Sleptsov Net Computing

T

Dmitry A. Zaitsev
International Humanitarian University, Ukraine

INTRODUCTION

Recently many researchers introduce new models of hyper-computations, such as quantum computations, computations on cell membranes, spiking P neurons and DNA (Cook & Neary, 2013), capable breaking through the obstacle of intractable tasks. Petri nets have been known for years as a model of concurrent systems (Murata, 1989) but their computationally universal extensions are exponentially slow comparing Turing machines, especially when implementing arithmetic operations. A Sleptsov net concept, suggested quarter a century ago, recently acquired its second birth (Zaitsev, 2016) due to its ability of fast implementation of basic arithmetic operations. Firing a transition in a few instances at a step leads to universal constructs which run in polynomial time (Zaitsev, 2017). In Sleptsov net computing (Zaitsev, 2014a; Zaitsev & Jürjens, 2016), a program, written in Sleptsov net language preserving concurrency of an application area, runs on Sleptsov net processor which implements concurrent firing of transitions in multiple instances providing computations having ultra-performance.

BACKGROUND

A concept of algorithm was formalized for the first time by Alan Turing in 1936 in the form of an abstract machine which is traditionally called a Turing machine. Universal Turing machine which runs a given Turing machine is considered a prototype of a traditional computer. Besides Turing machines, other computationally universal systems appeared: recursive functions of Kleene, normal algorithms of Markov, tag rewriting systems of Post, register machines of Minsky. Variety of models is explained by controversial requirements of manifold application areas. Recent models employ facilities of massively parallel computations even in such simple constructs as elementary cellular automata which universality was proven in 2004 by Mathew Cook. Besides, smallest universal Turing machines were constructed in 2008 by Turlough Neary and Damien Woods which run in polynomial time. However, the way of programming cellular automata after Mathew Cook does not reveal their ability for massively parallel computing. Sleptsov net concept (Zaitsev, 2016) mends the flaw of Petri nets (Murata, 2013), consisting in incremental character of computations, which makes Sleptsov net computing (Zaitsev, 2014a; Zaitsev & Jürjens, 2016) a prospective approach for ultra-performance concurrent computing. In Zaitsev (2016), an overview of works, which refer to Sleptsov nets (Petri nets with multichannel transitions or multiple firing strategy), is presented.

MAIN FOCUS OF THE ARTICLE

Issues, Controversies, Problems

Definition of Sleptsov Net

A *Sleptsov net* (SN) is a bipartite directed multigraph supplied with a dynamic process (Zaitsev, 2016). An SN is denoted as $N=(P,T,W,\mu_0)$, where P and T are disjoint sets of vertices called *places* and *transitions* respectively, the mapping F specifies *arcs* between vertices, and μ_0 represents the initial state (*marking*).

DOI: 10.4018/978-1-5225-2255-3.ch672

The mapping W: ($P\times T$)→N∪{-1}, ($T\times P$)→N defines arcs, their types and multiplicities, where a zero value corresponds to the arc absence, a positive value – to the *regular arc* with indicated multiplicity, and a minus unit – to the *inhibitor arc* which checks a place on zero marking. N denotes the set of natural numbers. To avoid nested indices we denote $w^-_{j,i} = w(p_j, t_i)$ and $w^+_{i,j} = w(t_i, p_j)$. The mapping μ: P→N specifies the place marking.

In graphical form, places are drawn as circles and transitions as rectangles. An inhibitor arc is represented by a small hollow circle at its end, and a small solid circle represents the abbreviation of a loop. Regular arc's multiplicity, greater than unit, is inscribed on it and place's marking, greater than zero, is written inside it. Examples of SNs computing basic arithmetic and logic operations are shown in Figure 6.

To estimate *multiplicity of firability conditions* on each incoming arc of a transition, the following auxiliary operation is defined

$$x \succ y = \begin{cases} x \,/\, y, & if\ y > 0 \\ 0, & if\ y = -1,\ x > 0, \\ \infty, & if\ y = -1,\ x = 0. \end{cases}$$

To avoid inconsistency with infinite number of instances, here we prohibit transitions without input regular arcs.

The behavior (dynamics) of a SN could be described by the corresponding state equation similarly to (Zaitsev, 2012). The present work considers the behavior as result of applying the following *transition firing rule*:

- The number of instances of transition t_i firable at the current step is equal to

$$v_i = v(t_i) = \min_j(\mu_j \succ w^-_{j,i}), 1 \le j \le m,\ w^-_{j,i} \ne 0$$

- When transition t_i, $v_i > 0$ fires, for $u_i \le v_i$ it
  - extracts $u_i \cdot w^-_{j,i}$ tokens from each its input place p_j for regular arcs $w^-_{j,i} > 0$;
  - puts $u_i \cdot w^+_{i,k}$ tokens into each its output place p_k, $w^+_{i,k} > 0$;
- The net halts if firable transitions are absent.

When a transition, having a single regular incoming arc with multiplicity a from place p and a single regular outgoing arc with multiplicity b to place p', fires with $u_i = v_i$, it implements the following computations μ(p)=μ(p)mod a; $\mu(p') = \mu(p') + b \cdot (\mu(p)\ div\ a)$. Namely, it implements division by a with a remainder and multiplication by b. Choosing either a or b equal to unit we obtain either pure multiplication or pure division.

In Petri net, only one transition fires at a step while in Sawicki net (Burhard, 1981) a maximal set of firable transitions fires at a step. However in both Petri and Sawicki nets, only one instance of a transition fires at a step.

Transitions could be thought of as virtual actions. The number of really started actions depends on the amount of available resources represented by transitions' input places. Why we should restrict the number of transitions' instances to unit and fire them in sequence, as in Petri net, when available resources allow firing them simultaneously? Anyway, classical sequential order of transitions' firing could be obtained as a special case attaching a place to each transition connected with read arc and having marking equal to unit.

Various extensions of Petri nets are known such as priority, inhibitor, timed, loaded (colored), hierarchical, and nested Petri nets (Murata, 1989). Sometimes they sophisticate the basic model considerably. Our goal consists in obtaining hyper-performance at the cost of minimal modification which consist in the multiple firing