Chapter 122
Sleptsov Net Computing

Dmitry A. Zaitsev
International Humanitarian University, Ukraine

ABSTRACT

Motivation for new models of hyper-computations was presented. Sleptsov net was introduced compared to Petri and Salwicki nets. A concept of universal Sleptsov net, as a prototype of a processor in Sleptsov net computing, was discussed. Small universal Sleptsov net that runs in polynomial time was constructed; it consists of 15 places and 29 transitions. Principles of programming in Sleptsov nets, as composition of reverse control flow and data, have been developed. Standard control flow patterns include sequence, branching, loop, and parallel execution. Basic modules, which implement efficiently copying, logic, and arithmetic operations, have been developed. Special dashed arcs were introduced for brief specification of input and output data of modules (subnets). Ways of hierarchical composition of a program via substitution of a transition by a module were discussed. Examples of Sleptsov net programs for data encryption, fuzzy logic, and partial differential equations have been presented. Enterprise implementation of Sleptsov net programming promises ultra-performance.

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.

DOI: 10.4018/978-1-5225-7598-6.ch122

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*).

The mapping W: $(P\times T)\to N\cup\{-1\}$, $(T\times P)\to N$ 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: $P\to N$ 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

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