Constructing Petri Net Transducers with PNT$_{\varepsilon}^{\text{ooL}}$

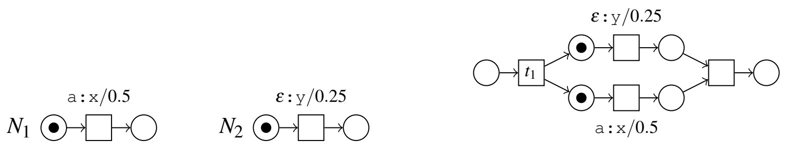
Markus Huber and Robert Lorenz

Department of Computer Science
University of Augsburg, Germany
firstname.lastname@informatik.uni-augsburg.de

Abstract This poster presents a tool for the modular construction of Petri net transducers by means of several composition operations as for example union, concatenation, closure, parallel and synchronous product and language composition. There are exports to PNML and several graphical output formats.
For the implementation, we use the SNAKES framework, which is a Python library supporting the rapid prototyping of new Petri net formalisms and provides many basic Petri net components and functionality.
In the context of our research activities, PNT$_{\varepsilon}^{\text{ooL}}$ serves as a scientific prototype for the development of an open library openPNT of efficient algorithms for the construction, composition, simulation and optimisation of PNTs which can be used in real world examples.

In [1] we introduced Petri net transducers (PNTs) and showed in [2] how they can be successfully applied in the field of semantic dialogue modelling for translating utterances into meanings. In [3] we presented a basic theoretical framework of PNTs.

In short PNTs are a formalism for the weighted translation of labelled partial orders (LPOs), which are words consisting not of a total order between their symbols but of a partial order. In this sense they are a generalisation of weighted finite state transducers (FSTs) translating words.



(a) A transducer N_1 and a generator N_2. (b) Parallel product of N_1 and N_2 after t_I has fired.

Figure 1. Some simple PNTs.

In Figure 1 one can see on the left side two simple PNTs called N_1 and N_2. Like for weighted FSTs input symbols, output symbols and weights are annotated to the transitions. So the transition of N_1 reads the symbol a and writes the symbol x (with weight 0.5) – thus N_1 translates the word a into the word x (with weight 0.5). The symbol ε is used to denote empty input or output. So the PNT N_2 is a generator which produces the word y. The weights are elements of an algebraic structure called bisemiring [3] which extends semirings by an additional operation. Thus, a bisemiring a set equipped with three operations, namely addition, sequential multiplication and parallel multiplication, satisfying certain consistency properties (for example the existence of neutral elements). Addition is used to combine the weights of alternative LPO-runs of a PNT, sequential multiplication of weights is used for sequentially composed LPOs and the weights of parallel composed LPOs are parallel multiplied. In the examples we use the extended Viterbi semiring $([0,1], max, \cdot, min)$.

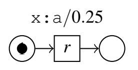
As for FSTs there exist composition operations for combining simple transducers to more complex ones. For example, Figure 1 shows on the right side the parallel product of the PNTs – an operation which does not exist for FSTs. In figures, we omit annotations of the form $\varepsilon{:}\varepsilon/1$ where $\overline{1}$ is the neutral weight w.r.t. sequential and parallel multiplication. The PNT realises the translation from the word `a` into the LPO `w=x`$\|$`y` where $\|$ denotes the parallel composition of LPOs. There are more operators defined for PNTs namely concatenation, union, closure, synchronous product and language composition of PNTs [3,2]. In Figure 2 on the right side the language composition of the PNT N_3 on the left side with the PNT shown on the right side of Figure 1 is illustrated. Here, a transition from the first PNT is merged with a transition from the second PNT, if the output of the first transition equals the input of the second one. The weight of the merged transition labelled r, t_3 is computed from the weights of the transitions labelled r and t_3 using sequential multiplication.

Note that PNTs are defined to always have a single source place which holds exactly one token at the initial state. Furthermore a PNT has a single sink place and per definition only such LPO-runs are considered, which lead to a state where exactly the sink place is marked by one token (the final state). Each LPO-run translates an LPO over input symbols into an LPO over output symbols via a projection onto input symbols resp. output symbols [3].

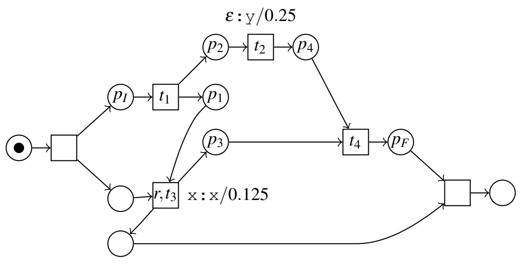
For such a formalism to be useful one needs a tool where PNTs can be implemented, analysed, combined, simulated, drawn and the like. Since the PNT-formalism is new we decided to start PNT$^{ooL}_{\varepsilon}$. We use the SNAKES framework [4] which is a Python library supporting the rapid prototyping of new Petri net formalisms and provides many basic Petri net components and functionality. Therefore we implemented PNT$^{ooL}_{\varepsilon}$ as a Python library extending SNAKES such that we essentially can use all the functionality already provided by SNAKES. All graphics in this paper were generated by PNT$^{ooL}_{\varepsilon}$.

The support of graphical output serves as a possibility to check the implementation and as a handy utility in the process of writing scientific papers. PNT$^{ooL}_{\varepsilon}$'s functionality supports fast construction of concrete example PNTs for case studies. PNML export can be used to analyse constructed example PNTs with other Petri net tools. In the context of our research activities, PNT$^{ooL}_{\varepsilon}$ serves as a scientific prototype for the development of an open library openPNT of efficient algorithms for the construction, composition, simulation and optimisation of PNTs which can be used in real world examples.

PNT$^{ooL}_{\varepsilon}$ can be downloaded as a ZIP-archive from our website `www.informatik.uni-augsburg.de/EduCoSci/PNTooL`. Assumed you have a working installation of Python, SNAKES, Graphviz, and dot2tex you only need to copy the `py`-files into the `plugins` sub-directory of your SNAKES installation.



(a) A simple PNT N_3.



(b) Language composition $N_3 \circ (N_1 \parallel N_2)$.

Figure 2. Some more PNTs.

References

1. R. Lorenz and M. Huber. Petri net transducers in semantic dialogue modelling. In M. Wolff, editor, *Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)"*, volume 64 of *Studientexte zur Sprachkommunikation*, pages 286 – 297, 2012.
2. R. Lorenz and M. Huber. Realizing the Translation of Utterances into Meanings by Petri Net Transducers. In P. Wagner, editor, *Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)"*, volume 65 of *Studientexte zur Sprachkommunikation*, pages 103 – 110, 2013.
3. R. Lorenz, M. Huber, and G. Wirsching. On weighted Petri Net Transducers. In *Proccedings of "35th International Conference on Application and Theory of Petri Nets and Concurrency"*, Lecture Notes in Computer Science. Springer, 2014.
4. F. Pommereau. The SNAKES toolkit. `https://www.ibisc.univ-evry.fr/~fpommereau/SNAKES/`, 11 2013.