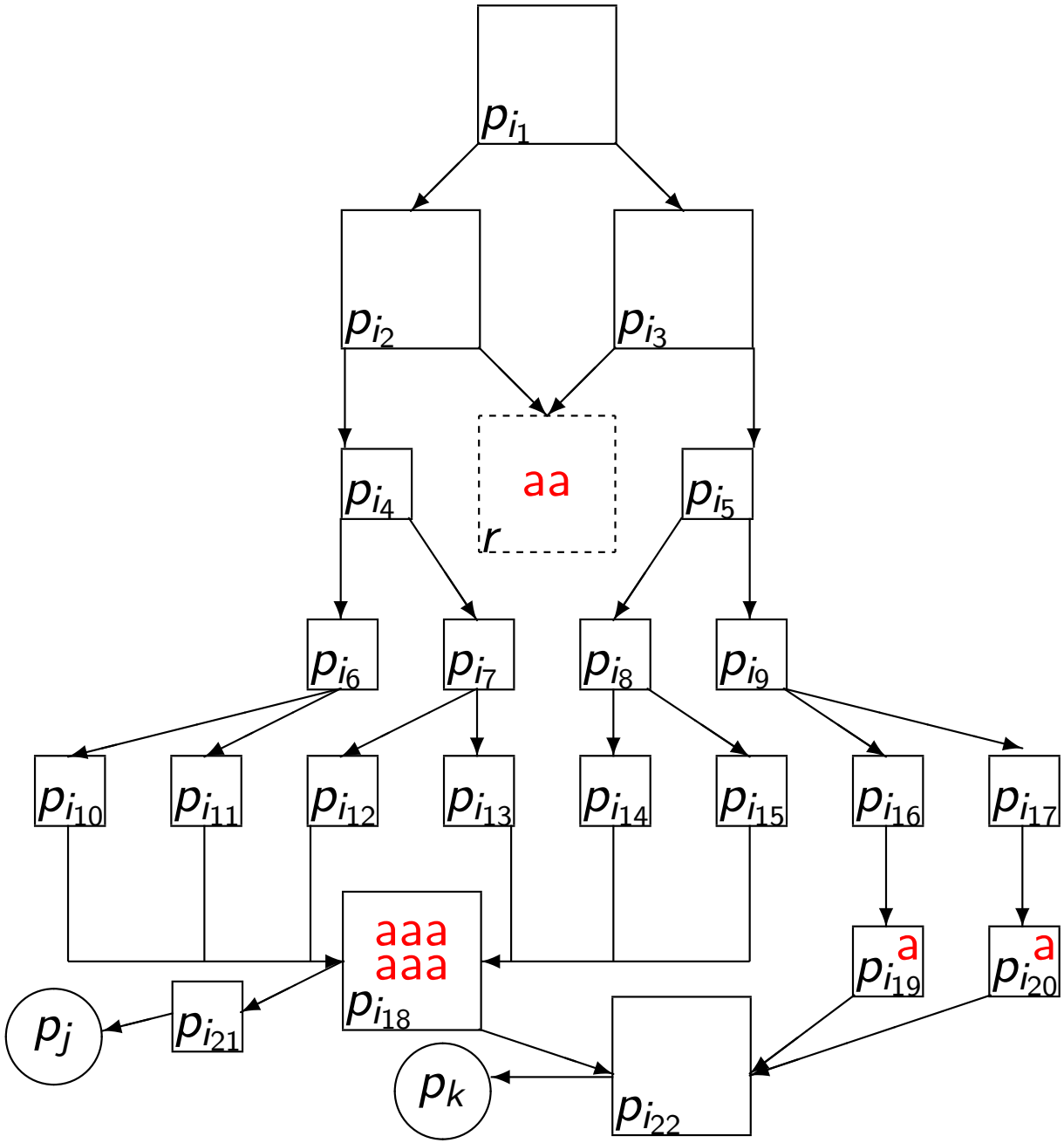
Simulating a non-deterministic ADD-instruction



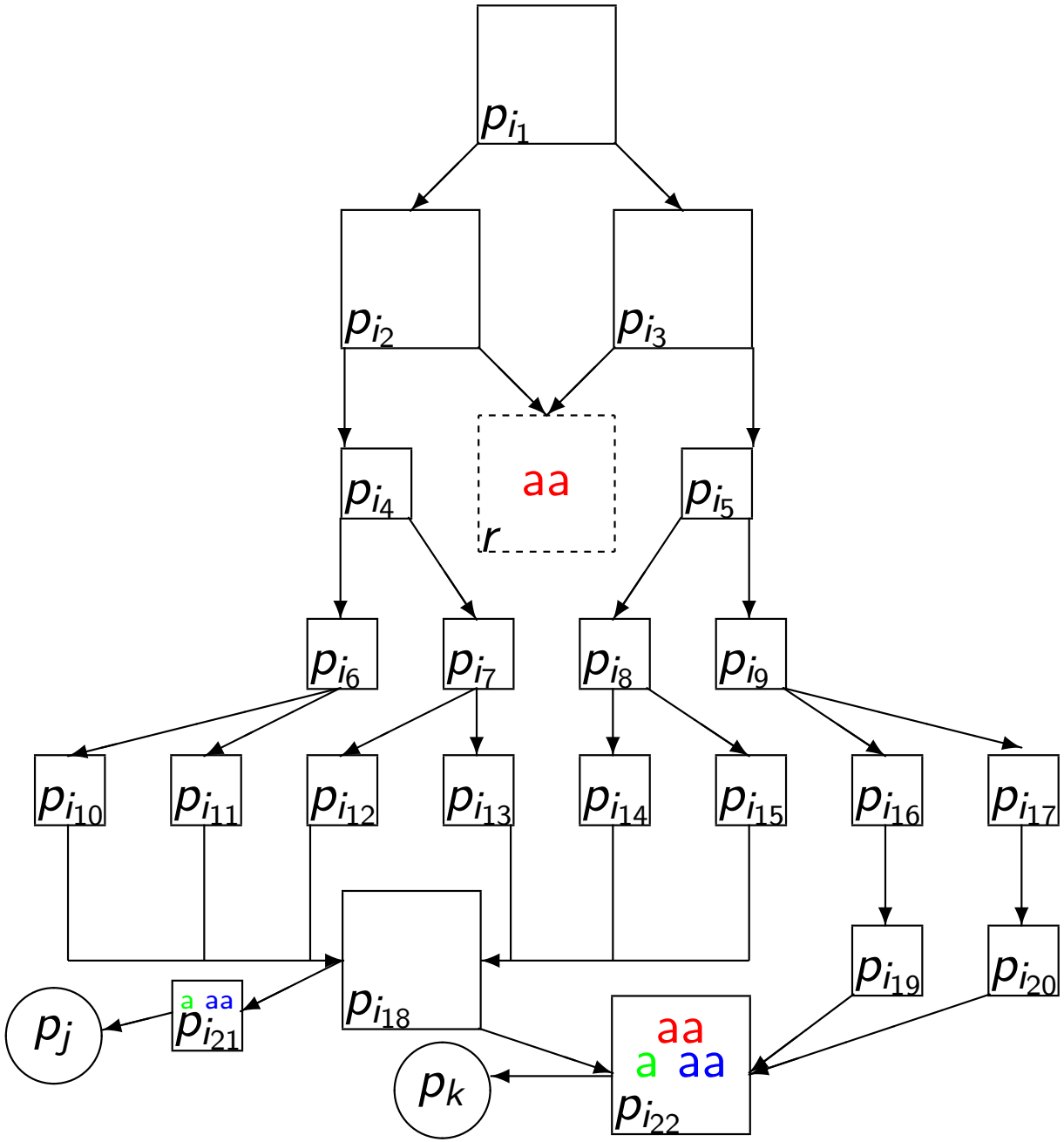
Simulating a non-deterministic ADD-instruction



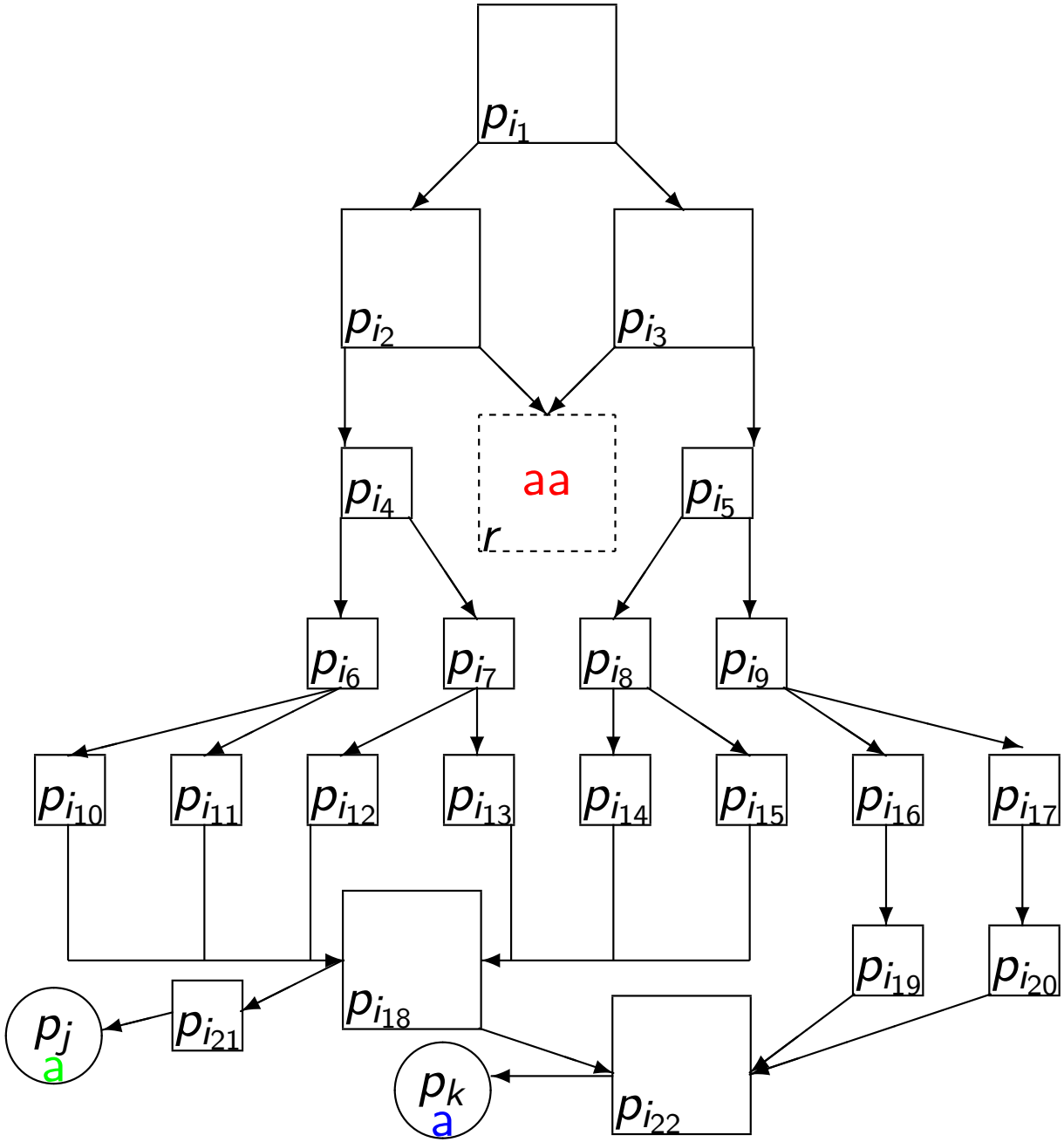
Simulating a non-deterministic ADD-instruction



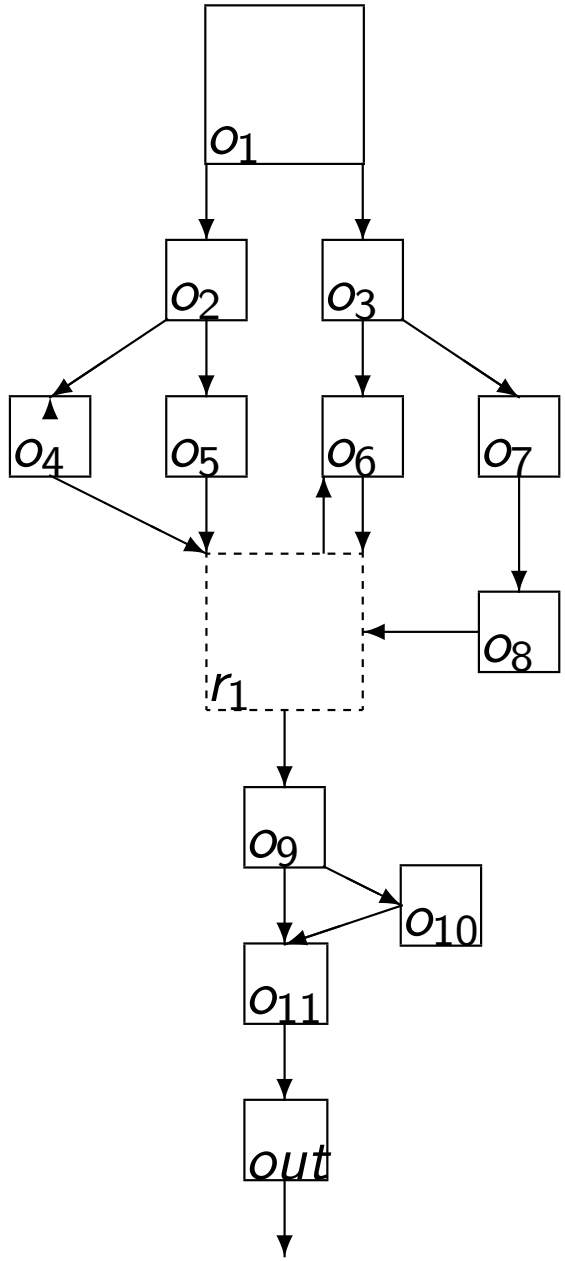
Simulating a non-deterministic ADD-instruction



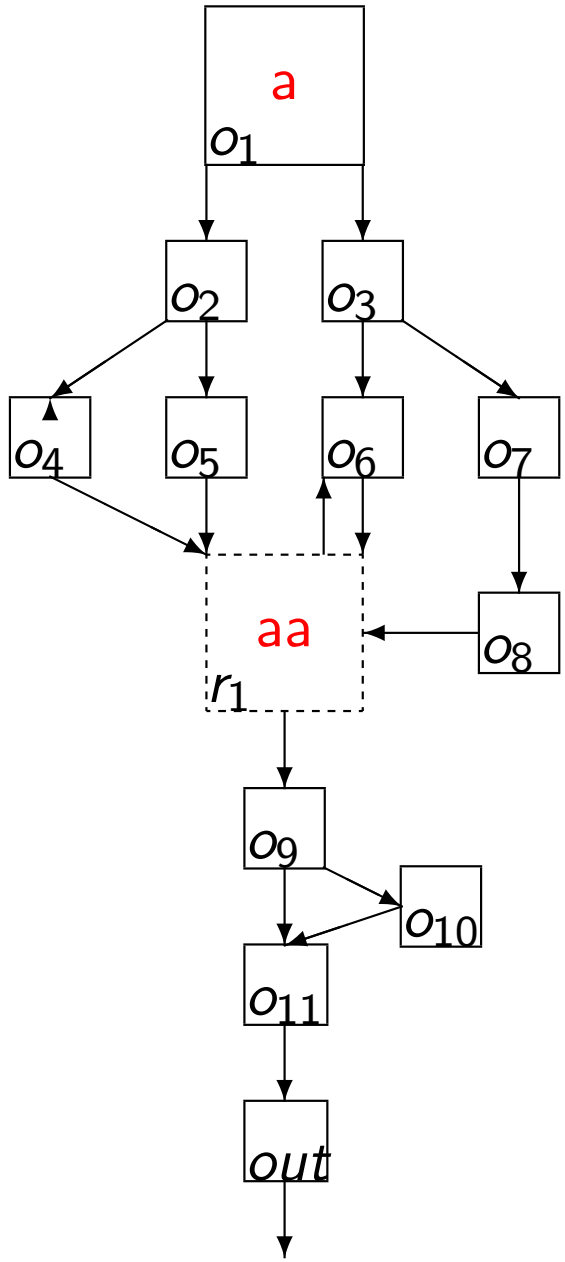
Simulating a non-deterministic ADD-instruction



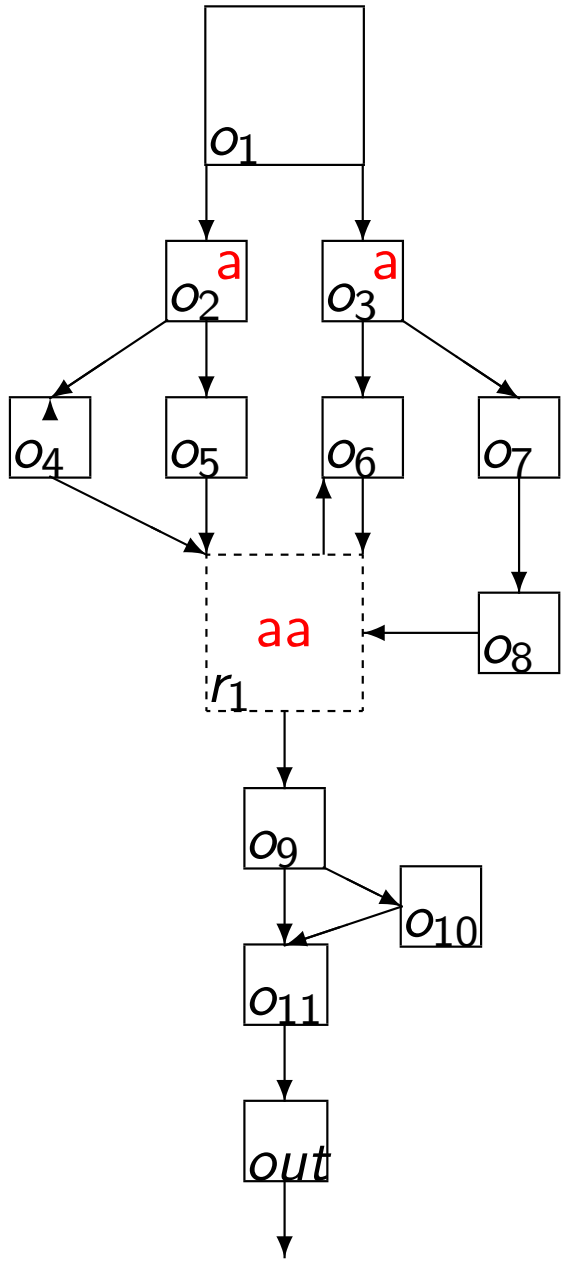
Output



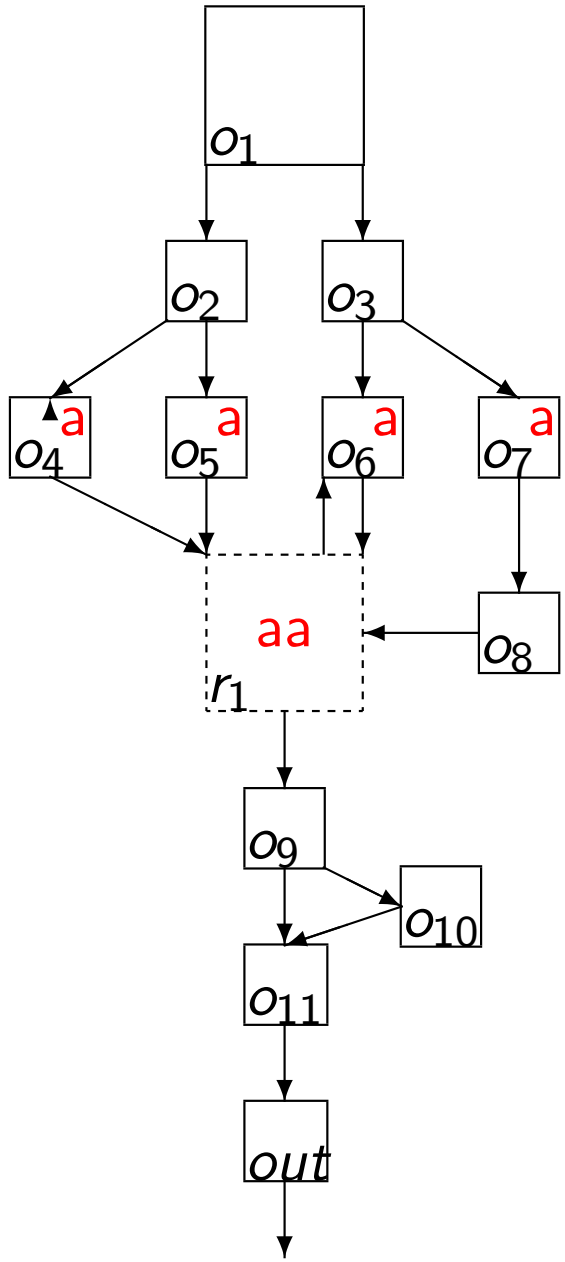
Output



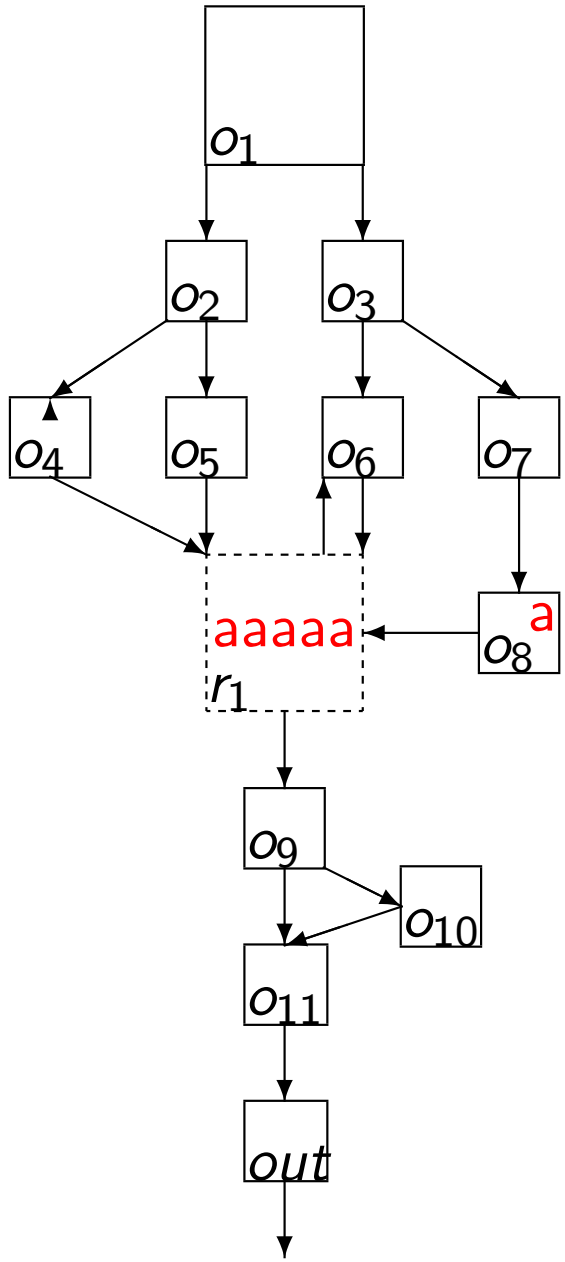
Output



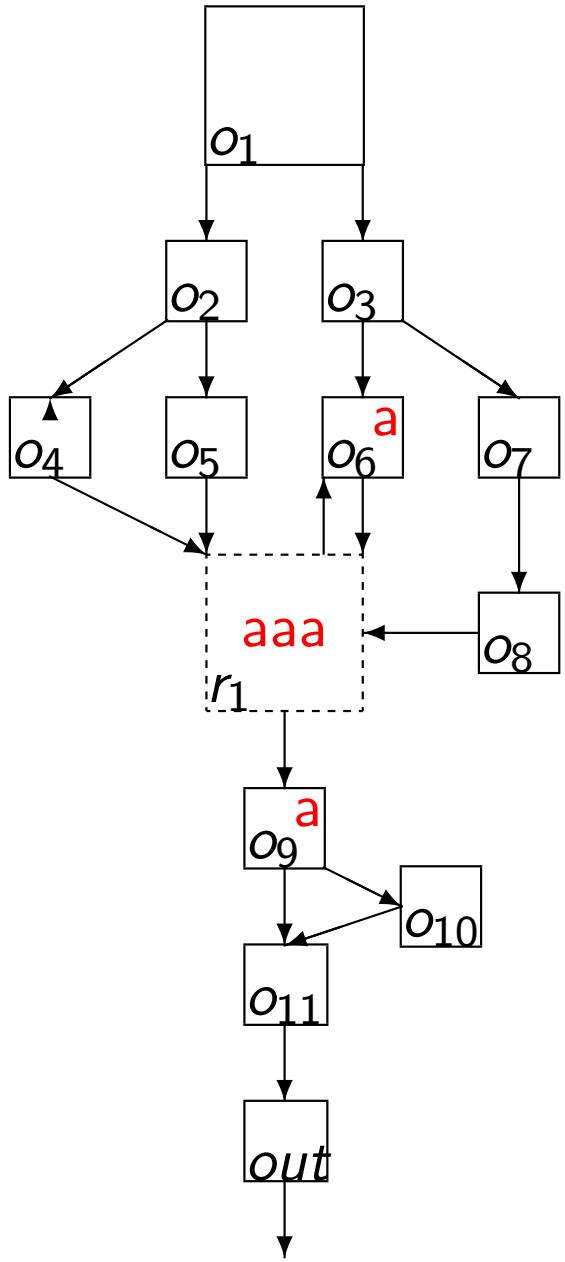
Output



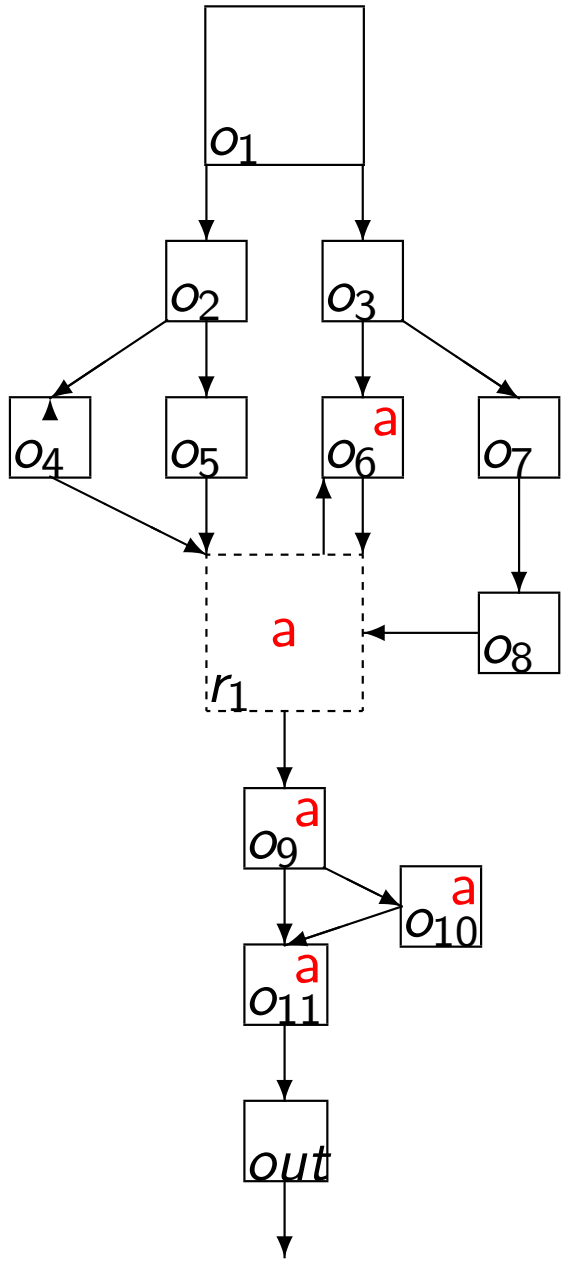
Output



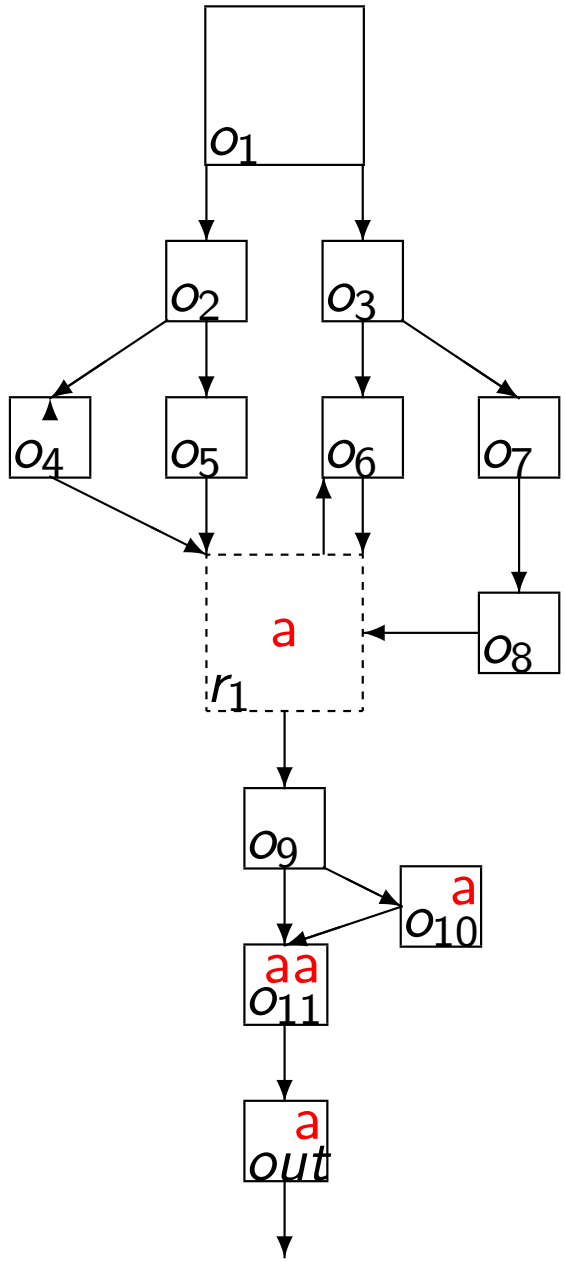
Output



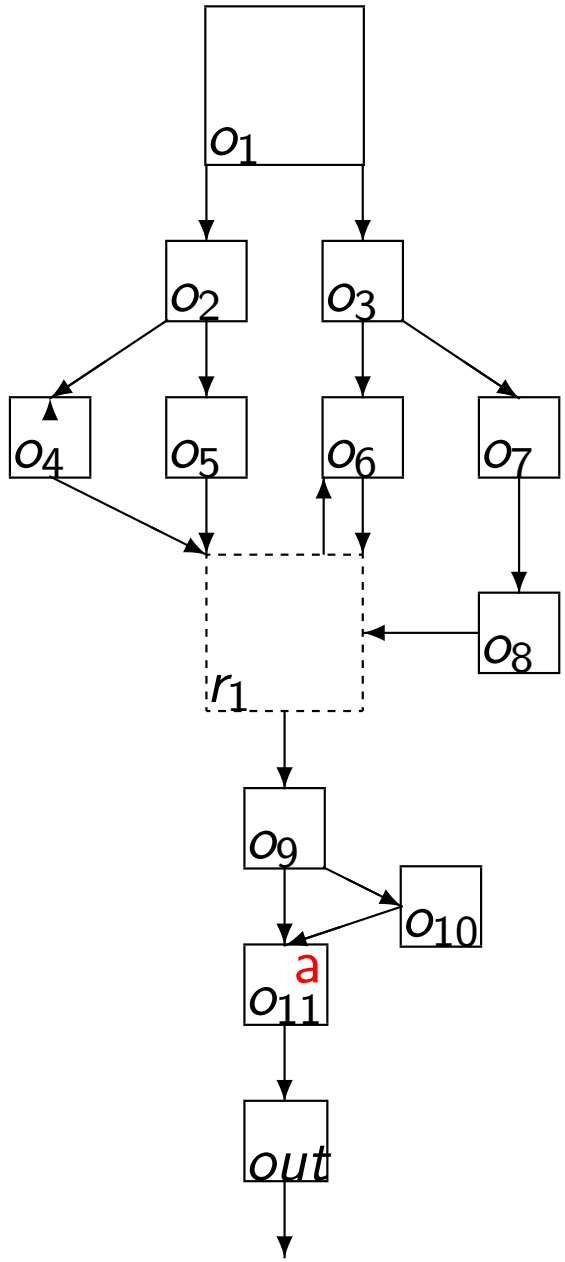
Output



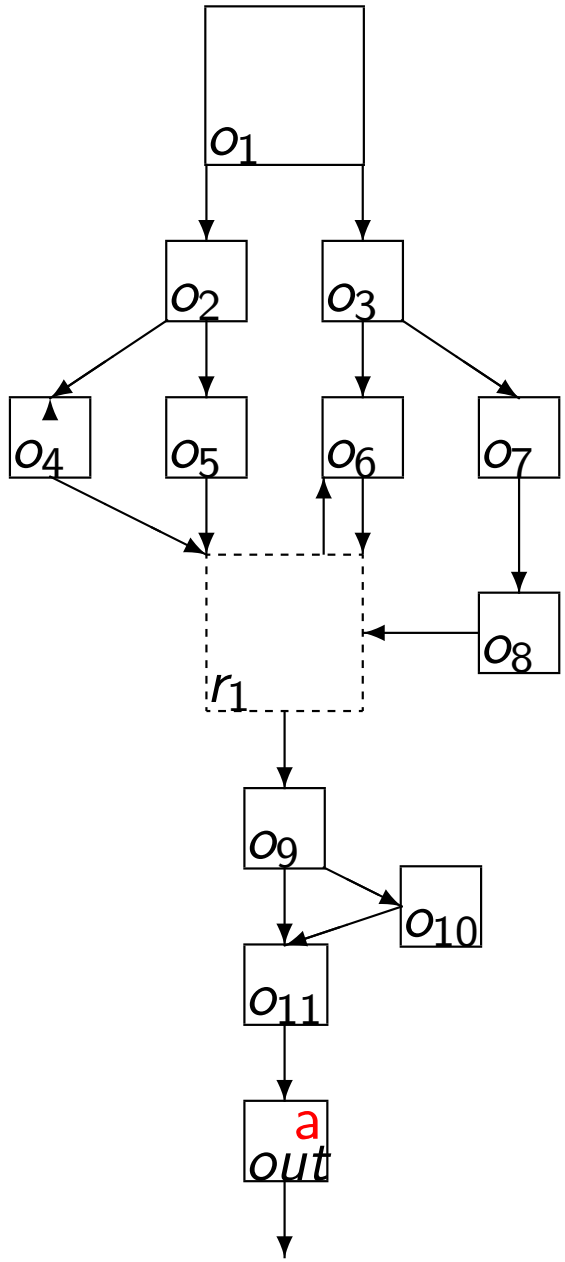
Output



Output



Output



Results

- Accepting spiking neural P systems (without delays) with only two types of neurons are computationally complete.
- Generating spiking neural P systems without delays with only two types of neurons are computationally complete.
- Corollary: Spiking neural P systems without delays with only three neurons with unbounded rules are computationally complete.

Suggestions for Future Work

- Which ingredients are needed to generalize the results for recursively enumerable sets of vectors of natural numbers?
- Which changes in the constructions of the proofs elaborated in this paper are necessary if the input (in the accepting case) or the output (in the generating case) are initially (finally) given as contents of an unbounded input (output) neuron?
- To which number of neurons with unbounded rules can the constructions in the proofs elaborated in this paper be reduced when simulating universal register machines (using spike trains with three spikes to introduce the code of the machine to be simulated as well as its input)?
- For all these variants of spiking neural P systems without delay, is one type of (unbounded) neurons sufficient?