Boolean and free-choice semantics of Event-driven Process Chains

Joachim Wehler
Ludwig-Maximilians-Universität München, Germany
joachim.wehler@gmx.net

Abstract. In parallel to the successful application of Event-driven Process Chains (EPCs) in commercial IT-projects there is an ongoing scientific debate about their semantics. It is even controversial, if one should provide EPCs with a local or with a non-local semantics. In this paper, we unify two local EPC semantics, which build on Petri nets. One of them is the free-choice semantics due to the translation of EPCs into free-choice systems. The other one is the Boolean semantics, which derives from translating the EPC into a certain class of high-level Petri nets. The free-choice semantics is restricted to EPCs with only AND or XOR-connectors, but allows arbitrary loops. The Boolean semantics allows connectors of arbitrary logical type. But at this stage, it covers only EPCs with well-structured loops. For EPCs, where both semantics are defined, we derive the free-choice semantics from the Boolean semantics and show as main result: The EPC is well-formed with respect to the Boolean semantics if and only if it is well-formed with respect to the free-choice semantics. After reviewing a series of examples from the literature we continue the investigation of the Boolean semantics for EPCs with circuits different from well-structured loops.

Keywords: EPC, local semantics, Boolean system, free-choice system, loop

1 Introduction

In the domain of business process management most commercial projects in Germany use the language of Event-driven Process Chains (EPCs) to model their processes. The widespread use of this language can be explained by two factors: In a successful manner EPCs combine intuitive comprehensibility with sufficient expressiveness. A second reason is a marketing aspect: EPCs are implemented into the tool ARIS[1], the market leader in the domain of business process modeling and controlling.
The language of EPCs has been invented by Keller, Nüttgens and Scheer in 1992 [KNS1992]. EPCs represent the control flow of a process as the interplay of three components: Events, functions and logical rules. The rules use connectors of logical type AND, XOR and OR. More specific, concurrency is represented by AND-splits and AND-joins. Strong or exclusive alternatives are modeled by XOR-splits and XOR-joins, while OR-splits and OR-joins model weak alternatives.

[1] ARIS is a product of IDS Scheer AG

Nearly from the beginning EPCs have attracted the interest of the academic domain. From 1994 until today there is a ongoing debate about the semantics of EPCs, about the concept of their well-formedness and about formal methods to verify EPCs.
The problem of the semantics for the logical EPC connectors shows up in full sharpness already for binary connectors. These are connectors with either one entry and two exits (*split*) or two entries and one exit (*join*). In the following we restrict to binary join- and split-connectors. About the semantics of AND-connectors there prevails agreement. Highly debated remains the semantics of XOR-connectors and OR-connectors. Most difficult seems the semantics of the OR-join.
The different approaches concerning the semantics of the connectors fall into two classes: *Local* semantics and *non-local* semantics. A local semantics considers only the given connector, a non-local semantics takes into consideration also a subset of all process runs. Typical non-local semantics are the semantics of Kindler [Kin2006], a typical local semantics is the free-choice semantics of van der Aalst [Aal1999].
The focus of this paper is to unify two different types of local EPC semantics. The paper is organized as follows: Chapter 2 *Examples of EPCs* presents a series of examples from the literature and provides them with a well-defined initial state. Chapter 3 *Non-local versus local semantics* shortly reviews some approaches to the problem of EPC semantics. Chapter 4 *Boolean semantics* is the main part of the paper. After unifying the free-choice and the Boolean semantics we prove: Well-formedness with respect to one semantics is equivalent to well-formedness with respect to the other semantics. The paper ends with Chapter 5 *Conclusion*, which also gives an outlook to further research.
The paper presupposes some familiarity with EPCs and Boolean systems. Such knowledge can be obtained by reading, e.g., [KNS1992], [Rum1999], [LSW1998], [Kin2006], ordered by increasing difficulty.

2 Examples of EPCs

In the present chapter we collect some EPCs from the literature. We will refer to them in later chapters to illustrate our reasoning about the semantics of EPCs.
The first question concerning the semantics of EPCs asks about the type of objects, which flow along the arcs of the EPC. In [KNS1992] the authors state, that an EPC represents the *flow of control*. Even when EPCs allow for augmenting each function by the processed data, these data do not participate in the flow.
All EPCs are assumed to be connected. EPCs from commercial projects are event-bounded, i.e. they start with a set of one or more events (*start* events) and end with one or more events (*end* events). The intended semantics expressed by this modeling rule is the following: The start events trigger the process, the end events describe the final state, which the process has to achieve. This simple rule assumes in particular, that a process has a final state at all, i.e. that it terminates.
Due to the lack of markings any EPC needs an additional specification: One has to determine, which combination of start events may happen and which combination of end events is intended. As long as the initial state of EPCs is undefined, one cannot expect a well-defined semantics for EPCs. Therefore several authors have borrowed the marking concept from formal process languages like Petri nets. Annotating the

EPC with a marking implies, that one considers no longer a plain EPC. Instead one asks for the semantics of a marked EPC.
In the present paper like in our previous paper [LSW1998] we use a different method, striving to stay within the range of the original EPC language. To provide an EPC with well-defined initial and final states we circuit each event-bounded EPC, i.e. we augment it with a *start/end connection*. A start/end connection introduces a separate event "start/end" and connects the events at the boundary to the new event with the help of the logical connectors from the EPC language. The initial state of an EPC occurs, when this distinguished event happens. The final state occurs, when it is achieved a second time. EPCs with start/end connection are strongly connected.
In the following we seek to present all examples as EPCs with start/end connection. Because we extracted some EPCs from the literature, we had to find out, which start/end connection expresses best the intention of the author. Sometimes it could be derived from an annotation of the EPC, for instance relating to the initial marking. But some other time it is unclear, if the EPC models a terminating process at all.
To simplify matters we often have omitted all events and functions, which are irrelevant to discuss the semantics of the logical connectors. Hence the EPCs are not necessarily shown with their correct syntax.

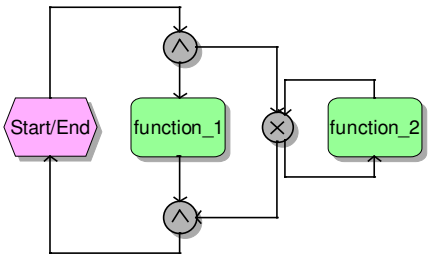

Fig. 1 Single loop

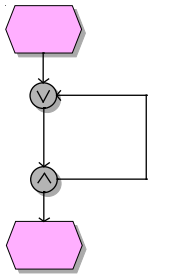

Fig. 2 Circuit [MA2006]

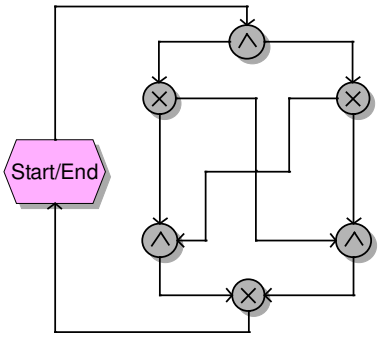

Fig. 3 Entangled AND and XOR [GT1984]

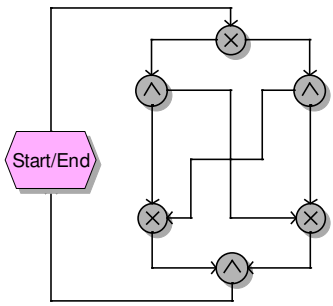

Fig. 4 Entangled XOR and AND [GT1984]

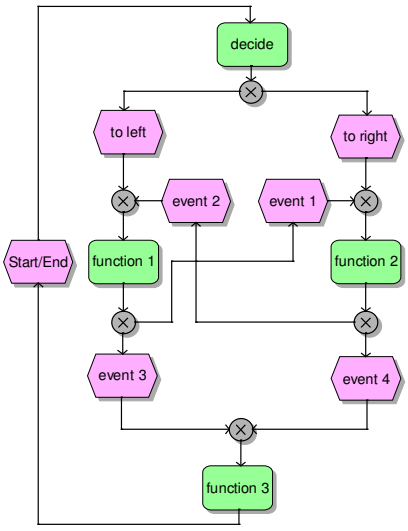

Fig. 5 Loop with multiple entries and exits

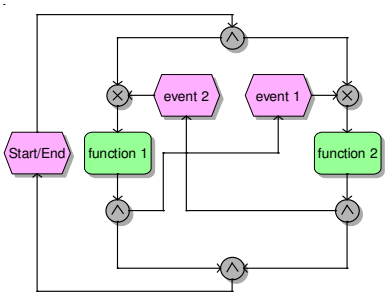

Fig. 6 Entangled circuit [Kin2006]

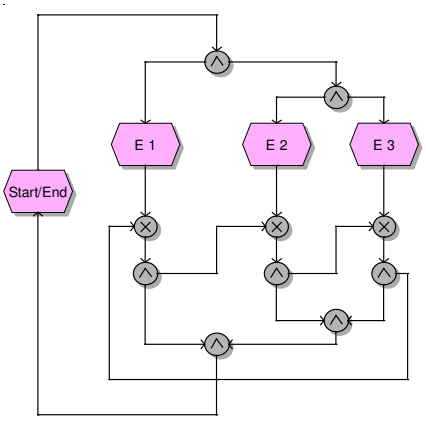

Fig. 7 Entangled circuit [Kin2006]

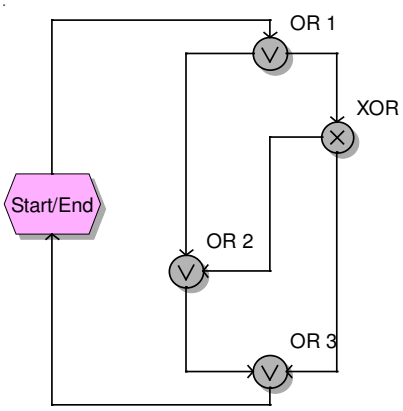

Fig. 8 Unpaired Or-connectors

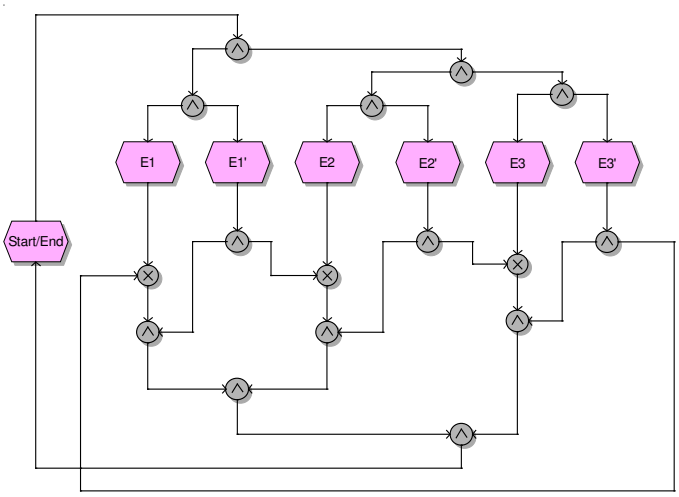

Fig. 9 Entangled AND/XOR [Kin2006]

3 Non-local versus local semantics

All authors reasoning about the semantics of EPCs have provided EPCs at least with the concept of state. Mostly, they even have translated EPCs into an other language with a well-defined semantics. Such languages are *transition systems* or *Petri nets*.

The most recent survey of proposals for EPC semantics is contained in [MA2006]. The authors also assess each proposal and state its limitations. Concerning non-local semantics the authors consider in particular the papers [Rit2000], [WEAH2005] and [Kin2006]. Concerning local semantics they refer to [CS1994][2], [LSW1998], [Aal1999] and [Rit2000]. In addition we would like to refer to [Rod1997] and [GL2005]. The work of Rodenhagen is an early proposal to translate EPCs into Petri nets. Gruhn-Laue translate EPCs into a modification of coloured Petri nets and use low- and high-tokens.

In our opinion the question of *non-local* semantics for EPCs has been settled by the result of Kindler. His work has been prompted by the attempt of Nüttgens-Rump [NR2002], to formalize their intended non-local semantics by a transition system. They introduced markings for the state of an EPC and considered transition systems as a formalization of statefull EPCs. The resulting transition system depends heavily on the semantics of the XOR-joins and OR-joins. In case one of the join-entries is marked, one has the choice between an optimistic non-local semantics and a pessimistic non-local semantics. In case of doubt the optimistic join fires, while the

[2] We consider the translation into coloured Petri nets a *local* EPC-semantics, because the constituents of Petri nets like places, arcs and transitions have a local semantics.

pessimistic one refrains from firing. The EPC has an *ideal semantics*, iff both non-local semantics coincide. Concerning the intended non-local semantics Kindler proves in [Kin2006]: An EPC has

1. either a unique ideal semantics
2. or two or more ideal semantics
3. or a pair of non-local semantics, which are not ideal but approximate best an intended ideal semantics.

An example from case 2 is the EPC in Fig. 6. Examples from case 3 are the cyclic EPCs in Fig. 7 and the acyclic EPC in Fig. 9. Apparently, in case 2 as well as in case 3 the EPC does not have a unique ideal semantics.

4 Boolean semantics

Now - the question of non-local semantics being settled – we focus on local EPC semantics. We do not aim at a new formal model for a semantics, which might have been intended by the creators of EPCs. Instead we unify two existing local semantics.
The Boolean semantics of an EPC is a local one: It is the semantics of a high-level Petri net, into which the EPC translates. A *state* of the EPC is a reachable marking of its Petri net.

Boolean system

A *Boolean system* is a coloured Petri net $BS = (BN, \mu)$ with a *Boolean net* BN and an *initial marking* μ. Each place of BN has the same token colour set

$$Boole = \{high, low\}.$$

A *high-token,* which marks a place, shows the presence of the control flow, while a *low-token* explicitly excludes the control flow. Every transition of a Boolean net has a binding colour set: An AND-transition has two binding elements, an XOR- transition has three and an OR- transition has four binding elements. But transitions of a Boolean net are not restricted to these logical types. More general formulas are allowed, specifically asymmetric OR-connectors like L_XOR are well-defined.

Logical type	Binding elements
AND	(1,1):=(high, high; high), (0,0):=(low, low; low)
XOR	(1,0):=(high, low; high), (0,1):=(low, high; high), (0,0):=(low, low; low)
OR	(1,0):=(high, low; high), (0,1):=(low, high; high) (1,1):=(high, high; high), (0,0):=(low, low; low)
L_XOR	(1,0):=(high, low; high), (1,1):=(high, high; high), (0,0):=(low, low; low)

Fig. 10 Binding elements of join-transitions of different logical type

Fig. 10 enumerates the binding elements or firing modes of join-transitions of different logical type. A notation like $(1,0) := (high, low, high)$ means that the binding

element *(1,0)* is enabled by a high-token at its left pre-place and a low-token at its right pre-place. When firing it creates a high-token at its post-place. The asymmetric L_XOR-join synchronizes the activation of both entries like an AND-join and it allows the exclusive activation of its left entry like one of the XOR-modes.
If one forgets about all colours of BS, i.e. about the difference between token elements and about the difference between all binding elements of the same transition, then one obtains the *skeleton* of the Boolean system: It is an ordinary Petri net

$$BS^{skel} = \left(BN^{skel}, \mu^{skel} \right)$$

with the same net structure, together with a canonical map

$$BS \longrightarrow BS^{skel}.$$

For the formal definition of Boolean nets and Boolean systems[3] we refer to ([LSW1998], Def. 2 and 7). All Boolean nets of the present paper are supposed to be *faithful concerning activation*: Any *high-mode*, i.e. any firing mode, which consumes at least one high-token, also creates at least one high-token.

As is well known, coloured Petri nets are no more than a compact notation for ordinary Petri nets. Every coloured Petri net expands into an ordinary Petri net, named its *flattening*: Every token element of a place of the coloured net induces a place of the flattening. Every binding element of the coloured net induces a transition of the flattening. And the token colour of any token of the coloured Petri net induces a token at the corresponding place of the flattening.
In case of a Boolean system $BS = \left(BN, \mu \right)$ the flattening of the different join-transitions from Fig. 10 are the ordinary nets from Fig. 11. An analogous flattening is obtained for the split-transitions just by reversing arcs. In Fig. 11 the gray places and transitions as well as their connecting arcs form part of an ordinary Petri net, which is called the *high-system*

$$BS^{high} = \left(BN^{high}, \mu^{high} \right)$$

of the given Boolean system. The white components form part of a second ordinary Petri net, which is called the *low-system* of the Boolean system. It is the subnet of the flattening, which is generated by all low-places and all low-transitions. Factoring out the low-system from the flattening gives the high-system as a quotient of the flattening.

[3] In the present paper it is not required as part of the definition, that the skeleton of a Boolean system is live and safe. But all examples will have this property

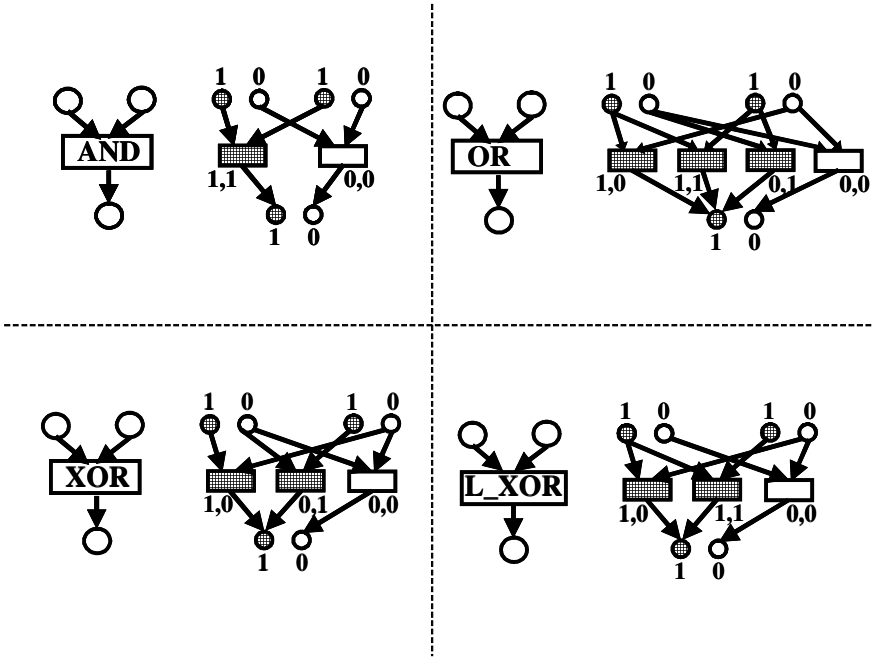

Fig. 11 Flattening of join-transitions of different logical type (1 = high, 0 = low)

If all transitions of a Boolean system are of logical type AND or XOR and if all places are unbranched, then the Boolean System is called a *bipolar system* ([LSW1998], Remark 7). Specifically, a bipolar system has no transitions of logical type OR. Bipolar systems have been invented by Genrich-Thiagarajan [GT1984].
The high-net of a bipolar system is a free-choice net of *restricted* type: Every place has either a single post-transition, or all post-transitions of the place have no other pre-place. For the definition of general free-choice systems and for their theory cf. [DE1995].

In the context of bipolar systems it is important to know, if the free-choice system excludes any *frozen tokens*. Absence of frozen tokens is a fairness property: It is not possible to fire an infinite occurrence sequence without moving all tokens.

1. Definition *(Frozen token)*
An ordinary Petri net (N,μ_0) has no *frozen tokens*, iff for every reachable marking μ holds: For every strictly less marking $\nu < \mu$ the Petri net (N,ν) has no infinite occurrence sequence.

2. Remark. For a live free-choice system the absence of frozen tokens is even a structural property: A live free-choice system has no frozen tokens iff it is safe and every T-component has a nonempty intersection with every P-component ([BD1990], Theor. 6.2).

In order to unify the Boolean and the free-choice semantics of an AND/XOR-EPC we use a result, which relates the behaviour of a bipolar system to the behaviour of its high-system and its skeleton. Here we characterize the behaviour of a Petri net by two properties named respectively safeness and liveliness.
A Petri net is *safe*, if any reachable marking marks no place with more than a single token. Safeness is an intuitive requirement for a Boolean system, which serves to model the control-flow of a system. Note, that safeness does not exclude the investigation of multiple process instantiation as long as different process instances do not interact.
We call a Boolean system *high-live*, if for any reachable marking, for any transition and for any of its high-modes there exists a follower marking, which activates the given high-mode. A high-live Boolean system does not contain any starving high-modes, which after a finite number of executions could not be enabled any longer. Liveliness is a plausible requirement to intensify the structural constraint of strong connectedness. It takes into account also the behaviour. Note that it is not reasonable to require that the Boolean system is live with respect to the low-modes, too.

3. Theorem *(Bipolar system and free-choice system)*
i) A bipolar system is high-live and safe if and only if its high-system is live and safe without frozen tokens and its skeleton is live and safe.
ii) Every restricted free-choice system, which is live and safe without frozen tokens, is the high-system of a bipolar system, which is high-live and safe.

Proof. Cf. [Weh2007], Theor. 4.6 and 4.9.

A free-choice net is named *well-formed*, if it has a live and bounded marking. Due to a result, which generalizes a theorem of Genrich, the existence of a live and bounded marking even implies the existence of a live and safe marking ([DE1995], Theor. 5.10). Hence a free-choice net is well-formed, iff it has a live and safe marking. For free-choice nets of restricted type, in particular for the high-nets of AND/XOR-EPCs, there exists a useful characterization of well-formedness due to Esparza-Silva [ES1990].

4. Theorem (*Well-formedness of restricted free-choice nets*)
A restricted free-choice net is well-formed, iff it is strongly connected, no elementary circuit has a TP-handle, and every PT-handle of an elementary circuit has a TP-bridge.

An *elementary path* in a net is a path without self-intersection. A *handle* at an elementary circuit is an elementary path, which starts and ends with a node of the circuit, but has no other node in common with the circuit. A handle, which starts with a transition and ends with a place, is called *TP-handle*. If it starts with a place and ends with a transition, it is called *PT-handle*. A *TP-bridge* of a handle at an elementary circuit is a path, which starts with a node of the handle and ends with a node of the circuit, but has no other nodes in common with the handle or the circuit. For formal definitions of these concepts cf. [ES1990].

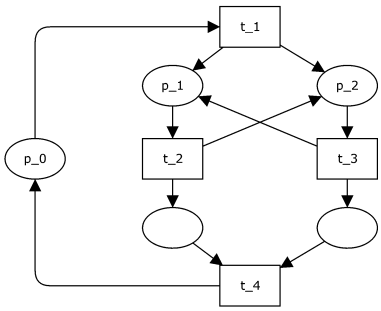

Fig. 12 Restricted free-choice net with TP-handles

The restricted free-choice net from Fig. 12 is not well-formed, because it contains several forbidden TP-handles: The elementary path (t_1, p_2, t_3, p_1) is a TP-handle at the left elementary circuit with six nodes, which passes through place p_0 and the three transitions t_1, t_2 and t_4.

Translation of EPCs without circuits

In this section we consider EPCs, which are a priori acyclic, i.e. they are circuit-free before performing their start/end connection from Chapter 2. Afterwards the circuited EPC translates into a Boolean system ([LSW1998], Proced. 1), which is an *elementary Boolean loop system*

$$EBLS = (BN, p, \mu):$$

The skeleton of the Boolean net BN is an *elementary loop* ([LSW1998], Def. 9). The place $p \in BN$ is the *baseplace*, i.e. the distinguished place corresponding to the event "start/end" of the EPC. The *initial marking* μ of BN has a single token, namely a high-token, which marks the baseplace. If $EBLS$ is high-live and safe, then the Boolean net BN and the original EPC is named *well-formed*.

The skeleton BN^{skel} as well as the high-net BN^{high} are strongly connected and the complements of their baseplaces p^{skel} and p^{high} do not contain any circuit. Note that all circuits from BN^{skel} and BN^{high} are elementary circuits, which pass through the baseplace. When cutting BN^{skel} or BN^{high} by splitting the baseplace into a source place and a sink place one obtains a Workflow net.

5. Remark (*Workflow net* [Aal1996])
A *Workflow net* WN is an ordinary net, which satisfies two conditions:

- It has two special places: a *source place* i with no ingoing arc, i.e. $pre(i) = \varnothing$, and a *sink place* o with no outgoing arc, i.e. $post(o) = \varnothing$, and

- after adding a new transition t with $pre(t) = o$ and $post(t) = i$ the net WN extends to a strongly connected net $\overline{WN}$.

By fusing the source and sink place of a Workflow net one obtains an ordinary net with baseplace, which is covered by circuits passing through the baseplace. Skeletons and high-nets of elementary Boolean loop systems are contained in the class of nets obtained from the fusion of Workflow nets. But the inclusion is proper, because elementary Boolean loop systems do not contain any loops, which omit the baseplace.

An a priori acyclic AND/XOR-EPC translates into a bipolar system. Its high-system is the same free-choice system, which has been detected by Genrich-Thiagarajan ([GT1984], Chap. 3). And it is exactly the free-choice system, which has been proposed by van der Aalst [Aal1999] as the translation of an AND/XOR-EPC.

6. Theorem *(Deriving the free-choice semantics from the Boolean semantics)*
The semantics of an a priori acyclic AND/XOR-EPC induced by the high-system of its Boolean system according to [LSW1998] equals the free-choice semantics according to [Aal1999].

Theorem 6 follows at once from the definition of the high-system. Accordingly the free-choice semantics of an a priori acyclic AND/XOR-EPC can be obtained from its Boolean semantics. But the Boolean semantics achieves more: It is defined for EPC connectors of arbitrary logical type. In particular, it provides the OR-join with a local semantics.

Note. In ([LSW1998], Def. 12) we proposed to model OR-connectors always as a completed alternative with a pair of split- and join-connectors. At this point we would like to emphasize, that this proposal is a modeling rule for EPCs, which facilitates their *analysis*. But the Boolean *semantics* of an OR-connector is well-defined also for isolated, unpaired connectors.

Van der Aalst discusses the OR-connector in the context of a completed OR-alternative ([Aal1999], Fig. 10). When marking both pre-places in Fig. 11 of transition (1,1) belonging to the OR-join, then not only transition (1,1), but also transition (1,0) and transition (0,1) are enabled. Van der Aalst rejects the gray subnet of the OR-join in Fig. 11 as a local semantics of the EPC connector. Instead he discusses three alternatives: First the resolution of an OR according to the formula

x OR y <=> (x AND y) XOR (x XOR y),

secondly the introduction of additional places equal to the two white low-places in Fig. 11 and thirdly a modification of the firing rule of Petri nets to a non-local rule. He rejects all three procedures. Note: Not only the second proposal but also the first one, which duplicates both branches of an OR-split, is not applicable for isolated OR-splits. In [Aal1999] the author abstains from defining a local semantics of the OR-connector. With different collaborators he devotes a series of successive papers to

the task to provide the OR-connector either with a non-local semantics, e.g., [WEAH2005] [4], or with a local semantics, e.g., [MA2006].
We follow the rejection of the high-net as a candidate for translating the OR-join. Fig. 11 shows the reason, why the high-net is insufficient to capture the intended semantics of an OR-join: The three transitions (1,1), (1,0) and (0,1) from the flattening of the Boolean OR-transition belong to a common cluster. Marking both pre-places from the low-net is necessary to decide which transition of the cluster is enabled. Apparently this cluster obstructs the free-choice property. In our opinion this fact prevents any translation of an OR-join into a free-choice net.

Analysis of behaviour

Safeness and liveliness are two properties to describe the behaviour of a Petri net. As a third property we now add being cyclic. A Boolean system is called *cyclic*, if the initial marking is reachable from any reachable marking. A cyclic Boolean system can always return to its initial state.

7. Lemma *(Liveliness and absence of frozen tokens)*
Consider an elementary Boolean loop system $EBLS = (BN, p, \mu)$.
i) The skeleton of $EBLS$ is a live and safe T-net.
ii) If $EBLS$ is high-live and safe, then it is also cyclic and any reachable marking, which marks the baseplace p, equals the initial marking μ.
iii) The high-system of a bipolar system $EBLS$ has no frozen tokens, if it is live and safe.

Proof. i) The skeleton is a strongly connected T-system. Each place belongs to a circuit and every circuit is marked at μ^{skel} with the single token at p^{skel}. Hence the skeleton is live and safe.
ii) We apply part i). It is well-known, that a live T-system is also cyclic. Moreover two reachable markings agree, if they mark every circuit with the same number of tokens ([DES1995], Theor. 3.21). Because every circuit passes through p^{skel}, the initial marking μ^{skel} is the only reachable marking with a token at p^{skel}. Now we lift this result from the skeleton to the Boolean system $EBLS$. By assumption $EBLS$ is high-live and therefore deadlock-free. Hence any enabled occurrence sequence of its skeleton lifts to an enabled occurrence sequence of $EBLS$. Consider a reachable marking μ_1 of $EBLS$. Any occurrence sequence of the skeleton, which is enabled at μ_1^{skel} and after firing creates the initial marking, lifts to an occurrence sequence of $EBLS$, which is enabled at μ_1 and after firing creates a marking μ_2 with a single token. This token marks the baseplace p. Because an elementary Boolean loop

[4] In Definition 16 of this paper there has to be added the requirement, that firing one of the resulting transitions in question consumes exactly one token from every marked pre-place.

system is faithful with respect to activation, this token must be a high-token, which proves $\mu = \mu_2$.

iii) Every elementary circuit of *BN* as well as of its high-net passes through respectively p and p^{high}. Because P-components and T-components of the high-net are strongly connected, they are covered by elementary circuits. Hence all P-components and all T-components of the high-net pass through p^{high}. According to Remark 2 the high-system has no frozen tokens, q. e. d.

Lemma 7, part ii) demonstrates a property, which has been called sound in the context of Workflow nets.

8. Remark (*Soundness* [Aal1996])
A Workflow net *WN* is named *sound*, if marking the source place with a single token provides an initial marking μ, such that (WN, μ) has the following properties:

- Each reachable marking has a follower marking, which marks the sink place,
- each reachable marking, which marks the sink place, does not contain a second token and
- for each transition t of *WN* there exists a reachable marking, which enables t.

It is well-known, that a Workflow net *WN* is sound, iff the extended net $\overline{WN}$ is live and bounded with respect to the induced marking $\overline{\mu}$.

Theorem 6 rises the following question concerning a priori acyclic AND/XOR-EPCs: How does soundness with respect to their free-choice semantics relate to soundness with respect to their Boolean semantics? The answer is given by Theorem 9, which heavily depends on Theorem 3:

9. Theorem *(Behaviour with respect to the Boolean and the free-choice semantics)*
Any a priori acyclic AND/XOR-EPC translates into a live and safe free-choice system, iff it translates into a high-live and safe Boolean system.

Proof. The elementary Boolean loop *EBLS* of the EPC is a bipolar system. Its high-system $EBLS^{high}$ defines the free-choice semantics of the EPC according to Theorem 6. If the bipolar system *EBLS* is high-live and safe, then its high-system $EBLS^{high}$ is live and safe according to Theorem 3. Conversely: According to Lemma 7, part i) the skeleton $EBLS^{skel}$ is live and safe. In order to apply Theorem 3 it suffices to show, that $EBLS^{high}$ has no frozen tokens. This follows from Lemma 7, part iii), q. e. d.

The two EPCs from Fig. 3 and Fig. 4 translate into elementary Boolean systems with the same skeleton, which is a live and safe T-system. But their corresponding high-systems differ. Both high-systems are safe. But the high-system of the EPC from Fig. 4 is also live, while the high-system of the EPC from Fig. 3 has a reachable marking, which is a deadlock. According to Lemma 7, part iii) and Theorem 3 the Boolean system of the EPC from Fig. 4 is high-live and safe, while the Boolean system of the

EPC from Fig. 3 is not high-live and safe. Hence the EPC from Fig. 4 is well-formed, while the EPC from Fig. 3 lacks well-formedness.
The AND/XOR-EPC from Fig. 9 translates into an elementary Boolean loop system, which is a bipolar system. Hence the Boolean semantics of the EPC is well-defined. The skeleton of the bipolar system is live and safe. The bipolar system is not high-live and safe according to Theorem 3. Because Theorem 4 shows, that the high-net is not well-formed due to the existence of TP-handles. The example is taken from ([Kin2006], Fig. 4). Kindler proves, that the EPC lacks a non-local ideal semantics.

EPCs with circuits

In the present section we consider a priori cyclic EPCs. They contain circuits already before attaching their start/end connection.
If a circuit of an EPC has a unique XOR-node, which serves as entry-point as well as exit-point, then the circuit is considered a loop and the intended semantics of the XOR-node is as following: The control flow either enters the loop or it passes the loop or it leaves the loop. Concerning entering or passing the node has the same semantics as an XOR-split: Either the control flow enters a given branch of the alternative or it passes this branch. After the control flow has entered the loop scope, it is decoupled from the rest of the net. There is no interaction between the control flow inside the loop and the control flow outside.
The free-choice semantics of an AND/XOR-EPCs considers only a single type of tokens which correspond to high-tokens. Hence it is suggesting that the free-choice semantics does not discriminate between an XOR-split at the start of a loop and an XOR-split at the start of an alternative. In both cases the XOR-splits translates into a place, branched in forward direction. A similar reasoning holds for XOR-joins.
The situation is different for the Boolean semantics. If the control flow enters a given branch at an XOR-split, it has decided not to activate the other branch. Hence the Boolean system propagates a high-token into the activated branch and a low-token along the other one. On the other hand: If the control flow enters the loop at an XOR-split, the decision about activation or not-activation of the loop's complement is postponed until the control flow exits from the loop. Hence the Boolean semantics propagates a high-token into the loop and no other token into its complement. Therefore the loop entry as well as the loop exit cannot be represented by a Boolean XOR-transition. Instead one has to use a branching place.
A Boolean system treats the high-token alike in both situations, but it discriminates concerning the low-token. Free-choice systems, which do not care about low-tokens, can deal with both situations in the same fashion.

In [LSW1998] we have undertaken a first step to deal with circuits of an EPC. We have investigated *well-structured loops*. A well-structured loop is an elementary circuit, which starts and ends with the same XOR-connector. The connector has two entries and two exits. EPCs with all elementary circuits of this type translate into a *Boolean loop system* $BLS = (BN, p, \mu)$: The Boolean net BN, being a *Boolean loop tree*, is covered by a finite family of elementary Boolean loops. Each elementary Boolean loop represents a single loop with its baseplace serving as loop-entry and as

loop-exit. Each loop is linked up in a tree-like fashion by fusing its baseplace with a distinguished place of the loop's complement ([LSW1998], Def. 9, 10). The distinguished place $p \in BN$ is the baseplace of the root and the marking μ of BN marks p with a single high-token and has no other tokens.

The *Boolean loop tree* as well as the original EPC is named *well-formed*, iff BLS is high-live and safe. Analogously to Lemma 7 one can prove the following result.

10. Remark *(Boolean loop system)*
Consider a Boolean loop system $BLS = (BN, p, \mu)$.

i) The skeleton of BLS is live and safe.
ii) A Boolean loop tree is well-formed, iff each of its elementary Boolean loops is well-formed.
iii) If BLS is high-live and safe, then it is also cyclic and any reachable marking, which marks the baseplace p, equals the initial marking μ.

Theorem 9 and Remark 10 imply the following Theorem 11.

11. Theorem *(Behaviour with respect to the Boolean and the free-choice semantics)*
Any AND/XOR-EPC, which has only well-structured loops, translates into a live and safe free-choice system, iff it translates into a high-live and safe Boolean loop system.

Thanks to Theorem 11 and 4 it is easy to verify the well-formedness of an AND/XOR-EPC with only well-structured loops: One checks the high-net of each elementary Boolean loop for the different types of handles and bridges. If the high-net is well-formed, then its basemarking is live and safe, because each P-component is marked with the single token at the basepoint (cf. [DE1995], Theor. 5.8 and 5.9).

The EPC from Fig. 1 with a single loop is well-structured. Fig. 13 shows its Boolean loop system BLS. It defines the Boolean semantics of the EPC. The Boolean loop tree is covered by two elementary loops, the root with baseplace p_0 and a second loop with baseplace p_1. Each elementary Boolean loop, when marking its baseplace with a high-token, is a high-live and safe elementary Boolean loop system.

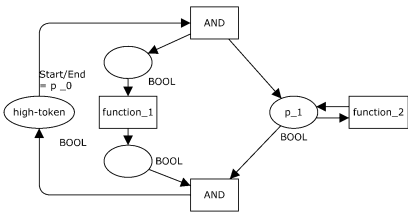

Fig. 13 Boolean loop system BLS

12. Remark *(Counterexample)*
Apparently one can disregard the loop character of the EPC from Fig. 1 and translate the AND/XOR-EPC directly into the bipolar system BS from Fig. 14. Its initial

marking has an additional low-token to assure liveliness of the skeleton. But BS does not provide the intended semantics of the given EPC. It is not high-live: Depending on the selected firing mode of the XOR-transition one can reach a deadlock.

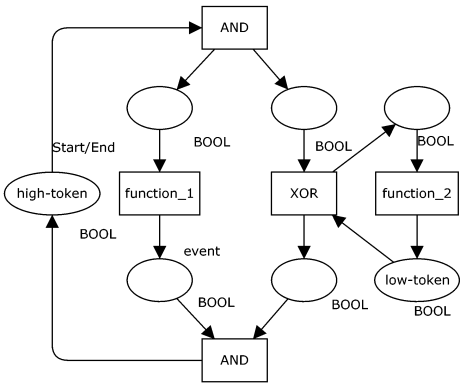

Fig. 14 Bipolar system BS

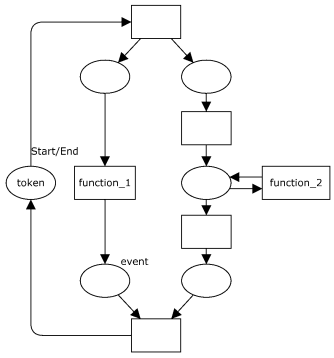

Fig. 15 High-system FS of BS

Fig. 15 shows the high-system FS of BS from Fig. 14. The free-choice system FS is live and safe, but it has a frozen token: After firing the split-transition one can fire the loop ("function_2") an infinite time without moving the token at the pre-place "event" of the join-transition. Hence Theorem 3 does not apply to this situation.
Note that FS defines the free-choice semantics of the EPC from Fig. 1 according to van Aalst. It also results from place fusing the high-systems of the two loop components of BLS from Fig. 13.
The EPC from Fig. 1 illustrates the following issue, mentioned in the beginning of this section: The Boolean semantics of an EPC discriminates between an XOR-connector either as entry point of a well-structured loop (Fig. 13) or as XOR-split of an alternative (Fig. 14). The free-choice semantics (Fig. 15) considers both the same.

Due to its specified composition from elementary loops a Boolean loop tree excludes any circuits as in Fig. 2, Fig. 5, Fig. 6 or Fig. 7. It does not provide a Boolean semantics for the corresponding EPCs. We do not even know, which process is intended by the EPC from Fig. 2. It seems reasonable to consider the event with the outgoing arc as start event. But what is the final state of the EPC, does there exist a final state at all, does the process terminate?

We now take a step forward in the investigation of EPCs with circuits. The three EPCs from Fig. 5, Fig. 6 and Fig. 7 have in common the existence of an elementary circuit, which neither passes through the baseplace nor forms a well-structured loop.

The EPC from Fig. 5 models the following scenario: You need an information, which can be obtained from two information offices. Therefore you decide, which office to ring up first. Assume you ring up the first office first ("function 1"). Either it answers the call or you ring up the second office ("function 2"). Now either the second office

answers the call or you try again the first office … . After leaving the loop you complete your work ("function 3").[5]

The elementary circuit passing through "function 1" and "function 2" is a loop with two entry points as well as with two exit points. We translate each XOR-connector, which corresponds to an entry or exit point, into a branching place of the Petri net. We have to assure, that no reachable marking marks the loop with more than one token. Therefore we connect all entry places with a single ENTER-connector to the places corresponding to the events "to_left" and "to_right". The connector waits for information, that one of the two events happens and the other cannot happen. Then its firing mode reduces the pair of high- and low-token at its two preplaces to a single high-token. This high-token initializes the loop at the entry point corresponding to the triggering event. The ENTER-connector has two high-modes and two low-modes according to Fig. 16. Note, that each firing mode consumes two tokens, but creates only one token.

Connector	Binding elements
ENTER	left_high := (high, low; high, -), right_high := (low, high; -, high), left_low := (low, low; low, -), right_low := (low, low; -, low)

Fig. 16 Binding elements of the ENTER-connector

Symmetrically, there is an EXIT-connector with analogous firing modes. It splits an arriving high-token into a pair of high- and low-tokens and an arriving low-token into a pair of low-tokens. For syntactical reasons we have to add inside the loop four unbranched transitions.

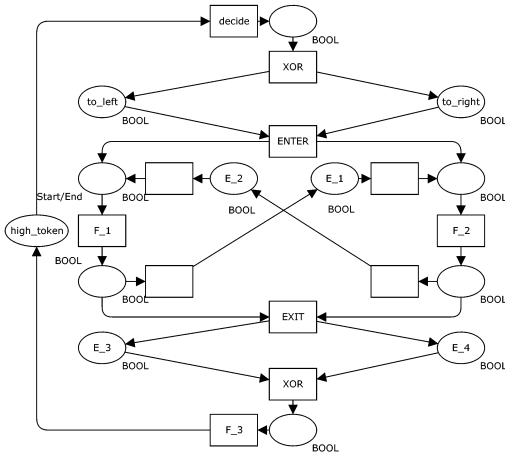

Fig. 17 BS as translation of the EPC from Fig. 5

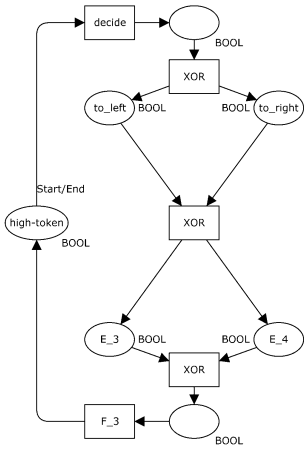

Fig. 18 Abstraction BS_{abstr}

[5] I thank Walter Vogler for communicating this example to me.

Fig. 17 shows the final Petri net BS. It is safe and every high-mode is live.

The EPC from Fig. 6 has been discovered by Kindler as an example of an EPC with two non-local, but ideal semantics. We refrain from translating the EPC into a Boolean system: The two exit points of the loop are represented by AND-connectors, but we do not know about a corresponding loop semantics.

The EPC from Fig. 7 has been discovered by Kindler as an example of an EPC without any ideal semantics at all. Instead, it has a pair of non-local semantics, which are best approximations of an intended ideal semantics. The loop has three exit points, which are represented by AND-connectors. By similar reasons we refrain from translating the EPC into a Boolean system.

The Boolean semantics of a hierarchy of EPCs

The Boolean semantics of an EPC is defined by a coloured Petri net. Hence the semantics of a hierarchy of EPCs is well-defined as soon as one has at hand an analogous concept for coloured Petri nets.
A hierarchy of coloured Petri nets is a family of coloured Petri nets, which are linked by morphisms in a tree-like fashion. We have proposed the concept of a morphism between coloured Petri nets in a previous paper ([Weh2007], Def. 2.6). The fibres of such a morphism refine a place by a place-bordered subnet, a transition by a transition-bordered subnet, a binding element by a T-flow and a token element by a P-flow. The morphism maps certain occurrence sequences of the Petri net in the domain to occurrence sequences of the Petri net in the image. Apparently the semantics of all Petri nets is independent from each other.

A simple example for such a morphism is the abstraction map

$$abstr : BS \longrightarrow BS_{abstr}$$

from the Petri net BS of Fig. 17 onto the Petri net BS_{abstr} of Fig. 18: The elementary circuit in the centre of BS as well as the transitions ENTER and EXIT map onto the central XOR-transition of BS_{abstr}. All other nodes and arcs from BS map injectively onto the corresponding net element of BS_{abstr}. Hence the EPC from Fig. 5, which translates into BS, abstracts onto the EPC corresponding to BS_{abstr}.

5 Conclusion

The task to formalize the intended semantics of EPCs is an ill-posed problem. As Kindler rightly remarks “Actually, there is not a single well-accepted informal semantics for EPCs. There are many variants and modifications.”

We are skeptic about all attempts to enrich EPCs with additional concepts like tokens, internal states, reset arcs, context, time … . These concepts are either introduced in an

ad-hoc fashion or they are borrowed from formal process languages, which have their own well-defined semantics. Of course such concepts are useful, the concept of state is even indispensable for process modeling. Instead of borrowing concepts from other languages, we are advocating to first translate the EPC into a target language and then to argue on the solid ground of that formal language.

Since 1994 we have accomplished numerous process modeling projects in the commercial field. Never there was a problem to read a given EPC as a Boolean net - at least after untangling the intended loops. And after discussing with the domain experts the intended start/end connection it was also possible to provide the initial marking. Of course this type of discussion with the customer from the user department seldom uses explicit formal reasoning. Neither it applies any algorithm, which transforms a given EPC into a well-formed one. Main ingredients are the consultants experience with similar cases and her knowledge about the standards and pitfalls of process modeling.
Questions concerning the intended semantics of EPCs are challenging from a theoretical point of view. But in the context of commercial applications they are not debated. We explain this lack of interest in a formal semantics with the imprecision of "real-world" EPCs. Up to now, commercial projects had no need to model precise EPCs, which are executable like a program.

In our opinion the question of *non-local* semantics for EPCs has been settled by the remarkable result of Kindler: There is no single transition relation, which captures the non-local semantics intended by Nüttgens and Rump. In general, there exist two related transition relations, which approximate best the intended semantics from two different directions.

The problem to handle EPC circuits in the context of Boolean systems is not yet solved. The example from Fig. 5 and the proposal of its translation in Fig. 17 mark the starting point of some work in progress. We study Boolean systems with components corresponding to loops with multiple entry and exit points.

Concerning Boolean systems another question seems interesting from a theoretical point of view. For a high-live and safe bipolar system all binding elements of the low-system are irrelevant for the flow of the high-tokens. The next step would be to study high-live and safe Boolean systems with transitions of logical type L_XOR or OR: Does there exist a maximal, non-empty subsystem of the low-system, which can be factored out without disturbing the flow of the high-tokens? One of the simplest Boolean systems in question is the Boolean system of the EPC from Fig. 8.

Acknowledgments

I thank the anonymous referees as well as Peter Rittgen for their helpful comments during the review. I tried to follow their suggestions and to answer their questions.

6 References

[Aal1996] *van der Aalst, Wil*: Structural Characterizations of Sound Workflow Nets. Computing Science Reports 96/23, Eindhoven University of Technology, Eindhoven, 1996

[Aal1999] *van der Aalst, Wil*: Formalization and verification of event-driven process chains. Information & Software Technology, 41 (10), 1999, p. 639-650

[BD1990] *Best, Eike; Desel, Jörg*: Partial Order Behaviour and Structure of Petri Nets. Formal Aspects of Computing (1990), p. 123-138

[CS1994] *Chen, R.; Scheer, August-Wilhelm*: Modellierung von Prozeßketten mittels Petri-Netz Theorie. Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 107, Saarbrücken 1994

[DE1995] *Desel, Jörg, Esparza, Javier*: Free Choice Petri Nets. Cambridge University Press, Cambridge 1995

[ES1990] *Esparza, Javier; Silva, Manuel*: Circuits, Handles, Bridges and Nets. In: *Rozenberg, Grzegorz* (Ed.) Advances in Petri nets 1990. Lecture Notes in Computer Science, vol. 483, Springer, Berlin 1990, p. 210-242

[GL2005] *Gruhn, Volker; Laue, Ralf*: Einfache EPK-Semantik durch praxistaugliche Stilregeln. In: *Nüttgens, Markus; Rump, Frank* (Hrsg.): EPK 2005. Geschäftsprozeßmanagement mit Ereignisgesteuerten Prozeßketten. Proceedings, Hamburg 2005

[GT1984] *Genrich, H.J.; Thiagarajan, P.S.*: A Theory of Bipolar Synchronization Schemes. Theoretical Computer Science 30 (1984), p. 241-318

[Kin2006] *Kindler, Ekkhart*: On the semantics of EPCs: Resolving the vicious circle. Data & Knowledge Engineering 56, (2006), p. 23-40

[KNS1992] *Keller, Gerhard; Nüttgens, Markus; Scheer, August-Wilhelm*: Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)". Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, Saarbrücken 1992

[LSW1998] *Langner, Peter; Schneider, Christoph; Wehler, Joachim*: Petri Net based Certification of Event-driven Process Chains. In: *Desel, Jörg; Silva, Manuel* (Eds.): Application and Theory of Petri Nets 1998. Lecture notes in Computer science, vol. 1420. Springer, Berlin et al. 1998

[NR2002] *Nüttgens, Markus; Rump, Frank*: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: *Desel, Jörg; Wesel, M.* (Hrsg.): Promise 2002 – Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen. GI Lecture Notes in Informatics P-21, pp. 64-77, Gesellschaft für Informatik, 2002

[MA2006] *Mendling, Jan; van der Aalst, Wil*: Towards EPC Semantics based on State and Context. In: *Nüttgens, Marcus; Rump, Frank*: EPK 2006. Geschäftsprozeßmanagement mit Ereignisgesteuerten Prozessketten. 5. Workshop der Gesellschaft für Informatik, Wien, 2006

[Rit2000] *Rittgen, Peter*: Paving the road to Business Process Automation. In Proc. of the Europ. Conf. on Information Systems (ECIS), p. 313-319, 2000

[Rod1997] *Rodenhagen, Jörg*: Darstellung ereignisgesteuerter Prozeßketten (EPK) mit Hilfe von Petrinetzen. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Hamburg 1997

[Rum1999] *Rump, Frank*: Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten. Teuber, Stuttgart et al. 1999

[Weh2007] *Wehler, Joachim*: Free Choice Petri Nets without frozen tokens and Bipolar Synchronization Systems. arXiv: cs/0609095, 2007

[WEAH2005] *Wynn, Moe; Edmond, David; van der Aalst, Wil; ter Hofstede, Arthur*: Achieving a General, Formal and Decidable Approach to the OR-join in Workflow using Reset nets. In *Ciardo, G.; Darondeau, P.* (Eds.): Applications and Theory of Petri Nets 2005, vol. 3536 of Lecture Notes in Computer Science, p. 423-443. Springer, Berlin et al. 2005