Petri Net Tranducers in Semantic Dialogue Modelling

Markus Huber and Robert Lorenz

Augsburg University, Germany
robert.lorenz@informatik.uni-augsburg.de

Abstract: In this paper we introduce Petri net transducers for the translation of non-sequential languages and present an application in the field of semantic dialogue modelling. Petri net transducers are a natural generilization of finite state transducers which in general provide more compact and intuitive models, are more flexible in use and have a higher expressive power.

1 Introduction

In this paper we introduce Petri net transducers (PNTs), a natural generalization of finite state transducers (FSTs), for the translation of so called non-sequential languages and present an application in the field of semantic dialogue modelling.

A non-sequential language contains words not consisting of a total order on their symbols but consisting of a partial order. Such words are called *partial words*. Figure 1 shows two partial words, an input and an output word of a Petri net transducer (PNT), in the form of directed acyclic graphs: The nodes represent the symbols of the partial word and the arrows a partial order on these symbols. For example, the input word consists of the symbols a, b, c, where a precedes c and all other occurrences of symbols are unordered. The output word consists of one occurrence of the symbol x and two occurrences of the symbol y.

Non-sequential languages can be used as compact representations of sequential languages since a partial word is uniquely determined by the set of its so called *linearizations*. A linearization of a partial order is a total order including the partial order. For example, the input word from Figure 1 has the linearizations *abc*, *bac* and *acb*.

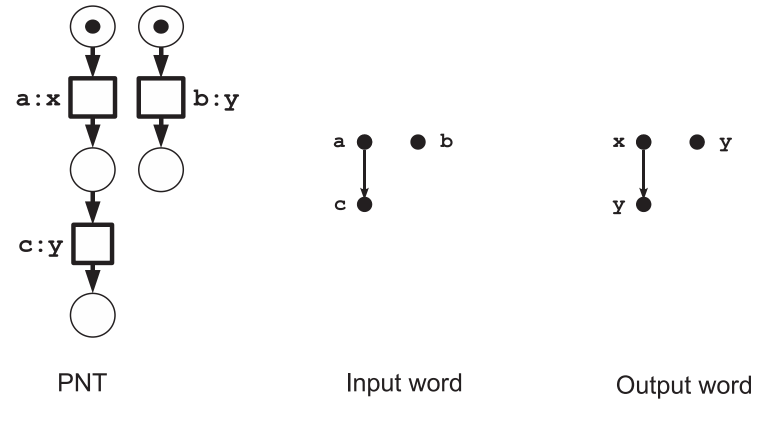

Figure 1 - Example of a PNT, translating a partial word into another partial word.

Moreover, non-sequential languages can provide additional information concerning concrete application areas. For example, symbols may represent action names of a concurrent system. In this case the partial order is used to model causal dependencies between action occurrences. Unordered action occurrences are interpreted to be causally independend or *concurrent*, which means that they can be observed in any order and also simultaneously. A PNT may be used to translate between non-sequential runs of the systems on different levels of abstraction.

In the second part of this paper we present an application where symbols represent

- on the one side words from utterances a speech dialogue system can understand (for example ”Call Peter Parker”),
- on the other side semantic categories the system can deal with (for example *person*, *firstname*, *lastname*).

Utterances are modelled as total orders and PNTs are used to assign so called *meanings* to utterances, where a meaning is a partial order representing the relation between semantic categories and sub-categories.

PNTs are a natural generalization of FSTs, since FSTs are used to translate sequential languages (consisting of words with a total order on their symbols) into sequential languages. There are several proposals to apply FSTs in the area of speech dialogue systems. One is the translation of speech signals into regocnition results up to the syntax level[2]. There are some publications using FSTs also on the semantic level [7] and there are some extensions of FSTs introducing restricted forms of parallelism in order to express additional information [4, 5]. An actual publication introduces the concept of so called (nested) multi-sequential languages as a concept ”between” sequential and non-sequential languages and examines their applicability in speech processing and processability by FSTs [10]. PNTs are able to translate general partial languages, thus they provide more compact and intuitive models than FSTs, are more flexible in use and have a higher expressive power. On the other side, there is a fully developed theory of FSTs including many useful operations on FSTs with efficient implementations in standard libraries [6, 12].

There are already several publications introducing PNTs and applying them in different application areas [9, 8, 1], however these are mainly case studies. Up to now there is no common basic formal definition and no theory on PNT-operations as for FSTs. Moreover, all existing definitions only make use of sequential semantics of PNTs.

The paper is organized as follows: In the first part (Section 2) we present a basic formal definition of PNTs for the translation of partial languages which is a proper generalization of FSTs. In the second part (Section 3) we discuss a simple example showing the applicability of PNTs in the area of semantic dialogue modelling.

2 Basic Formalism

In this section we develop a basic formalism of Petri net transducers (PNTs) as a natural generilization of finite state transducers (FSTs).

2.1 Mathematical Preliminaries

We start with necessary standard mathematical notions and definitions including labelled partial orders.

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

Given a function f from X to Y and a subset Z of X we write $f|_Z$ to denote the *restriction* of f to the set Z.

Given a finite set X, the symbol $|X|$ denotes the *cardinality* of X. The set of all subsets of X is denoted by $\mathscr{P}(X)$.

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

Let X, T be sets and $l : X \to T$ be a labelling function assigning to each $x \in X$ a label $l(x)$ from T. Such a labelling function can be lifted to subsets $Y \subseteq X$ in the following way: $l(Y)$ is the multiset over T given by $l(Y)(t) = |l^{-1}(t) \cap Y|$.

Given a binary relation $R \subseteq X \times Y$ and a binary relation $S \subseteq Y \times Z$ for sets X,Y,Z, then their composition is defined by $R \circ S = \{(x,z) \mid \exists y((x,y) \in R \wedge (y,z) \in S)\} \subseteq X \times Z$. 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 and the symbol R^* denotes the *reflexive transitive closure* $R^+ \cup \{(x,x) \mid x \in X\}$ of R. We also write aRb to denote $(a,b) \in R$.

Let A be a finite set of characters. A *(linear) word* over A is a finite sequence of characters from A. For a word w its lengths $|w|$ is defined as the number of its characters. The symbol ε denotes the *empty word* satisfying $|\varepsilon| = 0$. By A^* we denote the set of all words over A. A *(classical) language over A* is a (possibly infinite) subset of A^*.

A *(concurrent) step over A* is a multiset over A. A *step sequence* or *stepwise linear word* over A is a finite sequence of steps over A. A *step language over A* is a (possibly infinite) set of finite step sequences over 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* (all graphs considered in this paper are finite). The set of nodes of a directed graph G is also denoted by $V(G)$. The *preset* of a node $v \in V$ is the set ${}^\bullet v = \{u \mid u \to v\}$. The *postset* of a node $v \in V$ is the set $v^\bullet = \{u \mid v \to u\}$. The *preset* of a subset $W \subseteq V$ is the set ${}^\bullet W = \bigcup_{w \in W} {}^\bullet w$. The *postset* of a subset $W \subseteq V$ is the set $W^\bullet = \bigcup_{w \in W} w^\bullet$. 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. The set of maximal nodes of an acyclic directed graph $G = (V, \to)$ is the set $Max(G) = \{v \mid v^\bullet = \emptyset\}$, the set of its minimal nodes is the set $Min(G) = \{v \mid {}^\bullet v = \emptyset\}$. An acyclic directed graph $(V, \to')$ is an *extension* of an acyclic directed graph $(V, \to)$ if $\to \subseteq \to'$. An acyclic directed graph $(V', \to)$ is a *prefix* of an acyclic directed graph $(V, \to)$ if $V' \subseteq V$ and $(v' \in V') \wedge (v \to v') \Rightarrow (v \in V')$. An acyclic directed graph $(V', \to)$ is a *sub-graph* of an acyclic directed graph $(V, \to)$ if $V' = U \setminus W$ for prefixes $(U, \to)$ and $(W, \to)$. Then $(W, \to)$ is called *prefix of the sub-graph* $(V', \to)$.

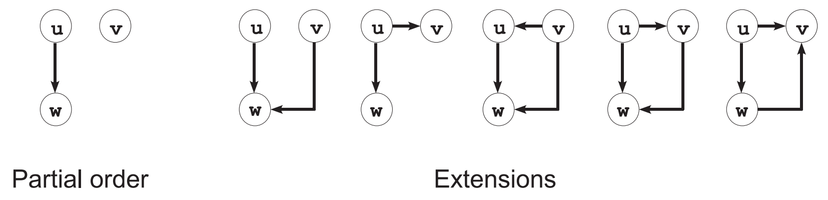

Figure 2 - Example of a partial order and all of its extensions.

A *partial order* over a set V is a binary relation $<\subseteq V \times V$ which is irreflexive ($\forall v \in V : v \not< v$) and transitive ($<=<^+$). We associate a finite partial order $<$ over V with the directed graph $(V,<)$.

Two nodes $v, v' \in V$ of a partial order $(V,<)$ are called *independent* if $v \not< v'$ and $v' \not< v$. By $co_< \subseteq V \times V$ we denote the set of all pairs of independent nodes of V. A *co-set* is a subset $C \subseteq V$ fulfilling $\forall x,y \in C : x\, co_<\, y$. A *cut* is a maximal co-set w.r.t. set inclusion. For a co-set C of a partial order $(V,<)$ and a node $v \in V \setminus C$ we write $v < C$, if $v < s$ for an element $s \in C$ and $v\, co_<\, C$, if $v\, co_<\, s$ for all elements $s \in C$. The sets $Max(po)$ and $Min(po)$ are cuts.

The *skeleton* of a finite partial order $po = (V,<)$ is the minimal relation $\prec\subseteq<$ satisfying $\prec^+=<$.

Graphically, nodes of partial orders are drawn as small circles and the order relation by (drawn-through) arrows between nodes. Figure 2 shows an example partial order together with all of its extensions. The nodes u and v as well as w and v are independent.

A *net* is a 3-tuple $N = (P,T,F)$, where P is a finite set of *places*, T is a finite set of *transitions* disjoint from P and $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*. A *marking* of a net assigns to each place $p \in P$ a number $m(p) \in \mathbb{N}_0$, i.e. a marking is a multiset over P. A *marked net* is a net $N = (P,T,F)$ together with an *initial marking* m_0. Graphically, places are drawn as circles, transitions as squares and the flow relation as arrows between places and transitions. A marking m is illustrated by drawing $m(p)$ tokens inside place p.

A *place/transition Petri net* (*PT-net*) is a 4-tuple $N = (P,T,F,W)$, where (P,T,F) is a net and $W : (P \times T) \cup (T \times P) \to \mathbb{N}_0$ is a *weight function* satisfying $W(x,y) > 0 \Leftrightarrow (x,y) \in F$.

Graphically, the number $W(x,y)$ is assigned to an arrow from x to y, if $W(x,y) > 1$ (that means, $W(x,y) = 1$ for arrows (x,y) without assigned weight). Figure 3 shows a marked PT-net having only arc weights 1.

We introduce the following multisets of places:

- ${}^\bullet t(p) = W(p,t)$ and $t^\bullet(p) = W(t,p)$ for transitions t.
- ${}^\bullet \tau(p) = \sum_{t\in T} \tau(t)\, {}^\bullet t(p)$ and $\tau^\bullet(p) = \sum_{t\in T} \tau(t) t^\bullet(p)$ for multisets of transitions τ.

The definition of executions of PT-nets depends on the *occurrence rule* of transitions, stating in which markings a transition (or a multiset of transitions) can occur and how these markings are changed by its occurrence. A transition $t \in T$ *can occur* in a marking m, if $m \geq {}^\bullet t$. A multiset of transitions τ *can occur* in m, if $m \geq {}^\bullet \tau$.

If a transition t occurs in a marking m, the resulting marking m' is defined by $m' = m - {}^\bullet t + t^\bullet$. If a multiset of transitions τ occurs in m, then the resulting marking m' is defined by $m' = m - {}^\bullet \tau + \tau^\bullet$. We write $m \xrightarrow{t} m'$ ($m \xrightarrow{\tau} m'$) to denote that t (τ) can occur in m and that its occurrence leads to m'.

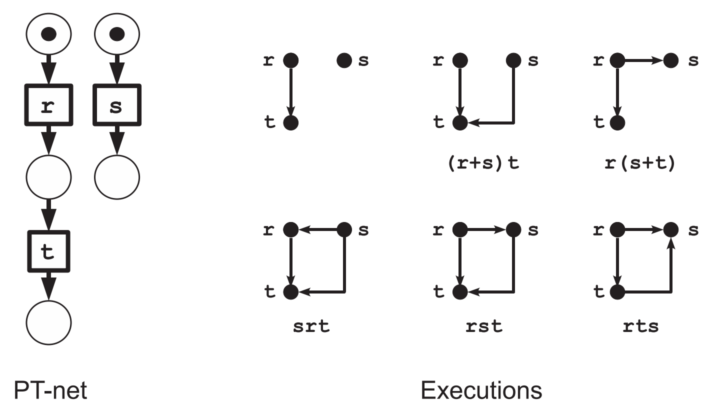

Figure 3 - A PT-net together with all of its executions represented by labelled partial orders (without prefixes).

The number $W(p,t)$ represents the number of tokens *consumed from p* by an occurrence of t and the number $W(t,p)$ represents the number of tokens *produced in p* by an occurrence of t.

The occurrence of a multiset of transitions τ in a marking m means, that all transitions in τ occur in parallel.

For example, in the intial marking of the PT-net shown in Figure 3 the transitions r,s and the multiset of transitions $(r+s)$ can occur.

The notion of *execution* depends on the chosen net semantics.

A *sequential execution in m* of a PT-net is a finite sequence of transitions $\sigma = t_1 \ldots t_n$ such that there are markings $m_1,\ldots,m_n$ satisfying $m \xrightarrow{t_1} m_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} m_n$. The PT-net shown in Figure 3 has the sequential executions *r*, *s*, *rs*, *sr*, *rt*, *rst*, *srt* and *rts*.

A *step execution in m* of a PT-net is a finite sequence of multisets of transitions $\sigma = \tau_1 \ldots \tau_n$ such that there are markings $m_1,\ldots,m_n$ satisfying $m \xrightarrow{\tau_1} m_1 \xrightarrow{\tau_2} \ldots \xrightarrow{\tau_n} m_n$. The PT-net shown in Figure 3 has as step executions all sequential executions and additionally $(r+s)$, $(r+s)t$ and $r(s+t)$.

We write $m \xrightarrow{\sigma} m_n$ to denote the occurrence of such executions σ.

Each sequential execution is also a step execution. The markings which can be reached from the initial marking via sequential executions (resp. step executions) are called *reachable*.

If τ is a multiset of transitions which can occur in a marking m and $\tau = t_1 + \ldots + t_n$ for transitions $t_1,\ldots,t_n$, then $t_1 \ldots t_n$ is a sequential execution in m, i.e. the transitions in τ can occur in m in arbitrary sequential order.

We use partial orders labelled by transition names to represent single non-sequential runs of PT-nets. The nodes of a partial order represent transition occurrences and its arrows an "earlier than"-relation between transition ocurrences in the sense that one transition occurrence can be observed earlier than another transition occurrence. If there are no arrows between two transition occurrences, then these transition occurrences are independend and are called *concurrent*. Concurrent transition occurrences can be observed in arbitrary sequential order and in parallel. This interpretation of arrows is called *occurrence interpretation*.

A *labelled partial order* (*LPO*) *over T* is a 3-tuple $(V,<,l)$, where $(V,<)$ is a partial order and

$l : V \to T$ is a labelling function on V. LPOs are also called *partial words*.

We only consider LPOs up to isomorphism, i.e. only the labelling of events is of interest, but not the event names. Formally, two LPOs $(V, <, l)$ and $(V', <', l')$ are *isomorphic*, if there is a renaming function $I : V \to V'$ satisfying $l(v) = l'(I(v))$ and $v < w \Leftrightarrow I(v) <' I(w)$.

In Figures, we do not show the names of the nodes of an LPO, but only their labels.

A *linear order* is an LPO $(V, <, l)$ where $<$ is a total order, i.e. there is no independence between transition occurrences: $\forall u, v \in V : u < v \vee v < u$. Linear orders represent sequential executions of Petri nets in the obvious way and can be identified with linear words. For example, Figure 3 shows LPOs representing the sequential executions rst, srt and rts.

The *set of linearizations* of an LPO is the set of linear LPOs which are extensions of this LPO. For example, the LPOs representing rst, srt and rts in Figure 3 are are linearizations of the LPO in the upper left corner. An LPO is uniquely determined by its set of linearizations.

A *stepwise linear LPO* is an LPO $(V, <, l)$ where the relation $co_<$ is transitive. The maximal sets of independent transition occurrences are called *steps*. The steps of a stepwise linear LPOs are linearly ordered. Thus, stepwise linear LPOs represent step executions of Petri nets and can be identified with stepwise linear words. For example, Figure 3 shows LPOs representing the step executions $(r+s)t$ and $r(t+s)$. The LPO in the upper left corner is not stepwise linear.

The *set of step-linearizations* of an LPO is the set of stepwise linear LPOs which are extensions of this LPO. For example, the LPOs representing $(r+s)t$ and $r(t+s)$ in Figure 3 are are step linearizations of the LPO in the upper left corner.

Let $N = (P, T, F, W, m_0)$ be a marked PT-net. An LPO $lpo = (V, <, l)$ is a *LPO-run* of N if each step-linearization of lpo is a step execution of N. Figure 3 shows a marked PT-net togther with all of its LPO-runs (without prefixes).

An LPO-run lpo of N is said to be *minimal*, if there exists no other LPO-run lpo' of N such that lpo is an extension of lpo'.

From the definition follows that extensions of LPO-runs also are LPO-runs. This means, the set of all LPO-runs can be deduced from the set of minimal LPO-runs.

In figures we often omit transitive arrows of LPOs for a clearer presentation.

2.2 PNT Syntax

A PNT is a Petri net which, for every transition occurrence, may read a symbol i from an input alphabet Σ and may print a symbol o from an output alphabet Ω. Graphically, these symbols are annotated to transitions in the form $i : o$. If no input symbol should be read or no output symbol should be printed, we use the empty word symbol ε as annotation. We use the basic Petri net class of place/transition nets to define PNTs.

Definition 1 (PNT) *A* PNT *is a tuple* $N = (P, T, F, W, m_0, \Sigma, \sigma, \Omega, \omega)$*, where*

- (P, T, F, W, m_0) *is a marked PT-net called the* underlying PT-net*,*
- Σ *is a set of* input symbols *and* $\sigma : T \to \Sigma \cup \{\varepsilon\}$ *is the* input mapping*,*
- Ω *is a set of* output symbols *and* $\omega : T \to \Omega \cup \{\varepsilon\}$ *is the* output mapping.

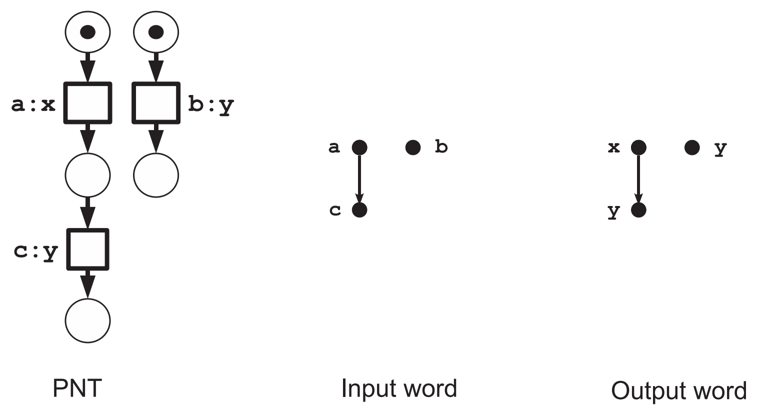

Figure 4 - A PNT translating an input word into an output word.

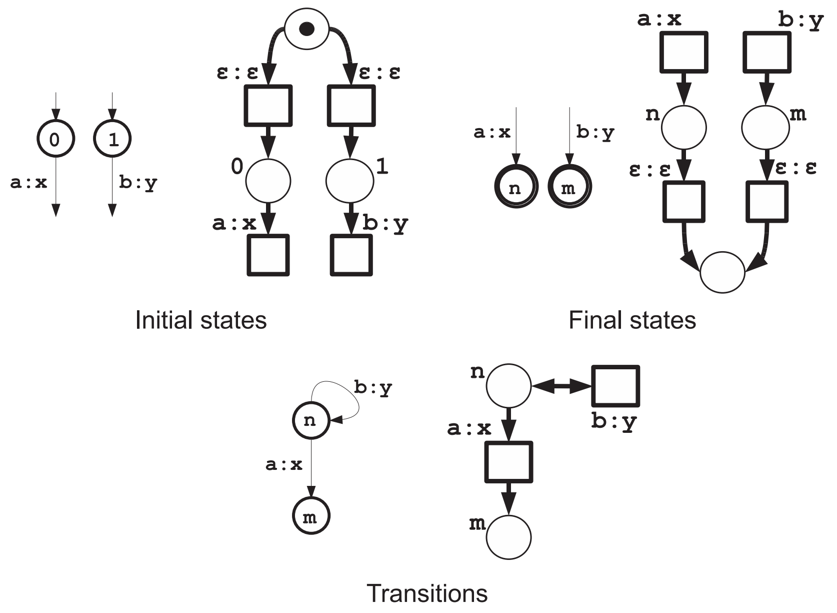

Figure 5 - Translating an FST into an equivalent PNT.

Figure 4 shows a PNT. In the next subsection we formally define PNT semantics determining the translation of partial language. Briefly, input words are translated into output words, where input and output words are derived from LPO-runs by renaming nodes with input and output symbols according to the input and output mappings.

PNTs are a proper generilization of FSTs, i.e. each FST can be written as a PNT. Figure 5 illustrates the translation of FSTs into equivalent PNTs, where we call FSTs and PNTs *equivalent* if they define the same relation on languages. The main idea is to define a PNT whose reachable markings are all of the form, that exactly one place is marked by one token. Since each reachable marking represents a global state of the PNT, then each place represents a global state of the FST. Technically, this can be derived by using only unbranching transitions, i.e. $|{}^{\bullet}t| \leq 1$ and $|t^{\bullet}| \leq 1$ for all transitions t.

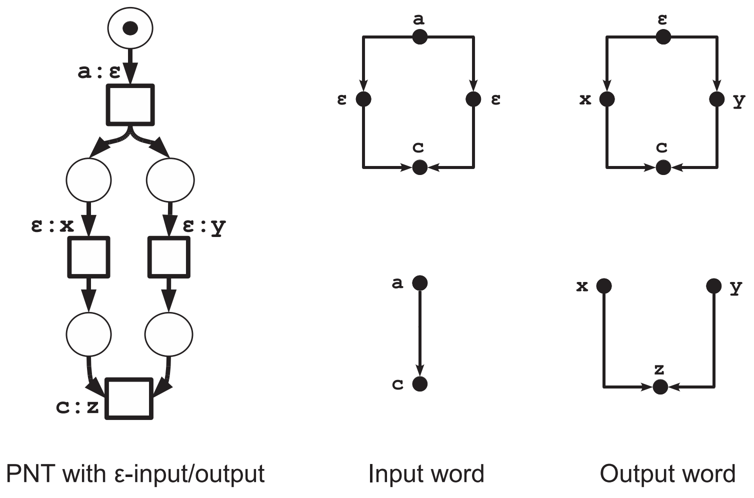

Figure 6 - Construction of input and output words.

2.3 PNT Semantics

Considering non-sequential semantics of Petri nets, a PNT can be used to translate a partial language into another partial language, where so called input words are related to so called output words.

Input and output words are defined as LPOs $(V, <, l)$ with a labelling function $l : V \to A \cup \{\varepsilon\}$ for some input or output alphabet A. Such LPOs we call *ε-LPOs*.

For each ε-LPO there exists an LPO with the same set of linearizations. This LPO can be constructed by successively collapsing ε-labelled nodes in the following way:

- If $l(v) = \varepsilon$ and ${}^\bullet v = \emptyset \vee v^\bullet = \emptyset$, then just delete v together with its adjacent edges.
- If $l(v) = \varepsilon$ and ${}^\bullet v \neq \emptyset \wedge v^\bullet \neq \emptyset$, then delete v together with its adjacent edges and add the edges ${}^\bullet v \times v^\bullet$.

The derived LPO we call *equivalent* to the ε-LPO.

Definition 2 (Input and Output Words) *Let $N = (P, T, F, W, m_0, \Sigma, \sigma, \Omega, \omega)$ be a PNT and let $lpo = (V, <, l)$ be an LPO-run of the underlying PT-net (P, T, F, W, m_0).*

The input word *$\sigma(lpo)$ corresponding to lpo is the LPO equivalent to the ε-LPO $(V, <, \sigma \circ l)$.*
The output word *$\omega(lpo)$ corresponding to lpo is the LPO equivalent to the ε-LPO $(V, <, \omega \circ l)$.*

Figure 6 shows an example for the translation of partial words in the presence of ε-inputs and -outputs.

2.4 PNTs and FSTs

The next research steps are:

- Definition and implementation of operations on PNTs as for FSTs, as for example rational operations (union, concatenation, closure), basic unary operation (reversal, inversion, projection), fundamental binary operations (composition, union, difference) and optimization operations (ε-removal, minimization).
- Extension of PNTs by weights from a semiring.

Then PNTs and FSTs can be combined via such operations, since each FST is a special PNT. In particular it will be possible to compose FSTS and PNTs and build hierarchical systems consisting of FSTs on some levels and of PNTs of other levels in such a way. In the next section we briefly describe such a system in the area of speech dialogue systems. Weights may be used in such systems to express uncertainty of recognition results and predictions of utterances of the user.

3 Semantic Dialogue Modelling

In this section we briefly present an application of PNTs within a new approach to develop the cognitive user interface of a hierarchical cognitive dynamic speech signal processing system.[1]

The system includes a semantic level used to interpret syntactic regocnition results of speech signals. These interpretations will be used to control a natural language dialogue, where user queries can be freely formulated and a dialogue with the user is initiated in which step by step missing information is collected in order to identify the action indended by the user together with the data necessary to perform the action.

In our approach, the system successively integrates recognition results of user queries into a state of information and generates a request concerning missing information together with an expectation for the next user query. This expectation is used as a semantic-driven configuration of the speech recognizer in the next dialog step.

In the following we use PNTs to translate recognition results of speech signals on the syntax level into semantic interpretations.

3.1 UMP Transducer

Figure 7 shows an application of PNTs in modelling semantics within a dialogue system. It translates recognition results of speech signals on the syntax level into semantic interpretations.

For this little example we consider a system which knows about two persons: Peter Parker with ID 1 and Parker Lewis with ID 2. To keep it very simple we consider also that only the following four utterances should be understandable:

- Peter
- Peter Parker
- Parker
- Parker Lewis

[1]This systems is developed in cooperation with institutes from TU Dresden (R. Hofmann) and BTU Cottbus (M. Wolff).

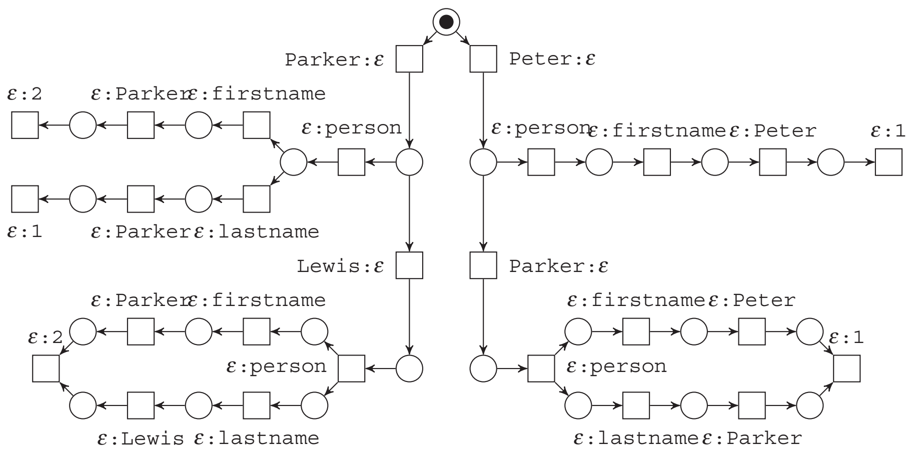

Figure 7 - A Petri net transducer relating utterances to different meanings.

In this context *understandable* means translatable into semantic categories the system can deal with. The categories for our example are *person*, *firstname* and *lastname*, all particular parts of the actual names and the relevant IDs. All these are elements of the output alphabet. The input alphabet is formed by all single words from the utterances. We call an input word *utterance* and an output word *meaning*.

The shown PNT relates for example the utterance ”Peter” non-ambiguously to the meaning *person.firstname.Peter.2*. The utterance ”Parker” in contrast has two different meanings because it is unclear whether ”Parker” is the firstname of person 1 or the lastname of person 2. In Figure 8 the corresponding partial orders (which are total orders in these cases) can be seen.

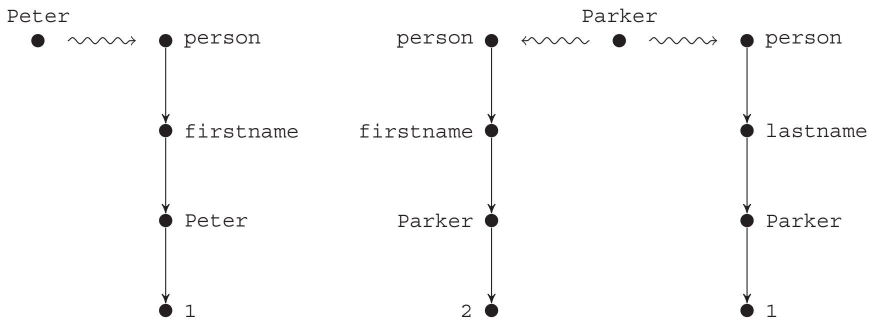

Figure 8 - Meanings for the utterances ”Peter” and ”Parker”.

Now let us take a look at the utterance ”Peter Parker”. Its meaning contains two parts: ”Peter” is the firstname of person 1 and ”Parker” is the lastname of person 1. This time there is no ambiguity in the ”Parker”-part because for the utterance as a whole the other interpretation is not feasible. In Figure 9 the partial orders for this situation are shown. The meaning represents the situation that a person can be identified by a combination of a firstname and a lastname, which are modelled as unordered categories.

As of now it is clear that the PNT from Figure 7 translates utterances our system should understand into meanings which are reasonable within the context of the system. So we do not try to produce every possible interpretation for every possible utterance but provide the system with a set of what we call *Utterance-Meaning-Pairs* (UMPs) [11] determining which utterances can be unterstood and which meanings of utterances are available (this set is determined by the

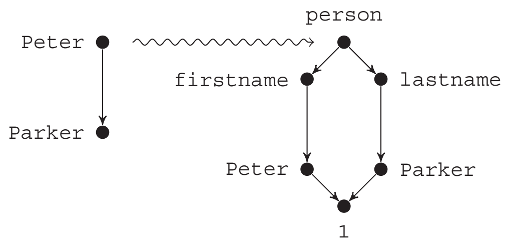

Figure 9 - Meaning for the utterance "Peter Parker".

application area of the system). PNTs relating utterances and meaning we call *UMP tranducers*.

3.2 Representation of Semantic Information

In the above example we used partial orders to represent meanings of utterances. This is not an ad-hoc notation but a simplified form of a general and universal formalism to model different kinds of information within a speech dialogue systems on the semantic level, as for example recognition results (as presented in the above example), the world model of the application (which we described only be words in the above example), the information state, the user model and semantic configuration of the speech recognizer in each dialogue step.

This formalism is called feature-value-relation (FVR) [3] and allows to structure semantic categories of information and to relate data to semantic categories. In the above example, we considered the semantic category *person* consisting of the sub-categories *firstname* and *lastname* and related concrete firstnames and lastnames of different persons from a database to these categories.

Moreover, in FVRs it is possible to assign weights to semantic categories or data pieces. Dependent on the kind of information, these weights have a different interpretation, for example:

- Consider an utterance with two alternative meanings and a situation within a dialogue where one of the meanings is more likely than the other. Then weights can express such predictions. In every dialogue step the system generates an expectation concerning the next utterance of the user in the form of an FVR. These weights get promoted downwards through the hierarchy and help the speech recogniser to achieve more suitable results.
- The regognition result produced by the speech recognizer in general contains some degree of uncertainty. This uncertainty also can be expressed by weights within an FVR.

Altogether, in the above example, a PNT is used to relate utterances to meanings given by FVRs without weights. The extension of PNTs by weights is a topic of future research.

References

[1] Biljon, W. R. van: *Extending Petri nets for specifying man-machine dialogues*. Int. J. Man-Mach. Stud., 28(4):437 – 455, 1988.

[2] Hoffmann, R., M. Eichner . M. Wolff: *Analysis of verbal and nonverbal acoustic signals with the Dresden UASR system. . Verbal and Nonverbal Communication Behaviours*, . 4775 . *LNAI*, . 200– 218. Springer, 2007.

[3] HUBER, M., C. KÖLBL, R. LORENZ, R. RÖMER . G. WIRSCHING: *Semantische Dialogmodellierung mit gewichteten Merkmal-Werte-Relationen*. . *Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)"*, . 53 . *Studientexte zur Sprachkommunikation*, . 25–32, 2009.

[4] KUSKE, D. . I. MEINECKE: *Branching Automata with Costs - A Way of Reflecting Parallelism in Costs*. Theoretical Computer Science, 328:53 – 75, 2004.

[5] LODAYA, K. . P. WEIL: *Series-parallel Languages and the bounded-width Property*. Theoretical Computer Science, 237:347 – 380, 2000.

[6] MOHRI, M.: *Weighted Automata Algorithms*. Springer, 2009.

[7] RAYMOND, C., F. BÉCHET, R. D. MORI . G. DAMNATI: *On the use of finite state transducers for semantic interpretation*. Speech communication, 48(3-4):288 – 304, 2006.

[8] WANG, F. Y., M. MITTMANN . G. N. SARIDIS: *Coordination specification for CIRSSE robotic platform system using Petri net transducers*. Journal of Intelligent and Robotic Systems, 9:209 – 233, 1994.

[9] WANG, F. Y. . G. N. SARIDIS: *A model for coordination of intelligent machines using Petri nets*. . *Proceedings of the IEEE International Symposium on Intelligent Control*, . 28–33. IEEE Comput. Soc. Press, 1989.

[10] WIRSCHING, G.: *Nichtsequentialität in der Sprachverarbeitung mit FST*. . *Proceedings of "Elektronische Sprachsignalverarbeitung (ESSV)"*, Studientexte zur Sprachkommunikation, 2012.

[11] WIRSCHING, G. . C. KÖLBL: *Language Modeling with Utterance-Meaning-Pairs*. . 2011-12, Institute of Computer Science, University of Augsburg, 2011.

[12] WOLFF, M.: *Akustische Mustererkennung*. Habilitation, 2009.