Some remarks on Petri Net Computers: Weak, Exact, and Strong

Dmitry Zaitsev

http://member.acm.org/~daze

For Petri nets, Rabin introduced a concept of a *weak Petri net computer*, to prove that the reachability set inclusion/equivalence problem is undecidable [1], which computes a result equal to or less than the required. For instance, a weak multiplier, shown in Fig. 3, computes $z \leq x \cdot y$. To compute $7 \cdot 6$ we put 7 tokens into *x* and 6 tokens into *y* and 1 token into *s*. The completion of the computations is indicated by the token arrival into *f* with result present in *z*. It can compute an exact product as well with the firing sequence $\sigma_1 = t_5(t_1 t_3^y t_2 t_4^y)^x t_6$. Though, plenty of other permitted sequences give smaller values, for instance, $\sigma_2 = t_5 t_6$ computes result equal to zero.

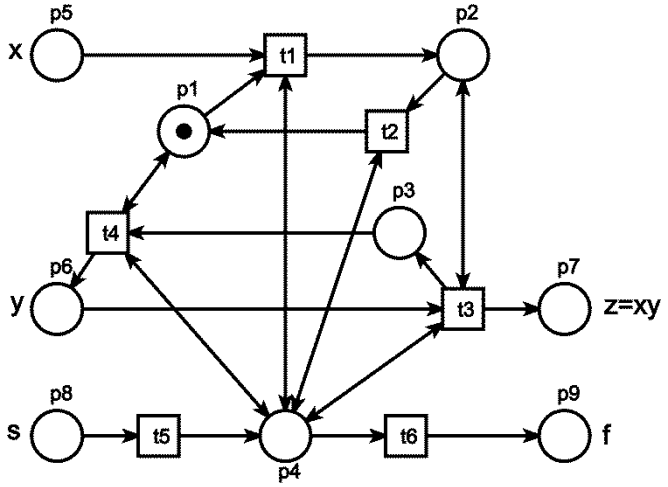

Fig. 1. A weak multiplier: $z \leq x \cdot y$ [1].

Intuitive perception of a word "computer" leads us to an *exact Petri net computer* which pure implementation is possible only in Turing-complete extensions of Petri nets, for instance in inhibitor nets [1] applied in [2] for building a universal net in an explicit form. An exact computer of $z = x \cdot y$ is represented in Fig. 2. Inhibitor arcs ensure that the only permitted sequence is σ_1.

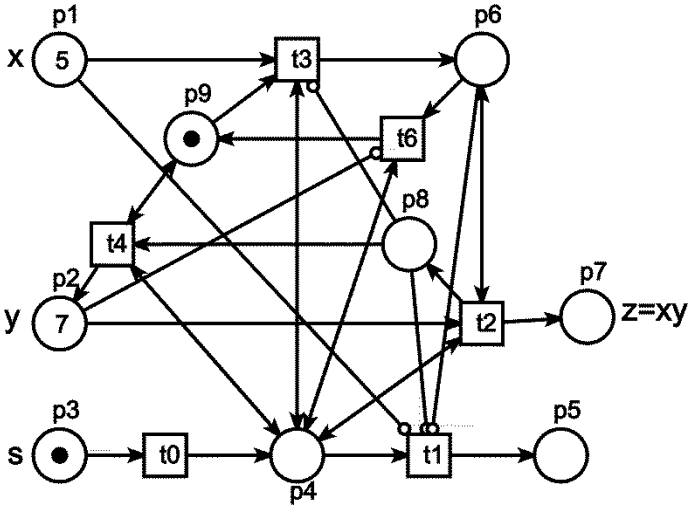

Fig. 2. An exact multiplier (inhibitor net): $z = x \cdot y$.

For conventional Petri nets exact computers exist only under certain restrictions, namely when the values of each variable is bounded, for instance by value k. In this case, the complementary places concept [1] is applied and check on zero is replaced by the check of complementary place on k. The corresponding exact multiplier is represented in Fig. 3. The net is shown with its initial marking for computing $7 \cdot 6 = 42$. We suppose that the source numbers are equal to or less than 10, then the result is bounded by the value 100. Complementary places have been created for *x, y, z*, p_2, p_3 and denoted using a prime symbol.

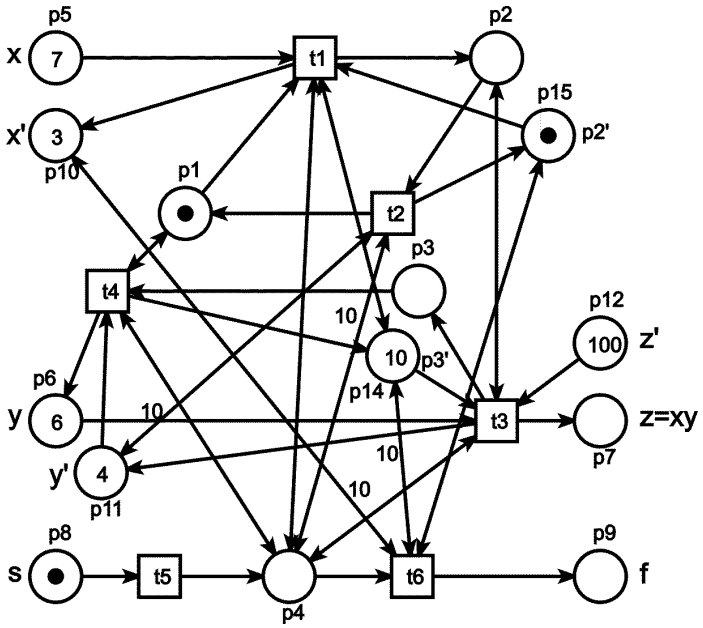

Fig. 3. An exact multiplier: $z = x \cdot y$ under $x, y \leq 10$.

Weak computers are useful in theoretical reasoning though from intuitional point of view they are no computers at all. Is it possible to make conventional Petri nets compute exact result without these unbecoming k-weighted arcs for complementary places? And we find the answer in legendary Lipton's report [4] with his concept of *a strong Petri net computer*.

In 1974 R.J. Lipton presented a construct [3] that proves that Petri net reachability problem has exponential space complexity. He introduced concepts of a parallel program and a strong computer. Lipton's parallel program contains four statements: start, accept, guess, and assign. A parallel program is a strong computer of a predicate when there is a reachable state with true/false result iff the predicate value is true/false, respectively. He stated that the acceptance problem for parallel program is reducible in polynomial time to the reachability problem for vector addition systems i.e. Petri nets.

Thorough interpretation of Lipton's results regarding composition of corresponding Petri net has been given in J. Esparza paper [4]. Finally, he composed programs of four basic routines INC, TEST, and DEC and sequences of their direct and recursive calls. Though no explicit Petri net has been constructed, the routines allowed thorough and clear proof of Lipton's theorems.

Here we construct explicit Petri nets which compute double exponent according to [3] using routines of Esparza [4]. These nets play a key part in Lipton's proof when composing nets which implement predicates checking whether variables values equal zero. The size of a net which strongly computes the double exponent 2^{2^k} depends linearly on the parameter k. Our program *depn* [5], for a given k, generates a Petri net which is a strong computer of 2^{2^k} . The net represents a sequence of k blocks

connected according to the scheme shown in the ***Cover Picture*** with a block enlarged. A block composes routines $INC_i, TST\_y_{i-1}, TST\_x_{i-1}, DEC_{i-1}$ and has size about 50 places and 50 transitions; the routines are represented separately in Fig. 4-6, respectively. Tests on zero TST_y_{i-1}, TST_x_{i-1} differ only in using variables x_i and x_i' instead of y_i and y_i'.

A block contains three pairs of places with their compliments x_i and x_i', y_i and y_i', s_i and s_i'. Subnet INC_{i+1} multiplies $x_i = 2^{2^i}$ by $y_i = 2^{2^i}$ and stores the obtained result $2^{2^i} \cdot 2^{2^i} = 2^{2^i+2^i} = 2^{2\cdot 2^i} = 2^{2^{i+1}}$ into the next block variables x_{i+1}, y_{i+1}, s_{i+1}'. Subnets TSTy_i, TSTx_i test whether y_i or x_i, respectively, equals zero. For testing a variable, the subnets try to decrement the variable value by 2^{2^i} with subnet DEC_i which uses recursive calls of TSTy_{i-1}, TSTx_{i-1} and so on till DEC_0 is reached which represents the bottom of recursion implementing decrement by $2^{2^0} = 2$ in an explicit way. Variables s_i and s_i' are used to pass parameters to the routine DEC_i.

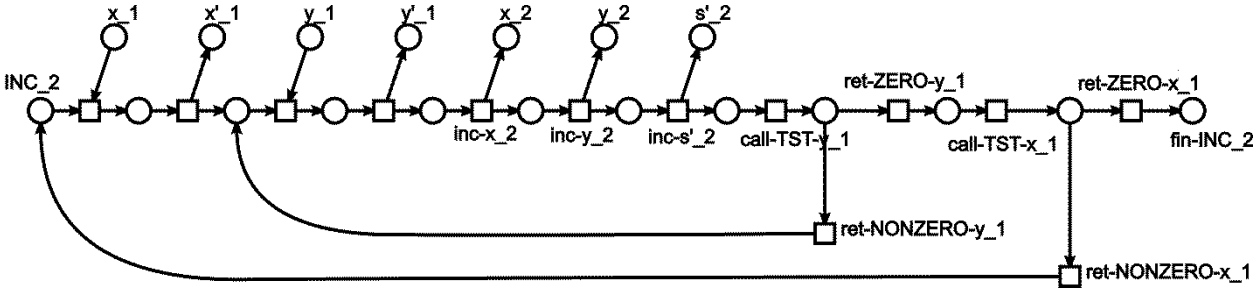

Fig. 4. Multiplication via increment INC_2.

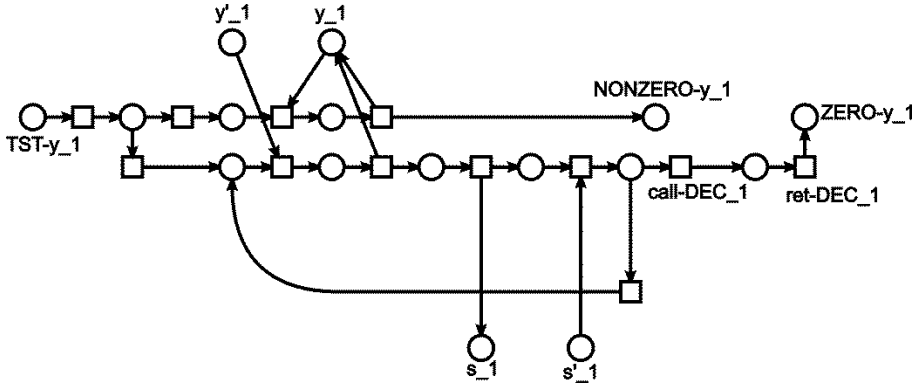

Fig. 5. Test on zero TSTy_1.

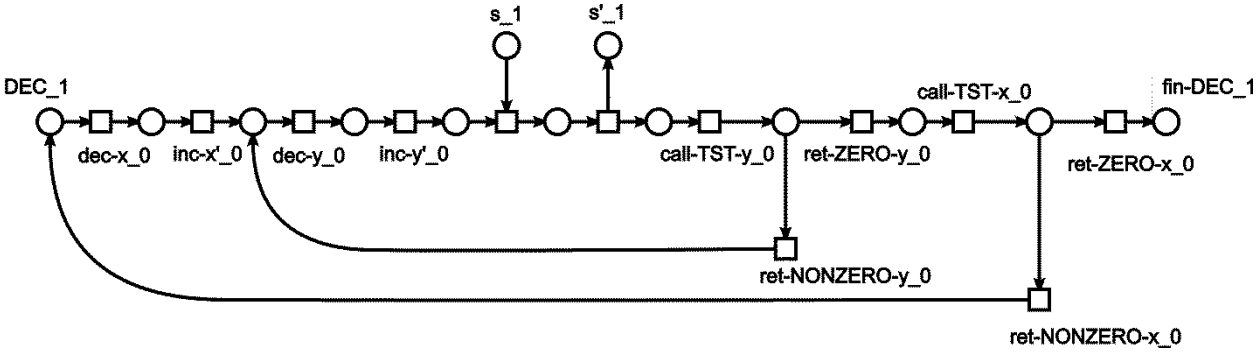

Fig. 6. Decrement by exponent DEC_1.

To grasp the concept of a strong computer, we should switch thinking of a Petri net computation as of a trace (a firing sequence of transitions) to thinking of it as of an entire reachability tree, which is considered as a nondeterministic computation. For instance, any trace of the net in Fig. 1 represents a weak computation leading to results in the range from 0 to $x \cdot y$. The net in Fig. 2 runs the only trace which gives the exact result $x \cdot y$. In Fig. 7 we show a reachability graph of the net *de2.ndr* which is a strong computer of $2^{2^2} = 16$. And the only marking with the place "fin-INC_2" (*p71*) containing a token belongs to it representing a state where the double exponent has been exactly computed:

```
3707 : p10*2 p108*16 p109*16 p110*16 p13*2 p55*4 p56*4 p57*4 p72 p8*2
```

that corresponds to $x_0 = 2$ $(p8)$, $y_0 = 2$ (p10), $s'_0 = 2$ (p13), $x_1 = 5$ $(p55)$, $y_1 = 4$ (p56), $s'_1 = 4$ (p57), $x_2 = 16$ $(p108)$, $y_2 = 16$ (p109), $s'_2 = 16$ (p110).

Thus the difference with a weak computation, that can be thought of as the entire reachability graph as well, consists in the fact that someone can recognize the exact result. The only marking containing an exact result is indicated with the marked final place. While for a weak computer any result in the range from 1 to $x \cdot y$ can be indicated by the marked final place.

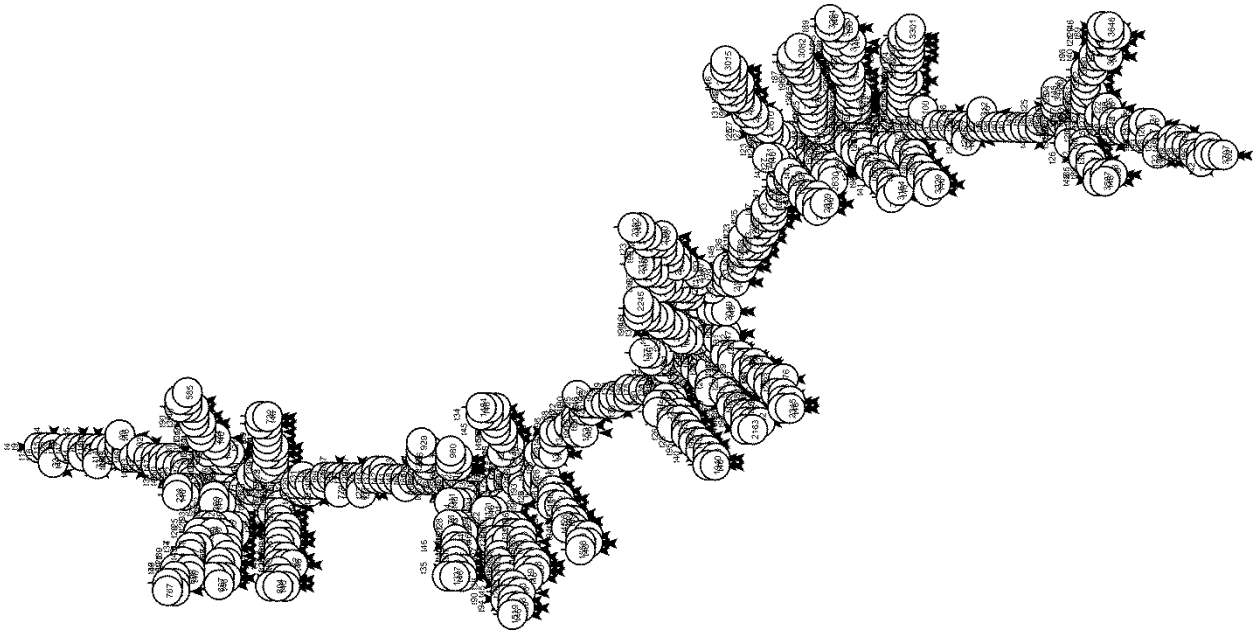

Fig. 7. Computation of the *de2.ndr* (reachability graph): $x_2, y_2 = 2^{2^2} = 16$.

To build the net for a definite value of the parameter k, say $k = 2$, we write the following command line

```
>depn 2 > de2.ndr
```

which creates file *de2.ndr* containing a Petri net strong computer of 2^{2^2} in the graphical format (.ndr) of modeling system Tina [6] according to the technique described in [7]. Tina offers either graphical tools integrated into a net editor *nd* or separate command line tools. In *nd* we can build a reachability graph with "reachability analysis" tool in verbose form and store it in a text file; then we search for "p71" in the text to find the only corresponding marking.

Brief on-fly check without storing the entire reachability graph is implemented with

```
>sift de2.ndr -f "-p71"
```

When having space bounds and an algorithm for a certain problem, we can generate the corresponding strong computer after estimating its definite space complexity for given data. From this point of view we can compose parametric specification [7] of a strong Petri net computer and, after inputting data, generate an instance of the net appropriate for the given data size. Thus, we have a universal meta-computer composed in the form of a parametric Petri net [7]. As a passage to the limit of this way of reasoning we come (from another side) to universality of infinite conventional Petri nets [8].

References

1. Peterson J.L. Petri Net Theory and the Modeling of Systems, Prentice Hall, USA, 1981.
2. Zaitsev D.A. Universal Petri net, Cybernetics and Systems Analysis, Volume 48, Number 4 (2012), 498-511.
3. Lipton R.J. The Reachability Problem Requires Exponential Space, Technical Report 63, Yale University, 1976.
4. Esparza J. Decidability and Complexity of Petri Net Problems - An Introduction. LNCS 1491, 1996, 374-428.
5. Zaitsev D.A. Generator of Petri nets which count double exponent 2^2^k after R.J.Lipton & J.Esparza constructs, 16.06.2016, https://github.com/dazeorgacm/depn
6. Berthomieu B. Time Petri Net Analyzer, http://projects.laas.fr/tina/
7. Zaitsev D.A., Zaitsev I.D., Shmeleva T.R. Infinite Petri Nets as Models of Grids (pp. 187-204). Chapter 19 in Mehdi Khosrow-Pour (Ed.) Encyclopedia of Information Science and Technology, Third Edition (10 Volumes). IGI-Global: USA, 2014.
8. Zaitsev D.A. Universality in Infinite Petri Nets. Proceedings of 7th International Conference, MCU 2015, Famagusta, North Cyprus, September 9-11, 2015, LNCS 9288, 180-197.