Segmenting Sequences Semantically

Using Petri Net Transducers for the Translation from Sequential Data to Non-Sequential Models

Markus Huber[1,2(✉)] and Matthias Wolff[1]

[1] Brandenburg University of Technology Cottbus-Senftenberg, Germany
[2] InnoTec21 GmbH, Leipzig, Germany
markus.huber@b-tu.de

Abstract In previous work we presented an extension and generalisation of finite state transducers (FSTs) to so-called Petri net transducers (PNTs). These are applicable to any form of transforming sequential input signals into non-sequential output structures – which can be used to represent the semantics of the input – by performing a weighted relation between partial languages, i.e. assigning one weight per related input-output-pair. This paper extends the framework by an additional weighted output where every node of the structure carries a weight. Moreover the output structure induces segments on all intermediate transducers and thus on the input giving a more detailed view on how the single weight emerged during the transformation. We extend the definitions of PNTs, show the resulting changes for the operation of language composition, and discuss the impact on other PNT-algorithms. The theory is accompanied by an example of translating a keyboard input stream into a structural representation of commands to be executed.

Keywords: Petri net transducer · Weighted labelled partial order · Concurrent semiring · Natural language understanding · Cognitive systems

1 Introduction

In [16] we introduced a special form of Petri nets for the weighted translation of partial languages. We showed how *Petri net transducers* (PNTs) are a generalisation of *finite state transducers* (FSTs) [2] and discussed several operations including language composition where a cascade of transducers translates from a source language into a target language by means of intermediate languages. In [23,14,15] we proposed the use of PNTs in the field of semantic dialogue modelling for the translation of utterances into meanings. We first give a brief overview of the area of research within which our current work is developed.

Cognitive systems (including speech dialogue systems) mainly consist of three parts allowing them to interact with their environments: perceptor, behaviour control, and actuator [7]. Of these perceptor and actuator each comprise a hierarchy of transforming units and bidirectional communication between the particular levels (cf. [21]).

The responsibility of the perceptor hierarchy is translating input signals into semantic representations where the relevant parts from the input are related to semantic categories

relevant to the system. While a low-level signal – e.g. the name Wolfgang Amadeus Mozart when spoken by a person – is sequential, its semantics is, in general, non-sequential – although there is an order relation between Wolfgang and Amadeus there is no order between those and Mozart.

We use the concept of *feature-values-relations* (FVRs) [10,9] as semantic representations. In figure 1 the semantic structure for the name Wolfgang Amadeus Mozart can be seen where the semantic categories are depicted as ellipses and the parts from the signal as rectangles. The order relation between Wolfgang and Amadeus is retained by the order of their semantic categories. Although the category of middle name exists, examples as *Pippilotta Viktualia Rullgardina Krusmynta Efraimsdotter Långstrump* or *Oscar Zoroaster Phadrig Isaac Norman Henkle Emmannuel Ambroise Diggs* should motivate the need for a more general approach to repetition. Note that although the example obeys a tree-like form, FVRs are neither limited to trees nor need to be single rooted. Which semantic categories and which parts of an input signal are of relevance for a concrete system depends on its scope of action. Therefore semantics is clearly not a function of (only) the input signal and especially not a function of its syntax. By assigning weights to the nodes of an FVR it is possible to represent the confidence of each single part of the semantics as opposed to an overall weight.

Examples for non-sequential semantic representations are concept relational trees [1] which structures are constraint to binary trees, and abstract meaning representation [17] and feature structures [18] which can have a more general structure. All of them are single rooted, do not provide a nice way to preserve order information of repeated parts, and are computed from syntactical analysis. Furthermore none of them carry weights on individual nodes and only the last two have assigned an overall weight. [20] uses relational trees where connections between entities relevant for the system are represented. The authors state that 'in most cases a relation link is analogous to a syntactic dependency link' and again these tree-like structures are single rooted, do not cover the preservation of order information, and there is only a single weight assigned.

We use *labelled partial orders* (LPOs) to represent FVRs, and PNTs to represent sets of LPOs and sets of pairs of LPOs. The transducer operation of language composition allows building hierarchical bidirectional translation systems and since PNTs are a generalisation of FSTs we can utilise results from existing research in the field of speech technology [8,25] to build on. The resulting transducer cascade allows bottom-up processing of input signals to output decoded meaning, and top-down processing using prior knowledge and expectations. Systems as described in [20] all have a gap between syntax and semantics since they first translate input signals into syntactic representations and do their semantic analysis afterwards. Our approach needs no premature decisions

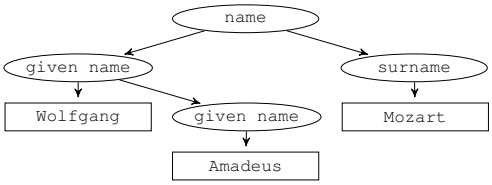

Figure 1. Semantic representation of 'Wolfgang Amadeus Mozart'.

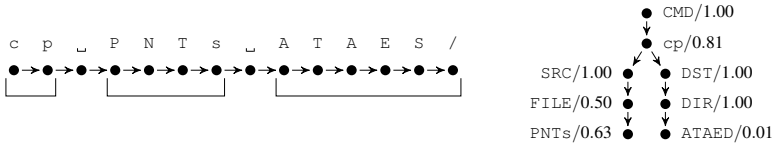

Figure 2. Segmented input sequence and weighted output structure

because we directly translate from the signal level into semantic representations. Also the priming of the perceptor hierarchy by simply appending another transducer which represents expected semantics is (to our knowledge) a unique feature.

The semantic structure – the output of the perceptor – serves as input to the behaviour control which computes the reaction of the cognitive system. These reactions also include interactions with the environment to request additional information for an incomplete structure. New input is incorporated into the already collected information. In [5] several operations on FVRs are described which admit an algebraic structure on the set of FVRs. In [20,19] a similar approach is described where relational trees are used to represent the information state of the system and some operations are described using so-called tree regular expressions. However, there is a complete lack of algebraic investigation.

The computed output of the behaviour control is again a semantic structure which gets translated by the actuator hierarchy into an output signal. According to system theory the actuator should use the same formalism as the perceptor (cf. [8]). Since PNTs provide the operation of inversion – by swapping the input and output label of each transition – they constitute a good choice. In contrast to PNTs tree transducers [12] and DAG transducers [18] are both derivation tree oriented and in general not closed under language composition and inversion. Both do not allow semantic priming in general but can be used for generation with some reservations.

Speech dialogue systems must operate under real-time conditions. There is a period of a few hundred milliseconds for analysing the input, computing the reaction, and generating the output. Systems such as the *Unified Approach to Signal Synthesis and Recognition* (UASR) [8,24] have shown that FSTs can fulfil these requirements. We are confident that PNTs are also capable of meeting these demands and are currently developing an embedded cognitive user interface for intuitive interaction with arbitrary electronic devices [3] where we already applied first results.

In previous work we also proposed a variant of PNTs capable of handling output LPOs of arbitrary width (rhPNTs [11]) which are needed to produce general structures as in figure 1. For the here presented work we do not use that extension but we will describe another enhancement of PNTs which is capable of segmenting the input (sensor signal, speech or some text) into semantic units and assigning meaningful weights – e.g. likelihoods or probabilities – to these segments. Figure 2 shows a simple example of an input text and the decoded semantics. The scenario is that a user types the command `cp PNTs ATAES/` on the keyboard trying to copy a file named `PNTs` to a folder named `ATAED`. The input – a sequence of character events – is translated into a semantic structure for executable commands including correction of the destination of the copy-command. So the system executes the user's intention not the command he typed.

The paper is organised as follows: In section 2 we recall basic definitions and introduce in section 3 the used translation formalism, always demonstrating the concepts on our example. Thereby subsections 3.1 and 3.2 reuse the definitions from [16] and subsection 3.2 extends the definitions from [11]. In section 4 we introduce the new notion of segments of weighted LPOs and extend Petri net transducers to make use of them. Afterwards, we introduce segmenting Petri net transducers fusing the concepts given so far and finally give a brief conclusion and outlook on future work in section 5.

2 Mathematical Preliminaries

By $\mathbb{N}_0$ we denote the set of *non-negative integers*, by $\mathbb{N}$ the set of *positive integers*.

The set of all *multisets* over a set X is the set ${\mathbb{N}_0}^X$ of all functions $f\colon X \to \mathbb{N}_0$. Addition $+$ on multisets is defined by $(m+m')(x) = m(x)+m'(x)$. The relation $\leq$ between multisets is defined through $m \leq m' :\Leftrightarrow \exists m''(m+m'' = m')$. We write $x \in m$ if $m(x) > 0$. A set $A \subseteq X$ is identified with the multiset m satisfying $(m(x) = 1 \Leftrightarrow x \in A) \wedge (m(x) = 0 \Leftrightarrow x \notin A)$. A multiset m satisfying $m(a) > 0$ for exactly one element a we call *singleton multiset* and denote it by $m(a)a$.

Given a binary relation $R \subseteq X \times Y$ over the sets X, Y and sets $A \subseteq X$ and $B \subseteq Y$, we denote the *image of* A by $R(A) = \{y \in Y \mid \exists x \in A\colon (x,y) \in R\}$ and the *preimage of* B by $R^{-1}(B) = \{x \in X \mid \exists b \in B\colon (x,b) \in R\}$. We denote by $\mathrm{Dom}(R) = R^{-1}(Y)$ the *domain of* R and call $R(X)$ the *image of* R. For $X' \subseteq X$ and $Y' \subseteq Y$ the restriction of R onto $X' \times Y'$ is denoted by $R|_{X' \times Y'}$.

Given a binary relation $R \subseteq X \times Y$ and a binary relation $S \subseteq Y \times Z$ for sets X, Y, Z, their composition is defined by $R \circ S = \{(x,z) \mid \exists y \in Y((x,y) \in R \wedge (y,z) \in S)\} \subseteq X \times Z$. For a binary relation $R \subseteq X \times X$ over a set X, we denote $R^1 = R$ and $R^n = R \circ R^{n-1}$ for $n \geq 2$. The symbol R^+ denotes the *transitive closure* $\bigcup_{n \in \mathbb{N}} R^n$ of R.

Let $\mathscr{A}$ be a finite set called an *alphabet*. A *word* over $\mathscr{A}$ is a finite sequence of symbols from $\mathscr{A}$. For a word w its length $|w|$ is the number of its symbols. The symbol ε denotes the *empty word* satisfying $|\varepsilon| = 0$ and is the neutral w.r.t. concatenation of words: $w\varepsilon = \varepsilon w = w$. By $\mathscr{A}^*$ we denote the set of all words over $\mathscr{A}$, including the empty word. A *step over* $\mathscr{A}$ is a multiset over $\mathscr{A}$. A *step sequence* over $\mathscr{A}$ is an element of $({\mathbb{N}_0}^{\mathscr{A}})^*$.

A *directed graph* is a pair $G = (V, \to)$, where V is a finite *set of nodes* and $\to \subseteq V \times V$ is a binary relation over V, called the *set of edges*. For a node $v \in V$ its *preset* is the set ${}^\bullet v = {\to}^{-1}(\{v\})$ and its *postset* the set $v^\bullet = \to(\{v\})$. A *path* is a sequence of (not necessarily distinct) nodes $v_1 \ldots v_n$ $(n > 1)$ such that $v_i \to v_{i+1}$ for $i = 1, \ldots, n-1$. A path $v_1 \ldots v_n$ is a *cycle* if $v_1 = v_n$. A directed graph is called *acyclic* if it has no cycles. For an acyclic directed graph $G = (V, \to)$ its *maximal nodes* is the set $\mathrm{Max}(G) = \{v \in V \mid v^\bullet = \emptyset\}$, its *minimal nodes* is the set $\mathrm{Min}(G) = \{v \in V \mid {}^\bullet v = \emptyset\}$. An acyclic directed graph $(V, \to')$ is an *extension* of an acyclic directed graph $(V, \to)$ if $\to \subseteq \to'$.

An *irreflexive partial order* over a set V is a binary relation $< \subseteq V \times V$ which is irreflexive $(\forall v \in V\colon v \not< v)$ and transitive $(<=<^+)$. A *reflexive partial order* over a set V is a binary relation $\leq \subseteq V \times V$ which is reflexive $(\forall v \in V\colon v \leq v)$, transitive $(\leq = \leq^+)$ and antisymmetric $(\forall v, w \in V\colon v \leq w \wedge w \leq v \implies v = w)$. We identify a finite partial order $\sim$ over V with the directed graph $(V, \sim)$. Given a partial order $po = (V, \sim)$ we call

two nodes $v, v' \in V$ *independent* if $v \not< v'$ and $v' \not< v$. By $\mathrm{co}_\sim \subseteq V \times V$ we denote the set of all pairs of independent nodes of V.

A *monoid* is a triple $(S, *, n)$ where S is a set, $*$ is a binary closed operation on S ($\forall a, b \in S\colon\ a * b \in S$), and n is the neutral w.r.t. $*$ ($\forall a \in S\colon\ a * n = a = n * a$). The operation is often written by juxtaposition ($ab = a * b$). If $*$ is idempotent ($\forall a \in S\colon\ aa = a$) the monoid is called *idempotent*. If $*$ is commutative ($\forall a, b \in S\colon\ ab = ba$) the monoid is called *commutative*. If there exists an *absorbing* element $\overline{0} \in S$ ($\forall a \in S\colon\ a\overline{0} = \overline{0} = \overline{0}a$) the monoid is called a *monoid with zero*. A monoid is called *ordered* if it is equipped with a reflexive partial order $\leq$ such that the operation $*$ is monotone ($\forall a, b \in S\colon\ a \leq b \implies (\forall c \in S\colon\ c * a \leq c * b \wedge a * c \leq b * c)$).

For a given monoid $(S, *, n)$ the operation $*$ defines a binary relation on S via $a \leq_* b :\Leftrightarrow ab = b$. If this relation is a reflexive partial order, the monoid is called *naturally ordered*.

An idempotent and commutative monoid $(S, *, n)$ is naturally ordered. Moreover, if S is equipped with the reflexive partial order $\leq_*$, then $\forall a, b \in S\colon\ ab = \sup\{a, b\}$, where the supremum is taken w.r.t. $\leq_*$.

3 Input and Output

We use LPOs extended by weights from an algebraic structure called bisemiring to represent the input and output structures as depicted in figure 2 and use PNTs by the means of non-sequential semantics of Petri nets for the weighted translation of LPOs.

3.1 Weighting of Structures

Except for definition 2 this subsection equates the state from [16].

A *labelled partial order* (*LPO*) *over a set* X is a 3-tuple $(V, <, l)$, where $(V, <)$ is an irreflexive partial order and $l\colon V \to X$ is a labelling function on V. In most cases, we only consider LPOs up to isomorphism, i.e. only the labelling of nodes is of interest, but not their names. Two LPOs $(V, <, l)$ and $(V', <', l')$ are *isomorphic* if there is a bijective renaming function $I\colon V \to V'$ satisfying $l(v) = l'(I(v))$ and $v < w \Leftrightarrow I(v) <' I(w)$. In figures, we only show the labels of nodes, and normally omit transitive arrows. If an LPO *lpo* is of the form $(\{v\}, \emptyset, l)$, then it is called a *singleton LPO*. A *step-wise linear LPO* is an LPO $(V, <, l)$ where the relation $\mathrm{co}_<$ is transitive. The maximal sets of independent nodes are called *steps*. The steps of a step-wise linear LPO are linearly ordered. Thus, step-wise linear LPOs can be identified with step sequences. A *step-linearisation* of an LPO *lpo* is a step-wise linear LPO *lpo*$'$ which is an extension of *lpo*.

The set of *series-parallel* LPOs (sp-LPOs) is the smallest set of LPOs containing all singleton LPOs (over a set X) and being closed under sequential and parallel product. For two LPOs $lpo_1 = (V_1, <_1, l_1)$ and $lpo_2 = (V_2, <_2, l_2)$, where $V_1 \cap V_2 = \emptyset$ is assumed, their *sequential product* is defined by $lpo_1 ; lpo_2 = (V_1 \cup V_2, <_1 \cup <_2 \cup V_1 \times V_2, l_1 \cup l_2)$ and their *parallel product* is defined by $lpo_1 \parallel lpo_2 = (V_1 \cup V_2, <_1 \cup <_2, l_1 \cup l_2)$. For an LPO *lpo* we denote by $\mathrm{SP}(lpo)$ the set of all series-parallel extensions of *lpo* and by $\mathrm{SP}_{\min}(lpo)$ the set of all minimal series-parallel extensions of *lpo* in $\mathrm{SP}(lpo)$. If *lpo* is an extension of *lpo*$'$, we write $lpo \leq lpo'$.

A *semiring* is a quintuple $\mathscr{S} = (S, \oplus, \otimes, \overline{0}, \overline{1})$, where $(S, \oplus, \overline{0})$ is a commutative monoid, $(S, \otimes, \overline{1})$ is a monoid with zero where $\overline{0}$ is the absorbing element, and $\otimes$ (the *S-multiplication*) distributes over $\oplus$ (the *S-addition*) from both sides. If $\otimes$ is commutative, then the semiring is called *commutative*.

A *bisemiring* is a six-tuple $\mathscr{S} = (S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$, where $(S, \oplus, \otimes, \overline{0}, \overline{1})$ is a semiring and $(S, \oplus, \boxtimes, \overline{0}, \overline{1})$ is a commutative semiring. The binary operation $\boxtimes$ on the set S is called *S-parallel multiplication*. If $\oplus$ is idempotent, the bisemiring is called *idempotent*.

A *concurrent semiring* is an idempotent bisemiring $(S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$ satisfying

$$\forall a, b, c, d \in S\colon (a \boxtimes b) \otimes (c \boxtimes d) \leq_{\oplus} (a \otimes c) \boxtimes (b \otimes d). \qquad \textbf{(CS)}$$

A *weighted LPO* (wLPO) over an alphabet $\mathscr{A}$ and a bisemiring $\mathscr{S} = (S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$ is a quadruple $(V, <, l, \nu)$ such that $(V, <, l)$ is an LPO over $\mathscr{A}$ and $\nu\colon V \to S$ is an additional *weight function*. We use all notions introduced for LPOs also for wLPOs.

The weight of sp-wLPOs is defined in the obvious way. The total weight is computed from the weights of the nodes through applying $\otimes$ to the sequential product and $\boxtimes$ to the parallel product of sub-wLPOs. This is well-defined, since the set of sp-wLPOs as well as the sub-structure $(S, \otimes, \boxtimes)$ of a bisemiring $(S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$ form an sp-algebra admitting an sp-algebra homomorphism from the set of sp-wLPOs into the bisemiring. For an sp-wLPO *wlpo* its weight is denoted by $\omega(wlpo)$.

The weight for general wLPOs is computed as the sum of the weights of all its series-parallel extensions. Condition (**CS**) ensures that less restrictive wLPOs yield bigger weights. So in the case of using concurrent semirings it suffices to consider only minimal series-parallel extensions.

Definition 1 (sp-Weight of wLPOs). *Let $wlpo = (V, <, l, \nu)$ be a wLPO over a concurrent semiring. Then its* sp-weight *is defined by $\omega_{\mathrm{sp}}(wlpo) = \bigoplus_{wlpo' \in \mathrm{SP}_{\min}(wlpo)} \omega(wlpo')$.*

Figure 3 shows on the left the semantic structure from our example. The weights are drawn from the concurrent semiring $([0,1], \max, \cdot, \cdot, 0, 1)$ – a trivial extension of the Viterbi semiring. The sp-weight is easy to compute since the structure is an sp-wLPO.

For wLPOs over the same bisemiring having isomorphic underlying LPOs we provide a new sum-operation. The result is again a wLPO with an isomorphic underlying LPO but all nodes carry the sum of the weights of their corresponding nodes.

Definition 2 (Sum of wLPOs). *Let $wlpo = (V, <, l, \nu)$ and $wlpo' = (V', <', l', \nu')$ be two wLPOs over the bisemiring $\mathscr{S} = (S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$ having isomorphic underlying LPOs and let $I\colon V \to V'$ be the corresponding bijective renaming function. Then their* sum *is defined by $wlpo \oplus wlpo' = (V, <, l, \nu \oplus (\nu' \circ I))$, where the sum of the weight functions is declared pointwise.*

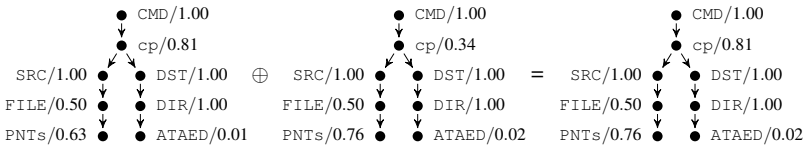

Figure 3. Semantic structure as wLPO and sum of wLPOs

Figure 3 also shows a second wLPO, representing the same semantics, which get summed up with the first one. The new weights equal the maximum of the original weights.

3.2 Translation of Structures

We use the concept of transducers to translate between LPOs by augmenting weighted transitions with input and output symbols. A run is represented as a wLPO over the transitions which can then be projected onto the input resp. output symbols. This subsection equates the states from [16,11] and extends the notions from [11] with weights. Additionally, definitions 5, 7 and 8 provide formalisations of some concepts from [16,11].

A *place/transition Petri net* (*PT-net*) is a 4-tuple $N = (P,T,F,W)$, where P is a finite set of *places*, T is a finite set of *transitions* disjoint from P, $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation* and $W\colon (P \times T) \cup (T \times P) \to \mathbb{N}_0$ is a *flow weight function* satisfying $W(x,y) > 0 \Leftrightarrow (x,y) \in F$. A *marking* of a PT-net assigns to each place $p \in P$ a number $m(p) \in \mathbb{N}_0$ of tokens, i.e. a marking is a multiset over P representing a distributed state. A *marked PT-net* is a PT-net $N = (P,T,F,W)$ together with an *initial marking* m_0. For (transition) steps τ over T we introduce the two multisets of places ${}^\bullet\tau(p) = \sum_{t \in T} \tau(t)W(p,t)$ and $\tau^\bullet(p) = \sum_{t \in T} \tau(t)W(t,p)$. A transition step τ *can occur* in m if $m \geq {}^\bullet\tau$. If τ occurs in m, the resulting marking m' is defined by $m' = m - {}^\bullet\tau + \tau^\bullet$. We write $m \xrightarrow{\tau} m'$ to denote that τ can occur in m and its occurrence leads to m'. A *step execution in m* is a finite step sequence over T $\tau_1 \dots \tau_n$ such that there are markings $m_1, \dots, m_n$ with $m \xrightarrow{\tau_1} m_1 \xrightarrow{\tau_2} \dots \xrightarrow{\tau_n} m_n$.

We use LPOs over T to represent single non-sequential runs of PT-nets, i.e. the labels of an LPO represent transition occurrences. For a marked PT-net $N = (P,T,F,W,m_0)$ an LPO $lpo = (V,<,l)$ over T is an *LPO-run* if each step-linearisation of lpo is a step execution of N in m_0. If an LPO-run $lpo = (V,<,l)$ occurs in a marking m, the resulting marking m' is defined by $m' = m - \sum_{v \in V} {}^\bullet l(v) + \sum_{v \in V} l(v)^\bullet$. We denote the occurrence of an LPO-run lpo by $m \xrightarrow{lpo} m'$.

A Petri Net Transducer is a PT-net where each transition is augmented with an input and an output symbol and additionally carries a weight drawn from a bisemiring.

Definition 3 (Petri Net Transducer). *A* Petri Net Transducer *(PNT) over a bisemiring $\mathscr{S} = (S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$ is defined as a tuple $N = (P,T,F,W,p_I,p_F,\Sigma,\sigma,\Delta,\delta,\omega)$, where*

- *(P,T,F,W,m_0) with $m_0 = p_I$ is a marked PT-net (called the* underlying PT-net*), $p_I \in P$ is the* source place *satisfying ${}^\bullet p_I = \emptyset$ and $p_F \in P$ is the* sink place *satisfying ${p_F}^\bullet = \emptyset$,*
- *Σ is a set of* input symbols *and $\sigma\colon T \to \Sigma \cup \{\varepsilon\}$ is the* input mapping*,*
- *Δ is a set of* output symbols *and $\delta\colon T \to \Delta \cup \{\varepsilon\}$ is the* output mapping *and*
- *$\omega\colon T \to S$ is the* weight function*.*

A wLPO $wlpo = (V,<,l,\nu)$ over T is a wLPO-run *of N*

- *if the underlying LPO $lpo = (V,<,l)$ is an LPO-run of N with $p_I \xrightarrow{lpo} p_F$ and*
- *if $\nu(v) = \omega(l(v))$ holds for all $v \in V$.*

We denote by wLPO(N) *the set of all wLPO-runs of* N.

A PNT can be used to translate a partial language into another partial language, relating so-called input words to so-called output words. Input and output words are defined as LPOs $(V,<,l)$ with a labelling function $l\colon V \to \mathscr{A} \cup \{\varepsilon\}$ for some input or output alphabet $\mathscr{A}$. Such LPOs we call *ε-LPOs*. For each ε-LPO $(V,<,l)$ we construct the *corresponding ε-free LPO* $(W,<|_{W\times W}, l|_W)$, $W = V \setminus l^{-1}(\varepsilon)$ by deleting ε-labelled nodes together with their adjacent edges. Since partial orders are transitive, this does not change the order between the remaining nodes.

Definition 4 (Input and Output Labels of Runs). *Let* $N = (P,T,F,W,p_I,p_F,\Sigma,\sigma,\Delta,\delta,\omega)$ *be a PNT and let* $wlpo = (V,<,l,\nu) \in$ wLPO(N). *The* input label *of wlpo is the LPO* $\sigma(wlpo)$ *corresponding to the ε-LPO* $(V,<,\sigma\circ l)$. *The* output label *of wlpo is the LPO* $\delta(wlpo)$ *corresponding to the ε-LPO* $(V,<,\delta\circ l)$.

For LPOs u over Σ and v over Δ, we denote by wLPO(N,u) *the subset of all wLPOs wlpo from* wLPO(N) *with input label* $\sigma(wlpo) = u$, *and by* wLPO(N,u,v) *the subset of all wLPOs from* wLPO(N,u) *with output label* $\delta(wlpo) = v$.

The *input language* of a PNT is the set of all input labels of its weighted LPO-runs. Its elements are called *input words*. *Output language* and *output words* are defined analogously.

A PNT assigns weights to all pairs of LPOs u over Σ and v over Δ based on the weights of its wLPO-runs (cf. [16,13]). Concerning the semantics, only the input output behaviour of PNTs is relevant. Since transitions also may have empty input and/or empty output, there are always (infinitely) many PNTs having the same semantics. For practical application, such PNTs are equivalent (cf. [16]).

For our example the PNT N_w generating the output word `cp PNTs ATAED/` is shown in the upper part of figure 4 on the following page. Note that the symbols themselves are sequences of symbols and the typo of the user was corrected. This was possible since the system already did several translations of the original input sequence. First the sequence was translated into a set of sequences where every input symbol created alternatives according to the keyboard layout. If you type an `S` on a QWERTZ keyboard it could be that your intention was to type a `D` since the two keys are neighboured. Figure 5 on the next page shows a PNT which realises this correction for the letter `S`. The next step was translating from the many sequences of characters into sequences of known words. The system could filter out many sequences since it knows only about some commands, files and folders. Since there was no folder named `ATAES` but there exists a folder `ATAED`, the shown sequence of words was generated. The PNT N_o in the lower part translates between the two words depicted below of it. Since there is a file `PNTs` and also a folder with the same name, the weight for `FILE` is also lesser than 1.

Note that the input and output languages of a PNT N are extension closed, since wLPO(N) is extension closed. This allows N_o from figure 4 to also accept the two step-linearisations `cp PNTs ATAED/` and `cp ATAED/ PNTs` as input. Indeed these would be translated into step-linearisations of the depicted output word with an additional edge between `PNTs` and `ATAED`.

We now introduce the central transducer composition operation of language composition. It allows to put several transducers in a row to translate from the input of the

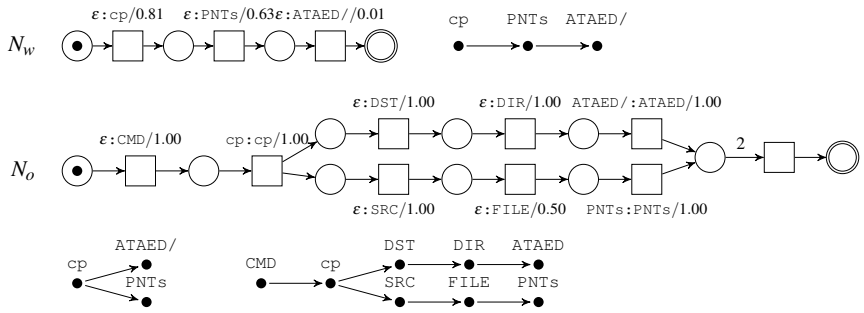

Figure 4. Two PNTs with input and output words

first one to the output of the last one utilising the intermediate ones. We consider a fixed concurrent semiring $\mathscr{S} = (S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$ and a PNT N_a over $\mathscr{S}$ with input alphabet Σ_a and output alphabet Δ_a and a PNT N_b over $\mathscr{S}$ with input alphabet $\Sigma_b = \Delta_a$ and output alphabet Δ_b. We assume N_a and N_b to be disjoint.

We define a binary relation $\stackrel{\circ}{=} \subseteq T_a \times T_b$ on the Cartesian product of the transitions from N_a and N_b by $t_a \stackrel{\circ}{=} t_b :\Leftrightarrow \delta_a(t_a) = \sigma_b(t_b)$ and say that t_a and t_b build a *composable pair*. Any transitions $t_a \in T_a$ with empty output ($\delta_a(t_a) = \varepsilon$) and $t_b \in T_b$ with empty input ($\sigma_b(t_b) = \varepsilon$) are called *non-invasive*. Any invasive transition which is not part of a composable pair is said to be *futile*.

The construction corresponds to the parallel product of N_a and N_b and merging each composable pair of transitions (t_a, t_b) to a new transition with input symbol $\sigma_a(t_a)$ and output symbol $\delta_b(t_b)$, weight $\omega_a(t_a) \boxtimes \omega_b(t_b)$ and connections ${}^\bullet t_a + {}^\bullet t_b$ and $t_a{}^\bullet + t_b{}^\bullet$. Moreover, we keep all non-invasive transitions of N_a and N_b unchanged, and omit all other transitions of N_a or N_b, i.e. all futile transitions.

Figure 6 on the following page shows the result of the language composition of N_w and N_o from figure 4. The nodes filled with dots establish the parallel product, the greyed transitions are the result of merging transitions from composable pairs, and all white transitions are non-invasive ones from N_o. The greyed nodes also represent N_w inside the result.

To provide a formal definition of language composition we set for $i \in \{a, b\}$

– for the construction of the parallel product of N_a and N_b
  - $P_\| = \{p_{\circ I}, p_{\circ F}\}$ as the new source and sink places,
  - $T_\| = \{t_{\circ I}, t_{\circ F}\}$ as the splitting and joining transitions,

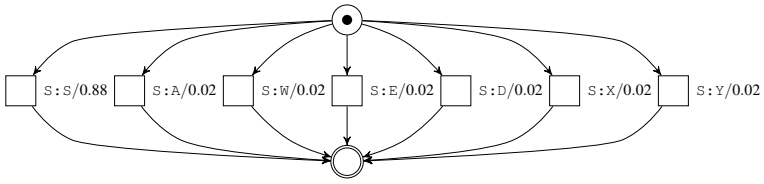

Figure 5. A PNT suggesting alternatives for a typed S

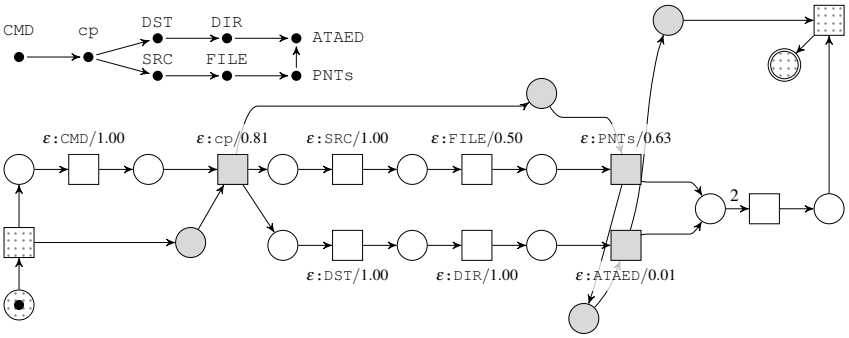

Figure 6. Language composition of N_w and N_o with output word

- $F_{\|} = \{(p_{\circ I}, t_{\circ I}), (t_{\circ I}, p_{I_a}), (t_{\circ I}, p_{I_b}), (p_{F_a}, t_{\circ F}), (p_{F_b}, t_{\circ F}), (t_{\circ F}, p_{\circ F})\}$ as the corresponding flow relation,

– for the merged transitions
  - $T^m = \mathring{=}$ using the composable pairs as new transitions,
  - $\pi_i \colon T^m \to T_i$ as the projections onto the first resp. second component of a composable pair and for any set X we also write π_i to mean (id_X, π_i) or (π_i, id_X),
  - $F_i^m = \{(p,t) \in P_i \times T^m \mid \pi_i(p,t) \in F_i\} \cup \{(t,p) \in T^m \times P_i \mid \pi_i(t,p) \in F_i\}$ to be the merged flow relation part from N_i,
  - $W_i^m = \sum_{(p,t) \in P_i \times T^m} W_i(\pi_i(p,t))\pi_i(p,t) + \sum_{(t,p) \in T^m \times P_i} W_i(\pi_i(t,p))\pi_i(t,p)$ to be the merged flow weight function part from N_i,
  - $\omega^m \colon T^m \to S$ as the merged weight function with $\omega^m(t) = \omega_a(\pi_a(t)) \boxtimes \omega_b(\pi_b(t))$,

– for the keeping of non-invasive transitions
  - T_i^n to be the non-invasive transitions from N_i,
  - $F_i^n = F_i|_{(P_i \times T_i^n) \cup (T_i^n \times P_i)}$ as the non-invasive flow relation of N_i.

Definition 5 (Language Composition). *Let $N_a = (P_a, T_a, F_a, W_a, p_I, p_F, \Sigma_a, \sigma_a, \Delta_a, \delta_a, \omega_a)$ and $N_b = (P_b, T_b, F_b, W_b, p_I, p_F, \Sigma_b, \sigma_b, \Delta_b, \delta_b, \omega_b)$ with $\Delta_a = \Sigma_b$ be two PNTs over the same concurrent semiring $\mathscr{S} = (S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$. Then, using the notations from above, the* language composition *$N_a \circ N_b$ is the PNT $N = (P, T, F, W, p_I, p_F, \Sigma, \sigma, \Delta, \delta, \omega)$ over $\mathscr{S}$, where*

– $P = P_{\|} \cup P_a \cup P_b$ *and* $T = T_{\|} \cup T^m \cup T_a^n \cup T_b^n$,
– $F = F_{\|} \cup F_a^m \cup F_b^m \cup F_a^n \cup F_b^n$,
– $W|_{F_{\|}} \equiv 1$, $W|_{F_a^m} \equiv W_a^m$, $W|_{F_b^m} \equiv W_b^m$, $W|_{F_a^n} \equiv W_a$, $W|_{F_b^n} \equiv W_b$,
– $p_I = p_{\circ I}$ *and* $p_F = p_{\circ F}$,
– $\Sigma = \Sigma_a$ *and* $\Delta = \Delta_b$,
– $\sigma|_{T_{\|}} \equiv \varepsilon$, $\sigma|_{T^m} \equiv \sigma_a \circ \pi_a$, $\sigma|_{T_a^n} \equiv \sigma_a$, $\sigma|_{T_b^n} \equiv \varepsilon$,
– $\delta|_{T_{\|}} \equiv \varepsilon$, $\delta|_{T^m} \equiv \delta_b \circ \pi_b$, $\delta|_{T_a^n} \equiv \varepsilon$, $\delta|_{T_b^n} \equiv \delta_b$,
– $\omega|_{T_{\|}} \equiv \overline{1}$, $\omega|_{T^m} \equiv \omega^m$, $\omega|_{T_a^n} \equiv \omega_a$ *and* $\omega|_{T_b^n} \equiv \omega_b$.

Since input and output languages of PNTs are extension closed it is possible that the operation of language composition propagates dependencies from one PNT to the other. Consider two PNTs N_a and N_b like in the above definition, an output word u' of N_a, an input word u of N_b with $u' \leq u$ and the output word v into which N_b translates u. Then N_b also accepts u' as input word and translates it into an output word $v' \leq v$.

This conjuncture can also be seen on the output word in figure 6 where the language composition took over the order between `PNTs` and `ATAED/` from N_w whereas the corresponding transitions from N_o were unordered.

The problem is discussed in more detail in [11] where solutions based on separating input and output processing are proposed. All those solutions have in common that the processing of weights, i.e. merging of transitions, no longer occurs where the output is produced. To overcome this we use another approach from [11] namely Hierarchical Petri Net Transducers. We extend them by augmenting the transitions with weights drawn from a bisemiring and introduce the additional concept of shared transitions.

A Hierarchical Petri Net Transducer is a PNT together with a refinement operation to substitute transitions by other PNTs. This induces sets of transitions which can be used to eliminate some edges from the input and output labels of runs by restricting the order relation. With this post processing transitions can be isolated from one another solving the problem of propagating unwanted order during language composition operation.

Definition 6 (Hierarchical Petri Net Transducer). *For a given PNT* $N_0 = (P_0, T_0, F_0, W_0, p_I, p_F, \Sigma_0, \sigma_0, \Delta_0, \delta_0, \omega_0)$ *over a bisemiring* $\mathscr{S} = (S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$ *a* hierarchical PNT *(hPNT) over* $\mathscr{S}$ *is a 6-tuple* $H = (N_0, \mathscr{N}, \rho, F_c, W_c, T_s)$*, where*

- N_0 *is the* initial PNT*,*
- $\mathscr{N} = \{N_1, \ldots, N_k\}$ *is a family of* refinement PNTs *over* $\mathscr{S}$ *whereat* $N_0, N_1, \ldots, N_k$ *are pairwise disjoint,*
- $\rho \colon T_0 \to \{1, \ldots, k\}$ *is a partial* refinement function *which is injective* ($\rho(t) = \rho(t') \implies t = t'$) *and associates transitions from the initial PNT with PNTs from* $\mathscr{N}$*. A transition* $t \in T_0$*, if* $t \in \mathrm{Dom}(\rho)$ *holds, is refined by the PNT* $N_{\rho(t)}$*. Any transition* $t \in T_0$ *for which* $t \notin \mathrm{Dom}(\rho)$ *holds is called* simple*. Any simple transition* t *must have empty input and output* $\sigma_0(t) = \varepsilon = \delta_0(t)$ *as well as neutral weight* $\omega_0(t) = \overline{1}$*.*
- $F_c \subseteq (P_0 \times \bigcup_{i=1}^{k} T_i) \cup (\bigcup_{i=1}^{k} T_i \times P_0)$ *is the* crossing flow relation *which allows to connect transitions from refinement PNTs to places from the initial PNT,*
- W_c *is the corresponding* crossing flow weight function *analogous to PT-nets and*
- $T_s \subseteq T_0$ *is the set of* shared transitions *which allow to relax the isolation of transitions.*

The crossing components are initially empty and only needed for holding information when computing the language composition of a PNT and an hPNT where the result is again an hPNT (see definition 8). The same holds for the set of shared transitions.

In figure 7 on the next page one can see the hPNT H_o where the transitions t_1, t_2, and t_3 are refined by the PNTs N_1 resp. N_2 resp. N_3.

The refinement operation is defined in an obvious way. Any non-simple transition is substituted with two new ε-transitions with neutral weight $\overline{1}$. One is called *down-going*

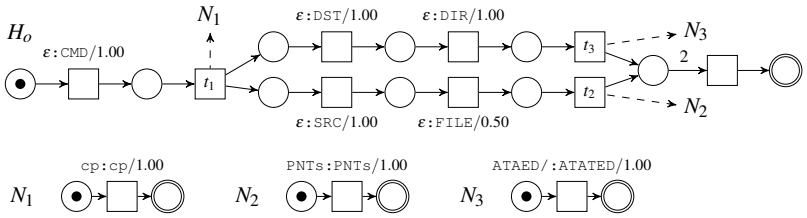

Figure 7. hPNT H_o, refinement PNTs N_i for transitions t_i ($i = 1,2,3$)

refinement transition and only inherits the incoming arcs and the other one is called *up-going refinement transition* and inherits only the outgoing arcs. The down-going refinement transition is connected to the source place of the refinement PNT and the up-going refinement transition with its sink place. For the formal procedure of refining we refer to [11]. The result of refining is a PNT N called the *interface of* H and the set of runs of N is per definition the set of runs of H. We use all other notations introduced for PNTs as well as definition 4 also for hPNTs by means of their interfaces.

The hPNT H_o in figure 7 has the same input and output words as the PNT N_o from figure 4 since the interface of H_o is equivalent to N_o.

But hierarchical PNTs have additional input and output labels where the order relation is restricted. Dependencies between different refinement PNTs are excluded except for shared transitions. Dependencies between the initial PNT and refinement PNTs other than by the refinement itself and through shared transitions are excluded as well. We focus on the output labels.

Figure 8 shows an output word of H_o where the nodes are grouped to visualise the allowed connections. The narrower lines represent the refinement paths.

We use the notations from [11] where ${}^\bullet T$ and $T^\bullet$ are the sets of down-going resp. up-going refinement transitions and set $T_r = {}^\bullet T \cup T^\bullet$ to be the set of all refinement transitions. We use $K = \rho(T_0)$ as the image of ρ and $K_0 = K \cup \{0\}$. Then we have $r = \bigcup_{i \in K_0} l^{-1}(T_i \cup T_r \cup T_s) \times l^{-1}(T_i \cup T_r \cup T_s)$ as the set of all allowed relations.

Definition 7 (Relaxed Output Label of Runs). *Let $H = (N_0, \mathscr{N}, \rho, F_c, W_c)$ be a hPNT and let $wlpo = (V, <, l, \nu) \in \mathrm{wLPO}(H)$. The* relaxed output label *of wlpo is the LPO $\delta_r(wlpo)$ corresponding to the ε-LPO $(V, <|_r{}^+, \delta \circ l)$.*

For LPOs u over Σ and v over Δ, we denote by $\mathrm{wLPO}_{O,r}(H,u,v)$ the subset of all wLPOs wlpo from $\mathrm{wLPO}(H,u)$ with relaxed output label $\delta_r(wlpo) = v$.

We consider a fixed concurrent semiring $\mathscr{S}$ and a PNT N_a over $\mathscr{S}$ with input alphabet Σ_a and output alphabet Δ_a and an hPNT H over $\mathscr{S}$ with its interface N_b and input alphabet $\Sigma_b = \Delta_a$ and output alphabet Δ_b. We assume N_a and N_b to be disjoint.

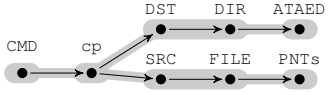

Figure 8. Output word of H_o

Using the notations introduced before definition 5 as well as the notations from above of definition 7 we additionally set

– the right partial language composition $N_a \,|\circ\, N_b$ to be the PNT $N = (P, T, F, W, p_I, p_F, \Sigma, \sigma, \Delta, \delta, \omega)$ with
  - $P = P_b$ and $T = T^m \cup T_b^n$,
  - $F = F_b^m \cup F_b^n$,
  - $W|_{F_b^m} \equiv W_b^m$, $W|_{F_b^n} \equiv W_b$,
  - $p_I = p_{I_b}$ and $p_F = p_{F_b}$,
  - $\Sigma = \Sigma_a$ and $\Delta = \Delta_b$,
  - $\sigma|T^m \equiv \sigma_a \circ \pi_a$, $\sigma|_{T_b^n} \equiv \varepsilon$,
  - $\delta|_{T^m} \equiv \delta_b \circ \pi_b$, $\delta|_{T_b^n} \equiv \delta_b$,
  - $\omega|_{T^m} \equiv \omega^m$, $\omega|_{T_b^n} \equiv \omega_b$ and
– the remaining flow relation $F(N_a \,|\circ\, N_b) = F_a^m$ as the non-invasive flow relation of N_a.

Definition 8 (Hierarchical Language Composition). *Let $N_a = (P_a, T_a, F_a, W_a, p_I, p_F, \Sigma_a, \sigma_a, \Delta_a, \delta_a, \omega_a)$ be a PNT over the concurrent semiring $\mathscr{S} = (S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$, H be an hPNT over $\mathscr{S}$, and N_b be the interface of H with $\Delta_a = \Sigma_b$. Then, using the notations from above, the* hierarchical language composition *$N_a \circ H$ is the hPNT $H' = (N_{0'}, \mathscr{N}', \rho', F_{c'}, W_{c'})$ where*

– $N_{0'} = N_a \circ N_0$,
– $\mathscr{N}' = \{N_{1'}, \ldots, N_{k'}\}$ *where* $N_{i'} = N_a \,|\circ\, N_i$ *for every* $i = 1, \ldots, k$,
– $\rho' = \rho$,
– $F_{c'} = F_c \cup \bigcup_{i=1,\ldots,k} F(N_a \,|\circ\, N_i)$,
– $W_{c'}|_{F_c} \equiv W_c$, $W_{c'}|_{F(N_a|\circ N_i)} \equiv W_a|_{F(N_a|\circ N_i)}$ *for every* $i = 1, \ldots, k$ *and*
– $T_{s'} = T_s \cup T_a^n$.

The hierarchical language composition corresponds to the language composition of the PNT and the interface of the hPNT. Technically, the above definition only states where to put the new places and transitions into.

Consider now the language composition $N_w \circ H_o$. The output word of the result would be the same as for $N_w \circ N_o$ depicted in figure 6 but the relaxed output word would be the one from figure 8 where the order between `PNTs` and `ATAED` is restricted away.

4 Adding and Spreading the Weight

The introduced framework allows for weighted relations between input and output but neither are weighted objects. After processing there is no way to see which part of a run had what impact on the result. Since we want to have a more detailed view of the weight we want to relate an input to a weighted output. Therefore we propose an additional definition of weighted output labels of runs for PNTs where the weight function is recomputed during the deletion of ε-labelled nodes. For this recomputation a wLPO

is segmented by its labels. This also has influence on the definition of equivalence for PNTs and results in technical restrictions for PNT-algorithms. Language composition of several PNTs can propagate the segments from one end to the other. The application to hPNTs leads to Segmenting Petri Net Transducers. These new ideas are introduced in two steps in the next subsections.

4.1 Segments of Weighted Structures

A wLPO is segmented by means of its labels. Every segment contains only one non-ε-labelled node but ε-labelled nodes can be part of more than one segment. The weight of a single segment is computed as its sp-weight.

For a given ε-wLPO $wlpo = (V, <, l, \nu)$ and for any subset $V' \subseteq V$ of nodes we set

- $V'_{\not\varepsilon} = \{v \in V' \mid l(v) \neq \varepsilon)\}$ to be the set of all non-ε-labelled nodes from V',
- $V'^{\varepsilon}_{V'} = V' \cup \{v \in V \setminus V_{\not\varepsilon} \mid \exists v' \in V' \colon v < v'\}$ to be the set of all nodes from V' and all their ε-labelled predecessors,
- $V_{V'} = V' \cup \{v \in V \mid \exists v' \in V' \colon v < v'\}$ to be the set of all nodes from V' and all their predecessors,
- $po_{V'} = (V', <|_{V' \times V'})$ to be the corresponding partial order on V',
- $\mathrm{Min}(V') = \mathrm{Min}(po_{V'})$ to be the set of all minimal nodes from V' and
- $\mathrm{Max}(V') = \mathrm{Max}(po_{V'})$ to be the set of all maximal nodes from V'.

Definition 9 (Segments of ε-wLPOs). *Let $wlpo = (V, <, l, \nu)$ be an ε-wLPO. Let N be a set. Then, using the notations from above, we compute the* segments $\mathrm{seg}(wlpo)$ *of $wlpo$ with the following procedure:*

1. *Initialise N with all nodes from V.*
2. *Let $w = (N, <|_{N \times N}, l|_N, \nu|_N)$ be the current ε-wLPO.*
3. *For every minimal non-ε-labelled node $l \in \mathrm{Min}(N_{\not\varepsilon})$ put $N^{\varepsilon}_{\{l\}}$ into* $\mathrm{seg}(wlpo)$.
4. *Set N to be $N \setminus N_{\mathrm{Min}(N_{\not\varepsilon})}$.*
5. *If $N_{\not\varepsilon} \neq \emptyset$ then continue with step 2.*
6. *Put N into* $\mathrm{seg}(wlpo)$.

For a non-ε-labelled node $v \in V_{\not\varepsilon}$ we say that the wLPO $\mathrm{seg}(v) = (S_v, <|_{S_v \times S_v}, l|_{S_v}, \nu|_{S_v})$, *where S_v is the one set from* $\mathrm{seg}(wlpo)$ *with* $\mathrm{Max}(S_v) = \{v\}$, *is the* segment of *$v$. The set from step 6 is called* ε-segment *of $wlpo$.*

For an ε-wLPO $(V, <, l, \nu)$ we construct the *corresponding ε-free wLPO* $(V_s, <_s, l_s, \nu_s)$, where $(V_s, <_s, l_s)$ is the corresponding ε-free LPO, and the weight function is defined by $\nu_s(v) = \omega_{\mathrm{sp}}(\mathrm{seg}(v))$ for all nodes $v \in V_s$.

Figure 9 on the following page depicts on the left an ε-wLPO which is a wLPO-run of N_o projected on its input symbols. Here the nodes are grouped to visualise the segments of the wLPO. On the right the corresponding ε-free wLPO is shown.

Note that for an ε-wLPO $wlpo$ and its corresponding ε-free wLPO $wlpo_s$ in general $\omega_{\mathrm{sp}}(wlpo) \neq \omega_{\mathrm{sp}}(wlpo_s)$ holds (even if $wlpo$ is series-parallel) since the weight of the ε-segment is not considered on the right side – which can be seen also in figure 9 – but the weight of some nodes is possibly taken into account multiple times.

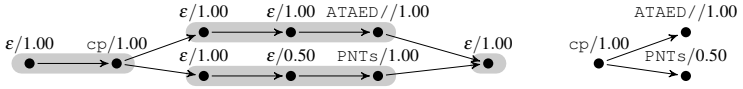

Figure 9. Segmented ε-wLPO and corresponding ε-free wLPO

4.2 Segmenting Petri Net Transducers

With the introduced notion of segments we extend the definitions for PNTs focusing on the output side. Extension of the input side is subject to further research. Eventually we introduce Segmenting Petri Net Transducers as hPNTs using the new concepts.

While the runs of PNTs are wLPOs, the input and output words of PNTs are unweighted LPOs. We add another output label resulting in weighted output words.

Definition 4 (add.) (Weighted Output Label of Runs). *Let* $N = (P,T,F,W,p_I,p_F,\Sigma,\sigma,\Delta,\delta,\omega)$ *be a PNT over a concurrent semiring* $\mathscr{S} = (S,\oplus,\otimes,\boxtimes,\overline{0},\overline{1})$ *and let* $wlpo = (V,<,l,\nu) \in \mathrm{wLPO}(N)$. *The* weighted output label *of wlpo is the wLPO* $\delta_\omega(wlpo)$ *corresponding to the* ε*-wLPO* $(V,<,\delta \circ l,\nu)$.

If different wLPO-runs result in the same output word, i.e. have isomorphic underlying LPOs, then we build the sum of all their corresponding ε-free wLPOs.

Definition 10 (Weighted Output of PNTs). *Let* $N = (P,T,F,W,p_I,p_F,\Sigma,\sigma,\Delta,\delta,\omega)$ *be a PNT over a concurrent semiring* $\mathscr{S} = (S,\oplus,\otimes,\boxtimes,\overline{0},\overline{1})$, *u be an LPO over* Σ *and v be an LPO over* Δ. *The* weighted output $N_{O,\omega}(u,v)$ *is defined by*

$$N_{O,\omega}(u,v) = \bigoplus_{wlpo \in \mathrm{wLPO}(N,u,v)} \delta_\omega(wlpo).$$

We set $N_{O,\omega}(u,v) = (\emptyset,\emptyset,l,\nu)$ *if* $\mathrm{wLPO}(N,u,v) = \emptyset$.

This definition favours the definition of the output weight of PNTs – the weight which is assigned to an input-output-pair (see definition 5 of [16]).

The *weighted output language* of a PNT is the union of all its weighted output and its elements are called *weighted output words*.

The additional semantics of PNTs should also be considered in the concept of equivalence (see definition 6 of [16]) to reflect the notion of segments. This would have a severe impact on algorithms for PNTs. Any modification of weights, input or output symbols would have to respect segments. Also ε-segments should be observed since it is not obvious how they interfere with the sequential product of PNTs.

Language composition of PNTs can propagate segments from one PNT to another. Consider a segment of the upper PNT from figure 10. It originates from one transition carrying a non-ε output symbol and a (probably empty) set of transitions with ε output symbol. All transitions can have ε or non-ε input symbols. During language composition with the lower PNT from figure 10, transitions with non-ε input symbols can get merged not changing their output symbols, or transitions from the other PNT having ε output symbols can be added. This way transitions from the lower PNT join the segments induced by the upper one. Additionally, their weight is incorporated into the segments.

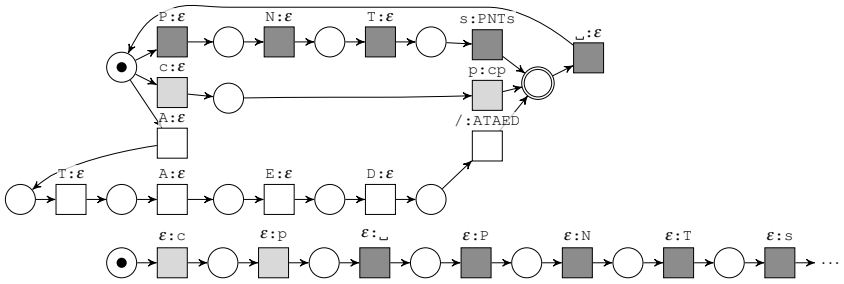

Figure 10. Propagation of segments

But segments can also be reduced or even deleted when transitions with non-ε input symbols are not carried over to the resulting PNT. Nevertheless, segments from one PNT induce a segmentation of another PNT through language composition and in general this segmentation will be different than the PNT's own one. This even raises the bar for algorithms since segments are subject to change depending on the PNT's environment.

Now we are able to translate an input signal into a weighted output structure. To address the problem of unwanted propagated order, we combine the notion of weighted output with hPNTs. Therefore the algorithm from definition 9 runs on relaxed output labels and is adapted in step 3 to only consider nodes originating from the same set of transitions or from the set of shared transitions. The modified algorithm leads to *hierarchical correspondence* of ε-wLPOs and ε-free wLPOs.

Definition 7 (add.) (Relaxed Weighted Output Label of Runs). *Let $H = (N_0, \mathcal{N}, \rho, F_c, W_c, T_s)$ be an hPNT over a concurrent semiring $\mathcal{S}$ and let $wlpo = (V, <, l, \nu) \in \mathrm{wLPO}(H)$. The* relaxed weighted output label *of wlpo is the wLPO $\delta_{r,\omega}(wlpo)$ hierarchical corresponding to the ε-wLPO $(V, <|_r{}^+, \delta \circ l, \nu)$.*

This time we build the sum of all hierarchical corresponding ε-free wLPOs of wLPO-runs leading to the same relaxed output word.

Definition 11 (Relaxed Weighted Output of sPNTs). *Let $H = (N_0, \mathcal{N}, \rho, F_c, W_c, T_s)$ be an hPNT over a concurrent semiring $\mathcal{S} = (S, \oplus, \otimes, \boxtimes, \overline{0}, \overline{1})$, u be an LPO over Σ and v be an LPO over Δ. The* relaxed weighted output $H_{O,r,\omega}(u,v)$ *is defined by*

$$H_{O,r,\omega}(u,v) = \bigoplus_{wlpo \in \mathrm{wLPO}_{O,r}(H,u,v)} \delta_{r,\omega}(wlpo).$$

We set $H_{O,r,\omega}(u,v) = (\emptyset, \emptyset, l, \nu)$ if $\mathrm{wLPO}_{O,r}(u,v) = \emptyset$.

Finally, we call hPNTs with shared transitions and the above extensions *segmenting Petri net transducers* or sPNTs for short.

A cascade of PNTs with an sPNT on top allows for a seamless propagation of information in both directions. An additional PNT on top representing semantic expectation leads to adjustments through the hierarchy down to the lowest level priming the whole recognition network. An incoming signal leads to adjustments up to the highest level resulting in weighted semantic structures directly derived from the input.

Since sPNTs still use terminal languages (cf. [6]) there is no added expressiveness compared to PNTs. Though a recursive refinement operation as introduced in [11] adds expressiveness it prohibits a closed representation of the interface thus making analysis much harder. When using clean transducers (see [16,11]) the closedness regarding composition operations still holds. It is our belief, that sPNTs have a bigger expressiveness concerning weighted output (after language composition) than PNTs since weights can now be multiplied without taking over dependencies.

5 Conclusion and Outlook

We introduced segmenting Petri net transducers for the translation of sequential input into non-sequential weighted output and the segmentation of the input on the basis of semantic structures. We showed how to compute segments on wLPOs and how segments are propagated during language composition.

State-of-the-art speech dialogue systems decide for an interpretation of an input signal based on a single overall weight assigned to the whole translation process. Segmenting allows using parts of the signal which are recognised well and initiating an enquiry call for the other parts. The notion of segments and their propagation also gives a nice solution to the problem of alignment from which tree transducers, DAG transducers, and other rewriting systems suffer.

The definition of equivalence for sPNTs and the combination with weighted input are subject to further research as well as analysis of algebraic properties of sPNTs and concrete technical restrictions for PNT-algorithms.

Most open questions arise from the fact that a single run is used to represent input and output. Another idea is to use a pair of loosely synced runs – one for input and one for output. This way propagation of dependencies could be completely circumvented in both directions. Namely the PNTs generating the semantic structure from our example force a prefix notation on the typed command and neither postfix nor infix notation would be accepted. However, such restrictions should be subject to the syntactic level – another PNT within the cascade.

Also notions from [22], namely the dependency of the flow weight function on the marking, could be used to eliminate the need for a recursive refinement operation. Regarding the mentioned real-time capabilities we plan to adapt on-the-fly composition algorithms for FSTs [4] to precomputed unfoldings of PNTs.

Future work will also deal with the computation of confidence scores, rather than weights, for the nodes of the perceived LPOs. To this end we want to create a hierarchy of semantic units leading to a more accurate computation of weights. These can be used to assess the reliability of semantic units of the input thus enhancing decisions made by the behaviour controller of cognitive systems.

Acknowledgements This work has been developed in the project Universal Cognitive User Interface (UCUI) which is partly funded by the German Federal Ministry of Education and Research (BMBF) within the research program IKT2020 (grant #16ES0297).

References

1. Chulan, U.A.U., Sulaiman, M.N., Mahmod, R., Selamat, H., Hamid, J.A.: Organizing the semantics of text with the concept relational tree. IJCSNS 8(9), 236 (2008)
2. Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. Monographs in Theoretical Computer Science, Springer (2009)
3. Duckhorn, F., Huber, M., Meyer, W., Jokisch, O., Tschöpe, C., Wolff, M.: Towards an autarkic embedded cognitive user interface. In: Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017. ISCA (2017)
4. Duckhorn, F., Wolff, M., Hoffmann, R.: A new epsilon filter for efficient composition of weighted finite-state transducers. In: Cosi, P., Mori, R.D., Fabbrizio, G.D., Pieraccini, R. (eds.) Interspeech 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011. ISCA (2011)
5. Geßler, P.: Kognitive Gerätesteuerung. Master's thesis, Brandenburgische Technische Universität Cottbus-Senftenberg, Deutschland (2017)
6. Hack, M.: Petri net languages. Tech. Rep. Memo 124, computation structures group, massachusetts institute of technology (1975)
7. Haykin, S.: Cognitive Dynamic Systems: Perception-action Cycle, Radar and Radio. Cambridge University Press, New York, NY, USA (2012)
8. Hoffmann, R., Eichner, M., Wolff, M.: Analysis of verbal and nonverbal acoustic signals with the Dresden UASR system. In: Verbal and Nonverbal Communication Behaviours. LNAI, vol. 4775, pp. 200–218. Springer (2007)
9. Huber, M., Kölbl, C., Lorenz, R., Römer, R., Wirsching, G.: Semantische Dialogmodellierung mit gewichteten Merkmal-Werte-Relationen. In: Hoffmann, R. (ed.) Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)". Studientexte zur Sprachkommunikation, vol. 53, pp. 25–32. TUDpress, Dresden (2009)
10. Huber, M., Kölbl, C., Lorenz, R., Wirsching, G.: Ein petrinetz-modell zur informationsübertragung per dialog. In: Lohmann, N., Wolf, K. (eds.) 15th German Workshop on Algorithms and Tools for Petri Nets, Algorithmen und Werkzeuge für Petrinetze, AWPN 2008, Rostock, Germany, September 26-27, 2008. Proceedings. CEUR Workshop Proceedings, vol. 380, pp. 15–24. CEUR-WS.org (2008), http://ceur-ws.org/Vol-380/paper03.pdf
11. Huber, M., Römer, R., Wolff, M.: Little Drop of Mulligatawny Soup, Miss Sophie? Automatic Speech Understanding provided by Petri Nets. In: Trouvain, J., Steiner, I., Möbius, B. (eds.) Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)". Studientexte zur Sprachkommunikation, vol. 86, pp. 122–129. TUDpress, Dresden (Mar 2017)
12. Jones, B., Andreas, J., Bauer, D., Hermann, K.M., Knight, K.: Semantics-based machine translation with hyperedge replacement grammars. In: COLING. pp. 1359–1376 (2012)
13. Lorenz, R.: Modeling Quantitative Aspects of Concurrent Systems Using Weighted Petri Net Transducers. In: Devillers, R.R., Valmari, A. (eds.) Application and Theory of Petri Nets and Concurrency – 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9115, pp. 49–76. Springer (2015), http://dx.doi.org/10.1007/978-3-319-19488-2_3
14. Lorenz, R., Huber, M.: Petri net transducers in semantic dialogue modelling. In: Wolff, M. (ed.) Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)". Studientexte zur Sprachkommunikation, vol. 64, pp. 286–297. TUDpress, Dresden (2012)
15. Lorenz, R., Huber, M.: Realizing the Translation of Utterances into Meanings by Petri Net Transducers. In: Wagner, P. (ed.) Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)". Studientexte zur Sprachkommunikation, vol. 65, pp. 103–110. TUDpress, Dresden (2013)

16. Lorenz, R., Huber, M., Wirsching, G.: On Weighted Petri Net Transducers. In: Ciardo, G., Kindler, E. (eds.) Application and Theory of Petri Nets and Concurrency – 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8489, pp. 233–252. Springer (2014), http://dx.doi.org/10.1007/978-3-319-07734-5_13
17. Pust, M., Hermjakob, U., Knight, K., Marcu, D., May, J.: Parsing english into abstract meaning representation using syntax-based machine translation. Training 10, 218–021 (2015)
18. Quernheim, D., Knight, K.: Towards probabilistic acceptors and transducers for feature structures. In: Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation. pp. 76–85. Association for Computational Linguistics (2012)
19. Ramachandran, D., Ratnaparkhi, A.: Belief tracking with stacked relational trees. In: 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue. p. 68 (2015)
20. Ramachandran, D., Yeh, P.Z., Jarrold, W., Douglas, B., Ratnaparkhi, A., Provine, R., Mendel, J., Emfield, A.: An end-to-end dialog system for tv program discovery. In: Spoken Language Technology Workshop (SLT), 2014 IEEE. pp. 602–607. IEEE (2014)
21. Römer, R., Wirsching, G.: Ein beitrag zu den natur- und geisteswissenschaftlichen grundlagen kognitiver systeme. In: Wagner, P. (ed.) Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)". Studientexte zur Sprachkommunikation, vol. 65, pp. 93–102. TUDpress, Dresden (2013)
22. Valk, R.: Self-modifying nets, a natural extension of petri nets. Automata, Languages and Programming pp. 464–476 (1978)
23. Wirsching, G., Huber, M., Kölbl, C., Lorenz, R., Römer, R.: Semantic dialogue modeling. In: Esposito, A., Esposito, A.M., Vinciarelli, A., Hoffmann, R., Müller, V.C. (eds.) COST 2102 Training School. Lecture Notes in Computer Science, vol. 7403, pp. 104–113. Springer (2011)
24. Wolff, M.: UASR: Unified Approach to Signal Synthesis and Recognition (2000–...). Online: https://www.b-tu.de/en/fg-kommunikationstechnik/research/projects/uasr, last visited: 31.03.2017
25. Wolff, M., Tschöpe, C., Römer, R., Wirsching, G.: Subsymbol-Symbol-Transduktoren. In: Wagner, P. (ed.) Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)". Studientexte zur Sprachkommunikation, vol. 65, pp. 197–204. TUDpress, Dresden (2013)