Technical Report, Reihe Informatik tr-ri-03-243
Institut für Informatik, Universität Paderborn, August 2003

On the semantics of EPCs: A framework for resolving the vicious circle

Ekkart Kindler
Computer Science Department, University of Paderborn, Germany
`kindler@upb.de`

Abstract: One of the most debatable features of *Event driven Process Chains* (EPCs) is their non-local semantics. Most non-local semantics for EPCs either have a formal flaw or are given informally only. In fact, it can be shown that there is no formal semantics that precisely captures the informal idea of the non-local semantics of EPCs.

In this paper, we formally define a non-local semantics for EPCs in the best way possible. To this end, we use standard techniques from fixed point theory.

Actually, there are several choices when defining non-local semantics for EPCs. These choices, however, do not compromise the application of the underlying fixed point theory. The mathematics used in this paper, can be considered as a semantical framework for formally defining non-local semantics for EPCs. This framework can be used for the discussion and, eventually, for settling the discussion on the semantics of EPCs.

1 Introduction

Ever since the definition of *Event driven Process Chains* (*EPCs*) in the early 90ties [KNS92], there has been a debate on their precise semantics. One feature recurrently provoking a debate is the *non-locality* of the semantics of the OR-join and the XOR-join connectors. On the one hand, this non-local semantics helps to simplify many models. On the other hand, there is no satisfactory formalization of this semantics yet. Many formalizations simply ignore the non-local semantics of the OR-join and XOR-join connectors; others provide ad-hoc solutions. Rittgen [Rit00] discusses some aspects of this problem and some approaches towards defining more satisfactory semantics for EPCs, which help resolving the problems. One concept proposed by Langner, Schneider and Wehler [LSW98], for example, is some additional synchronization, which is similar to *dead path elimination* in IBM's MQ Series[1] process model [LA94]. Rittgen himself introduces some new syntax for EPCs in order to (partially) cure the problem [Rit00].

One reason for the ongoing debate on the semantics of EPCs is inherent to the non-locality of the informal semantics for EPCs: In essence, a non-local semantics refers to itself in its own definition (see Sect. 2 for more details). Even worse, this self-reference occurs under a negation, which easily results in a mathematical, conceptual, and technical short-

[1]IBM MQ Series workflow was called FlowMark at that time.

circuit[2]. In [vdADK02], we pinpoint these arguments and prove that there cannot be a formal semantics that fully complies with the informal semantics of EPCs – the *vicious circle*.

From a theoretical point of view, the result of [vdADK02] should have settled the debate – against non-local semantics for EPCs. In practice, however, there are many EPC models exploiting the non-local semantics. Therefore, we set out to give a mathematically sound semantics for EPCs that comes as close as possible to the informal semantics[3], which will be presented in this paper. In fact, we provide a framework for easily defining non-local semantics for EPCs. Technically, this framework resolves the vicious circle by distinguishing two related transition relations for EPCs and by using fixed point theory for capturing self-references.

What is more, this framework comes with a characterization of *unclean* EPCs for each concrete definition of a semantics. These unclean EPCs are those that do not exactly capture the informal semantics and, therefore, are ambiguous. Maybe, this framework helps to, eventually, settle the debate about 'the semantics' of EPCs – this way, resolving the vicious circle of never-ending discussions.

2 The problem

In this section, we present the syntax and the non-local semantics of EPCs and discuss the problem with the informal semantics of EPCs. Here, we can give a rough outline only. For a more detailed motivation of EPCs, their syntax, and their semantics, we refer to [KNS92, NR02]. For a more detailed exposition of the problems with the non-local semantics of EPCs, we refer to [vdADK02].

Figure 1 shows an example of an EPC. It consists of three kinds of nodes: *events* graphically represented as hexagons, *functions* represented as rounded boxes, and *connectors* graphically represented as circles. The dashed arcs between the different nodes represent the *control flow*. The two black circles do not belong to the EPC itself; they represent a *state* of an EPC. A state, basically, assigns a number of *process folders* to each arc of the EPC[4]. Each black circle represents a process folder at the corresponding arc.

The semantics of an EPC defines how process folders are propagated through an EPC. Clearly, this depends on the involved node. For events and functions, a process folder is simply propagated from the incoming arc to the outgoing arc. The *transition relation* for events and functions is graphically represented in the top row of Fig. 2 (a. and b.). For connectors, the propagation of folders depends on the type of the connector (*AND*, *OR*, resp. *XOR*) and whether it is a *join* or a *split connector*. Figure 2 shows the transition

[2]In fact, it was a 'short-circuit' in the formal definition of a semantics for EPCs in [NR02] which, again, attracted our attention to this problem.

[3]Actually, there is not a single well-accepted informal semantics for EPCs. We refer to the informal semantics presented by Nüttgens and Rump [NR02].

[4]In other formalizations, process folders are assigned to nodes rather than to arcs. Assigning folders to arcs significantly simplifies the technical presentation of our ideas. Therefore, we have chosen to assign folders to arcs. But, this is not inherent to our approach.

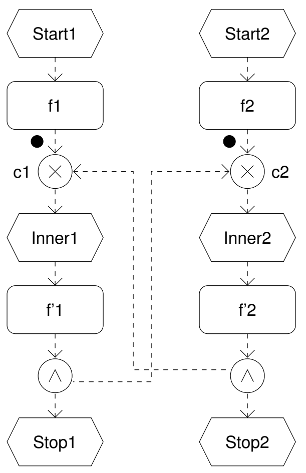

Figure 1: An EPC

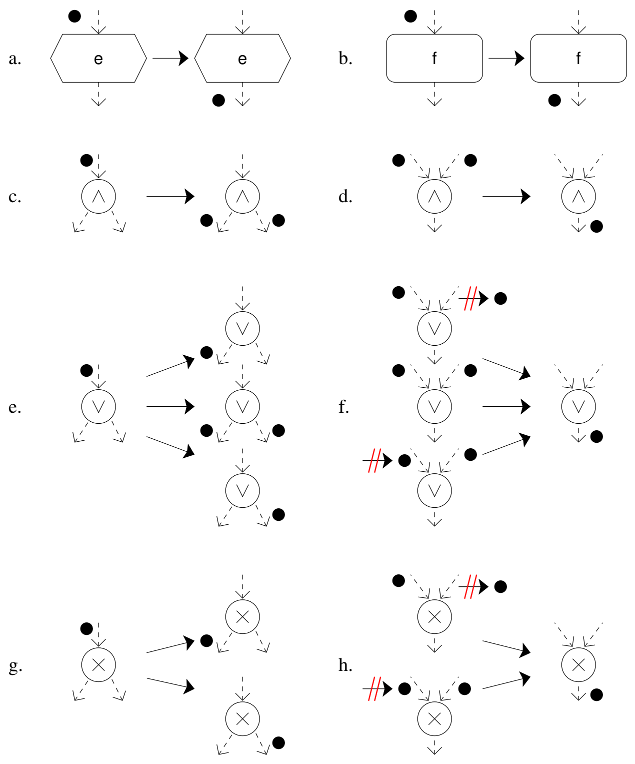

Figure 2: The transition relation for the different nodes

relation for the different connectors. For example, the AND-split connector (c.) propagates a folder from its incoming arc to all outgoing arcs. The AND-join connector (d.) needs one folder on each incoming arc; which are then propagated to a single folder on the outgoing arc.

The more interesting connectors are the OR-join and the XOR-join. Here, we focus on the XOR-join connector. An XOR-join connector (h.) waits for a folder on one incoming arc, which is then propagated to the outgoing arc. But, there is one additional condition: The XOR-join must not propagate the folder, if there is or there could arrive a folder on the other incoming arc. In Fig. 2.h, this is represented by a label $\not\!\!\!\not\!\rightarrow\bullet$ at the other arc. Note that this condition cannot be checked locally for the XOR-join connector, since whether a folder could arrive at the other arc or not depends on the overall behaviour of the EPC. Therefore, we call the semantics of the XOR-join *non-local*. Likewise, the OR-join connector (f.) has a non-local semantics. Note that, in both cases, the additional condition refers to the transition relation itself and that the transition relation occurs under a negation in this condition.

Basically, the intended non-local behaviour has two problems, a technical one and a conceptual one (see [vdADK02] for details):

1. In the definition of the transition relation, we refer to the the transition relation itself, which easily results in definitions that are not mathematically sound. In principle, this problem could be avoided by using some kind of fixed point semantics (the standard trick for giving semantics to objects that refer to themselves). The problem with the non-local semantics of EPCs, however, is that a fixed point does not exist in all cases.

2. The conceptual problem is that, for some EPCs, there is no formal semantics that exactly captures the informal semantics. The reason is that the self-reference occurs under a negation. For example, consider the EPC from Fig. 1 with one process folder on one of the incoming arcs of each XOR-join c_1 and c_2. For symmetry reasons, either both of them should be able to propagate this folder or both should not be able to propagate the folder. But, either way contradicts the definition of the informal semantics of the XOR-join.

One purpose of this paper, is to define a non-local semantics for EPCs that is mathematically sound and comes as close as possible to the informal semantics of EPCs. What is more, this definition allows us identifying problematic EPCs, i. e. EPCs for which the informal semantics is ambiguous.

The main purpose of this paper, however, is the presentation of a framework for defining and discussing different kinds of non-local semantics. The concrete semantics defined here, serves as an example for presenting the framework. The definition of ‘the semantics’ of EPCs is left to the EPC community.

3 The syntax of EPCs

In this section, we formalize the syntax of EPCs. Since the focus of this paper is on a formalization of the semantics of EPCs, we will omit some syntactical restrictions that are not relevant for our semantical considerations; moreover, we consider *flat EPCs* only, i. e. EPCs without subprocesses.

Basically, an EPC is a graph, i. e. it consist of nodes and arcs. In order to express some of the syntactical restrictions, we first introduce a simple notation for the ingoing and outgoing arcs of a node:

Notation 1 (Ingoing and outgoing arcs)

Let N be a set of *nodes* and let $A \subseteq N \times N$ be a binary relation over N, the *arcs*. For each node $n \in N$, we define the set of its *ingoing arcs* $n_{in} = \{(x, n) \mid (x, n) \in A\}$, and we define the set of its *outgoing arcs* $n_{out} = \{(n, y) \mid (n, y) \in A\}$.

An EPC consists of three different kinds of nodes, *events*, *functions*, and *connectors*, which are connected by *control flow arcs*. A connector can be either an AND-, an OR-, or an XOR-connector, which is indicated by labelling the connector correspondingly. Each function has exactly one ingoing and one outgoing arc, whereas each event has at most one ingoing and at most one outgoing arc. A connector has multiple ingoing arcs and one outgoing arc, or it has one ingoing arc and multiple outgoing arcs:

Definition 2 (EPC)

An *EPC* $M = (E, F, C, l, A)$ consists of three pairwise disjoint sets E, F, and C, a mapping $l : C \to \{and, or, xor\}$ and a binary relation $A \subseteq (E \cup F \cup C) \times (E \cup F \cup C)$ such that

- $|e_{in}| \leq 1$ and $|e_{out}| \leq 1$ for each $e \in E$,
- $|f_{in}| = |f_{out}| = 1$ for each $f \in F$, and
- $|c_{in}| \geq 1$ and $|c_{out}| \geq 1$ and either $|c_{in}| = 1$ or $|c_{out}| = 1$ for each $c \in C$.

An element of E is called an *event*, an element of F is called a *function*, an element of C is called a *connector*, and an element of A is called a *control flow* arc.

Note, that we have omitted some syntactical restrictions for EPCs:

- Functions and events should alternate along the control flow.
- The OR-split and the XOR-split connectors should be preceded by a function, which determines to which direction process folders are propagated.
- There should be no cycle of control flow that consist of connectors only.

Though these requirements are important from a pragmatical point of view, these restrictions are not necessary for defining the semantics of EPCs. So, we do not formalize these restrictions here. For a complete exposition of the syntax of EPCs, we refer to [NR02] .

In the definition of the semantics, we need to distinguish among different types of connectors: AND-, OR-, and XOR-, each of which can be either a *split* or a *join connector*. The corresponding sets are defined below.

Notation 3 (Nodes and connectors)

For the rest of this paper, we fix an EPC $M = (E, F, C, l, A)$. We denote the set of all its nodes by $N = E \cup F \cup C$ and we define the following sets of connectors:

AND-split: $C_{as} = \{c \in C \mid l(c) = and \wedge |c_{in}| = 1\}$

AND-join: $C_{aj} = \{c \in C \mid l(c) = and \wedge |c_{out}| = 1\}$

OR-split: $C_{os} = \{c \in C \mid l(c) = or \wedge |c_{in}| = 1\}$

OR-join: $C_{oj} = \{c \in C \mid l(c) = or \wedge |c_{out}| = 1\}$

XOR-split: $C_{xs} = \{c \in C \mid l(c) = xor \wedge |c_{in}| = 1\}$

XOR-join: $C_{xj} = \{c \in C \mid l(c) = xor \wedge |c_{out}| = 1\}$

At last, we define the *states* of an EPC: An assignment, of a number of process folders to each arc of the EPC.

Definition 4 (State of an EPC)

For an EPC $M = (E, F, C, l, A)$, we call a mapping $\sigma : A \to \{0, 1\}$ a *state* of M. The set of all states of M is denoted by Σ.

For simplicity, at most one folder at each arc is allowed in a state. But, we will see in Sect. 7 that this restriction can be easily released.

Note that, in our definition, we assign the process folders to the control flow arcs of the EPC, whereas most other formalizations (e. g. [Rum99, Rit00, NR02]) assign the process folders to the nodes of the EPC. Though this choice is not essential for the definition of the semantics, it allows us a smoother technical presentation of the semantics[5]. In addition, this choice makes the nodes of the EPCs the active parts, whereas the arcs become the passive parts. We feel that this is a more natural conception of EPCs – but this is a very personal view.

4 The transition relation $R(P)$

The semantics of an EPCs will be defined in terms of a *transition relation* between its states. In order to identify the node involved in a transition, the transition relation is a

[5]For example, there is no need to extend an EPC with *pre-connectors* as introduced in [NR02].

relation $R \subseteq \Sigma \times N \times \Sigma$. A single transition $(\sigma, n, \sigma') \in R$, represents a change from state σ to σ' that corresponds to node n.

In order to define and to argue on transition relations, we introduce some notations for restricting it and for its induced reachability relation:

Notation 5 (Restriction and reachability)

For some transition relation $R \subseteq \Sigma \times N \times \Sigma$, and some subset $N' \subseteq N$, we define the *restriction of* R to N' as $R|_{N'} = \{(\sigma, n, \sigma') \in R \mid n \in N'\}$.

By slight abuse of notation, we define the *reachability relation* R^* of R as the reflexive and transitive closure of the binary relation $\rightarrow = \{(\sigma, \sigma') \in \Sigma \times \Sigma \mid \exists n \in N : (\sigma, n, \sigma') \in R\}$.

As discussed in Sect. 2, we need to refer to the transition relation in its own definition when defining a non-local semantics for the OR-join and the XOR-join connectors. Such cyclic references, however, are not possible in a sound mathematical definition. In order to resolve this cycle, we assume that some transition relation $P \subseteq \Sigma \times N \times \Sigma$ is given, and we define another transition relation $R \subseteq \Sigma \times N \times \Sigma$ where we refer to P whenever we would like to refer to R itself. We write $R(P)$ in order to make this dependency of R from the transition relation P explicit[6].

For now, we assume that P is some given transition relation. So, there is no cycle in the definition of $R(P)$. Later, in Sect. 5, we will use a standard trick of semantics, fixed points, for establishing a reference of R to itself. But, this need not bother us right now.

In the following definition, we first define one transition relation R_n for each node $n \in N$ of the EPC separately. The overall transition relation $R(P)$ is the union of the transition relations R_n of all nodes n. Figure 2 from Sect. 2 gives a rough overview on the definitions of R_n for the different nodes n. More details will be given after the definition.

Definition 6 (Transition relation $R(P)$)

Let P be a transition relation for an EPC M. For each node $n \in N$, we define the transition relation $R_n \subseteq \Sigma \times N \times \Sigma$ as follows:

a. For $n = e \in E$ with $e_{in} = \{i\}$ and $e_{out} = \{o\}$, we define $R_e \subseteq \Sigma \times N \times \Sigma$ by $(\sigma, e, \sigma') \in R_e$ iff $\sigma(i) = 1$, $\sigma(o) = 0$, $\sigma'(i) = 0$, $\sigma'(o) = 1$, and $\sigma'(a) = \sigma(a)$ for each $a \in A \setminus \{i, o\}$.

a' For $n = e \in E$ with $e_{in} = \emptyset$ or $e_{out} = \emptyset$, we define $R_e = \emptyset$.

b. For $n = f \in F$ with $f_{in} = \{i\}$ and $f_{out} = \{o\}$, we define $R_f \subseteq \Sigma \times N \times \Sigma$ by $(\sigma, f, \sigma') \in R_f$ iff $\sigma(i) = 1$, $\sigma(o) = 0$, $\sigma'(i) = 0$, $\sigma'(o) = 1$, and $\sigma'(a) = \sigma(a)$ for each $a \in A \setminus \{i, o\}$.

c. For $n = c \in C_{as}$ with $c_{in} = \{i\}$, we define $R_c \subseteq \Sigma \times N \times \Sigma$ by $(\sigma, c, \sigma') \in R_c$ iff $\sigma(i) = 1$, $\sigma(o) = 0$ for each $o \in c_{out}$, $\sigma'(i) = 0$, $\sigma'(o) = 1$ for each $o \in c_{out}$, and $\sigma'(a) = \sigma(a)$ for each $a \in A \setminus (\{i\} \cup c_{out})$.

[6]This reflects the fact, that actually R is a function that takes a transition relation P and gives another transition relation.

d. For $n = c \in C_{aj}$ with $c_{out} = \{o\}$, we define $R_c \subseteq \Sigma \times N \times \Sigma$ by $(\sigma, c, \sigma') \in R_c$ iff $\sigma(i) = 1$ for each $i \in c_{in}$, $\sigma(o) = 0$, $\sigma'(i) = 0$ for each $i \in c_{in}$, $\sigma'(o) = 1$, and $\sigma'(a) = \sigma(a)$ for each $a \in A \setminus (c_{in} \cup \{o\})$.

e. For $n = c \in C_{os}$ with $c_{in} = \{i\}$, we define $R_c \subseteq \Sigma \times N \times \Sigma$ by $(\sigma, c, \sigma') \in R_c$ iff, for some $S \subseteq c_{out}$ with $|S| \geq 1$, we have $\sigma(i) = 1$, $\sigma(o) = 0$ for each $o \in S$, $\sigma'(i) = 0$, $\sigma'(o) = 1$ for each $o \in S$, and $\sigma'(a) = \sigma(a)$ for each $a \in A \setminus (\{i\} \cup S)$.

f. For $n = c \in C_{oj}$ with $c_{out} = \{o\}$, we define $R_c \subseteq \Sigma \times N \times \Sigma$ by $(\sigma, c, \sigma') \in R_c$ iff, for some $S \subseteq c_{in}$ with $|S| \geq 1$, we have $\sigma(i) = 1$ for each $i \in S$, $\widehat{\sigma}(a) = 0$ for each $\widehat{\sigma}$ with $\sigma(P|_{N\setminus\{c\}})^*\widehat{\sigma}$ and for each $a \in c_{in} \setminus S$, $\sigma(o) = 0$, $\sigma'(i) = 0$ for each $i \in S$, $\sigma'(o) = 1$, and $\sigma'(a) = \sigma(a)$ for each $a \in A \setminus (S \cup \{o\})$.

g. For $n = c \in C_{xs}$ with $c_{in} = \{i\}$, we define $R_c \subseteq \Sigma \times N \times \Sigma$ by $(\sigma, c, \sigma') \in R_c$ iff, for some $o \in c_{out}$, we have $\sigma(i) = 1$, $\sigma(o) = 0$, $\sigma'(i) = 0$, $\sigma'(o) = 1$, and $\sigma'(a) = \sigma(a)$ for each $a \in A \setminus \{i, o\}$.

h. For $n = c \in C_{xj}$ with $c_{out} = \{o\}$, we define $R_c \subseteq \Sigma \times N \times \Sigma$ by $(\sigma, c, \sigma') \in R_c$ iff, for some $i \in c_{in}$, we have $\sigma(i) = 1$, $\widehat{\sigma}(a) = 0$ for each $\widehat{\sigma}$ with $\sigma(P|_{N\setminus\{c\}})^*\widehat{\sigma}$ and for each $a \in c_{in} \setminus \{i\}$, $\sigma(o) = 0$, $\sigma'(i) = 0$, $\sigma'(o) = 1$, and $\sigma'(a) = \sigma(a)$ for each $a \in A \setminus \{i, o\}$.

We define the *transition relation* $R(P) = \bigcup_{n \in N} R_n$

Below, we briefly discuss the different cases of the above definition, which makes the graphical representation of Fig. 2 more precise.

For an event e or a function f with exactly one ingoing arc and exactly one outgoing arc (a. and b.), the folder is propagated from the ingoing arc to the outgoing arc. Note that the folder is only propagated, when there is no folder on the outgoing arc. For start and end events (which have no incoming arc or have no outgoing arc) the transition relation is empty (a').

An AND-split connector (c.) propagates a folder from an incoming arc to all its outgoing arcs. However, it will be propagated only, when there are no process folders on the outgoing arcs. Likewise, the AND-join (d.) waits for a folder on each incoming arc and propagates it to the outgoing arc, provided there is no process folder on the outgoing arc yet.

The OR-split connector (e.) is similar to the end split. But, it can propagate a folder from the incoming arc to any (but at least one) of its outgoing arcs provided that there are no folders yet. The set of outgoing arcs to which the folders are propagated is denoted by S in the definition.

The OR-join connector (f.) is more involved because of its non-local semantics. When there is a folder on at least one of its incoming arcs $S \subseteq c_{in}$ and no folder can arrive (according to P) on the other arcs without the occurrence of c, the folder is propagated to the outgoing arc. In order to formalize that no folder can arrive on the other incoming arcs $a \in c_{in} \setminus S$, the definition refers to the states $\widehat{\sigma}$ that can be reached from σ (with respect to P) without the occurrence of c: $\sigma\ (P|_{N\setminus\{c\}})^*\ \widehat{\sigma}$.

The XOR-split operator (g.) propagates the folder from the incoming arc to exactly one of its outgoing arcs. The formalization is similar to the one of the AND-split and the OR-split.

The XOR-join (h.) is similar to the definition of the OR-join. Instead of selecting some set S of incoming arcs on which a folder must be present, we select exactly one incoming arc i. We require that no folder can arrive on the other incoming arcs (with respect to P) before the occurrence of c.

Altogether, the transition relation $R(P)$ is defined as the union of all individual transition relations R_n of the nodes n. Note that R_n depends on P only for the OR-join and the XOR-join, which are the only connectors with a non-local semantics.

Note that there are several options on how the transition relation for each node type could be defined. One option concerns the question whether a folder on an outgoing arc should block the propagation of a folder from the ingoing arc. Another option would be to define a local semantics for the XOR-join[7]. Actually, the purpose of this paper is not to discuss these question, but to provide a framework for formally defining a semantics for EPCs. The exact definition of the semantics for the individual connectors is left to the a standardization effort within the EPC community. Actually, the precise definition of $R(P)$ is not crucial for making the rest of our theory work (see Sect. 7 for details). There is only one crucial condition: $R(P)$ must be a monotonously decreasing function.

Lemma 7 ($R(P)$ is monotonously decreasing)

The operation $R(P)$ is monotonously decreasing; i. e. for two transition relations $P \subseteq P'$ we have $R(P) \supseteq R(P')$.

Proof: The only relations that depend on P and P' in the definition of $R(P)$ and $R(P')$ are the relations R_c for the OR-join and the XOR-join connector. For each state σ the set of states $\widehat{\sigma}$ reachable from σ with respect to P and P' must be checked for an additional condition ($\widehat{\sigma}(a) = 0$). With $P \subseteq P'$, we have $\{\widehat{\sigma} \mid \sigma\ (P|_{N\setminus\{c\}})^*\ \widehat{\sigma}\} \subseteq \{\widehat{\sigma} \mid \sigma\ (P'|_{N\setminus\{c\}})^*\ \widehat{\sigma}\}$. Clearly, a smaller set means less restrictions because less states must satisfy the condition. Therefore, there are more transitions in R_c for P than for P'. $\square$

5 The semantics of EPCs

As mentioned above, the semantics of an EPC should be some transition relation on the states of the EPC. In the previous section, we have not defined a transition relation, but we have defined a transition relation $R(P)$, which depends on some given transition relation P. In this section, we will use $R(P)$ for defining the semantics of EPCs. On a first glance, there are two different ways to define this semantics:

[7]There is a debate whether the XOR-join should have a non-local semantic or not. For example, Rittgen [Rit00] proposes a local semantics for the XOR-join connector. Here, we follow Nüttgens and Rump [NR02] in giving it a non-local semantics.

1. We could use some transition relation P and then calculate $R(P)$ as the semantics of the EPC. Actually, this idea is used in the semantics of YAWL [vdAtH02]. The problem, however, is that we need to define P first. And it is by no means clear how P should be defined, and which definition is the best. YAWL, for example, uses a simple transition relation that ignores all OR-join connectors. Similar ideas came up in private discussions with Nüttgens and Rump during the discussions on [NR02, vdADK02]. But each choice appears to be ad hoc in some way.

2. A better solution would be to find some P such that we have $P = R(P)$ – i. e. P is some fixed point of R. In that case, P refers to itself in its own definition $R(P)$ – the fixed point trick. Therefore, a fixed point P of $R(P)$ would exactly meet our initial intension, which justifies to call such a P an *ideal semantics* of the EPC. The problem with this definition, however, is that for some EPCs such a P does not exist; for others there are several different ideal semantics, and it is impossible to characterize one (e. g. the least fixed point with respect to set inclusion) among these ideal semantics to be 'the semantics' of the EPC (see Sect. 6 for a more detailed discussion).

Since both of the above approaches are unsatisfactory, we try a combination of both: We are now looking for a pair of transition relations (P, Q), such that we have $Q = R(P)$ and $P = R(Q)$, i. e. one transition relation is the input for the definition of the other. We will see that such pairs exist for each EPC, and that we can characterize some particular pair that will be used as 'the semantics' of the EPC.

In order to prove the existence of such pairs by standard fixed point theory, we define the domain D of all pairs of transition relations and an order relation $\preceq$, which forms a complete lattice on this domain. Moreover, we define a function φ such that the fixed points of φ are exactly the pairs meeting the above requirement.

Definition 8

For an EPC with nodes N and states Σ, we define the *domain* $D = 2^{\Sigma \times N \times \Sigma} \times 2^{\Sigma \times N \times \Sigma}$, and we define the relation $\preceq$ on D as follows: For two elements $d = (P, Q)$ and $d' = (P', Q')$, we define $d \preceq d'$ if $P \subseteq P'$ and $Q \supseteq Q'$.

On D, we define the function $\varphi : D \to D$ by $\varphi((P, Q)) = (R(Q), R(P))$.

Note that $(D, \preceq)$ is a complete lattice on D, because $\preceq$ inherits this structure from $\subseteq$ and $\supseteq$. Moreover, the function φ is monotonic:

Lemma 9

The function φ on D is monotonic, i. e. for each $d \preceq d'$, we have $\varphi(d) \preceq \varphi(d')$.

Proof: Follows immediately from Lemma 7 and the definition of φ and $\preceq$. $\square$

A fixed point of φ is an element $d \in D$ such that $\varphi(d) = d$. Note that $d = (P, Q)$ is a fixed point of φ if and only if $P = R(Q)$ and $Q = R(P)$, which are exactly those pairs of transition relations we are heading for. What is more, we can show that φ has fixed points.

Proposition 10

1. The function φ has fixed points; in particular, it has a least fixed point and it has a greatest fixed point (with respect to $\preceq$).
2. If (P, Q) is a fixed point of φ then also (Q, P) is a fixed point of φ.
3. In particular, (P, Q) is the least point of φ, iff (Q, P) is the greatest fixed point of φ.
4. There is a unique fixed point of φ, iff the least fixed point has the form (P, P).

Proof:

1. As mentioned above, $(D, \preceq)$ is a complete lattice and, by Lemma 9, we know that φ is a monotonic function on D (with respect to $\preceq$). By the renown Knaster-Tarski-Theorem, φ has a least and a greatest fixed point.
2. Let (P, Q) be a fixed point of φ. In combination with the definition of φ, we have $(P, Q) = \varphi((P, Q)) = (R(Q), R(P))$, i. e. $P = R(Q)$ and $Q = R(P)$. Thus, we have $\varphi((Q, P)) = (R(P), R(Q)) = (Q, P)$. So, (Q, P) is also a fixed point of φ.
3. Let (P, Q) be the least fixed point of φ and (P', Q') be the greatest fixed point of φ. By 2, (Q, P) and (Q', P') are also fixed points of φ. Because (P, Q) is the least fixed point and (P', Q') is the greatest fixed point, we have $(P, Q) \preceq (Q, P) \preceq (P', Q')$ and $(P, Q) \preceq (Q', P') \preceq (P', Q')$. By the definition of $\preceq$, we have $Q \subseteq P' \subseteq Q$ and $P \subseteq Q' \subseteq P$. This implies $P' = Q$ and $Q' = P$.
4. Let (P, P) be the least fixed point of φ. By 3, we know that (P, P) is also the greatest fixed point. So, (P, P) is the unique fixed point of φ.

   On the other hand, if we know that there is a unique fixed point (P, Q), we know that it is the least and the greatest fixed point. By 3, we know $(P, Q) = (Q, P)$, i. e. $P = Q$.

$\square$

Proposition 10 says, that φ has two distinguished fixed points, the least and the greatest fixed point. Fortunately, if we know the least fixed point (P, Q), we know the greatest fixed point too: the reversed pair (Q, P). In particular, we have $P \subseteq Q$. P is the transition relation with the least transitions in it, and Q is the transition with the most transitions in its. So we can use the least fixed point for defining the semantics of the EPC.

Definition 11 (Semantics of an EPC)

Let M be an EPC and let (P, Q) be the least fixed point of φ (wrt. $\preceq$).

Then, we call P the *pessimistic transition relation* of the EPC M, and we call Q the *optimistic transition relation* of the EPC M.

Actually, we have defined two semantics. The *pessimistic semantics* P is the one that stops rather than doing something 'awkward'; the *optimistic semantics* Q does something 'awkward' rather than stopping indeliberately. Both semantics are related such that $P = R(Q)$ and $Q = R(P)$ – so one is the input transition relation for the definition of the other.

When the pessimistic and the optimistic semantics coincide, we know that we have an ideal semantics $P = Q = R(P) = R(Q)$. These are the EPCs with a *clean* semantics – without any ambiguity. Unfortunately, there are EPCs for which the pessimistic and the optimistic semantics do not coincide. We call these the EPCs with an *unclean* semantics.

6 The ideal semantics

Now, we have defined two semantics for an EPC, where the pessimistic semantics is the one that was intended in [NR02]. As mentioned in the introduction of Sect. 5, we would like to have an ideal semantics for an EPC, i. e. a transition relation P with $P = R(P)$. Here, we will show that an ideal semantics does not exist for all EPCs. Figure 3 shows

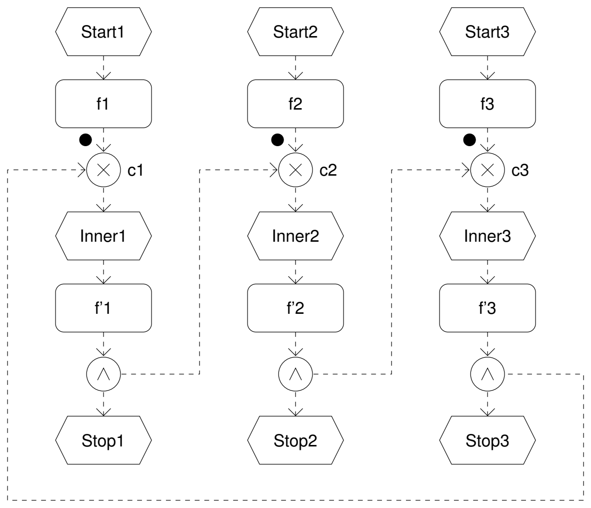

Figure 3: An EPC with no ideal semantics

an example. We argue indirectly that this EPCs has no ideal semantics: We assume that there is a transition relation P with $P = R(P)$ for this EPC. Now, we consider the state where there is a folder on each outgoing arc of the functions f_1, f_2, and f_3 as shown in Fig. 3. First, let us assume that according to P, connector c_i does not propagate the folder on its incoming arc. Then, according to the definition of $R(P)$, the subsequent connector $c_{(i+1)\, mod\, 3}$ will propagate the folder in $R(P)$. By $P = R(P)$, we know that, according to P, connector $c_{(i+1)\, mod\, 3}$ will propagate the folder. Second, let us assume that, according to P, c_i propagates the folder. By the same arguments, we can show that connector $c_{(i+1)\,mod\,3}$ will not propagate the folder according to P. Since we have an odd number of XOR-join connectors on the cycle, we can argue that if c_1 propagates the folder according to P, then it does not propagate it and vice versa – a contradiction. So, our assumption that there is an ideal semantics $P = R(P)$ must have been wrong.

For some other examples, there are ideal semantics. But, there may be different ideal semantics, which are symmetric such that one cannot be preferred to the other. For the example shown in Fig. 1 in Sect. 2, we have two completely symmetric ideal semantics[8]. In the first semantics, connector c_1 propagates the folder and connector c_2 does not propagate it. In the second semantics, connector c_2 propagates the folder and c_1 does not propagate it. Since both semantics are completely symmetric, one is as good as the other, there is no argument in favour of one of them.

These examples show that, in order to provide a semantics for all EPCs, we need to consider pairs of transition relations, as we did in our definition. Our definition gives a semantics to all (syntactically correct) EPCs – what is more, if the pessimistic and the optimistic semantics coincide, we have an ideal semantics. These are the EPCs for which the formal semantics precisely captures the informal semantics, the EPCs with a clean semantics. On the other hand, EPCs for which the the pessimistic and the optimistic semantics do not coincide, are problematic, because their formal semantics does not precisely capture the informal semantics. One benefit of our characterization is that we now have a clear definition of unclean EPCs.

7 The framework

In the previous sections, we presented a semantics for EPCs. Actually, it is not our intension to propose this semantics as 'the semantics' of EPCs. There are still some aspects of this semantics that need to be discussed. The main purpose of this paper is to define a framework for formalizing the semantics of EPCs, which allows us to discuss and to compare different semantics.

For defining a semantics for EPCs, it is enough to define the function $R(P)$. The semantics of each individual node n of an EPC could be changed by changing the definition of relation R_n in Def. 6. As long as the resulting function $R(P)$ is monotonously decreasing, the rest of the theory will define a pessimistic and an optimistic semantics in the very same way. Therefore, we can concentrate on the definition of $R(P)$ or even on R_n when discussing semantical issues.

The soundness of this framework is captured in the following theorem:

[8]The argument is the same as in the previous example. But, in this example we have an ideal semantics because the cycle consists of an even number.

Theorem 12 (Semantical Framework)
Let $M = (E, F, C, l, A)$ be an EPC.

1. Let $\mathcal{A}$ be some set. Then, we call $\sigma : A \to \mathcal{A}$ a *state* of M. The set Σ denotes the set of all states. A subset $P \subseteq \Sigma \times N \times \Sigma$ is called a *transition relation* of M with respect to $\mathcal{A}$.
2. Let $R : 2^{\Sigma \times N \times \Sigma} \to 2^{\Sigma \times N \times \Sigma}$ be a monotonously decreasing (with respect to $\subseteq$) function. We define the *domain* $D = 2^{\Sigma \times N \times \Sigma} \times 2^{\Sigma \times N \times \Sigma}$ and $\preceq$ on D as follows: For two elements $d = (P, Q)$ and $d' = (P', Q')$, we define $d \preceq d'$ if $P \subseteq P'$ and $Q \supseteq Q'$. Moreover, we define $\varphi : D \to D$ by $\varphi((P, Q)) = (R(Q), R(P))$.

Then φ has a least fixed point (P, Q) and a greatest fixed point (P, Q) (with respect to $\preceq$).

Proof: The proof follows exactly the lines of the proofs of Lemma 9 and Prop. 10. $\square$

In essence, for defining a semantics for EPCs, we first define some set $\mathcal{A}$, which represents the folders that are assigned to an arc of the EPC. In our example, $\mathcal{A}$ was the set $\{0, 1\}$; another reasonable choice would be the natural numbers[9], when we would like to have more than one folder on each arc.

Second, we need to define a transition relation $R(P)$ (on the states derived from the set $\mathcal{A}$), which in fact defines a function from a transition relation to a transition relation. The only requirement is that this function is monotonously decreasing. Then, $R(P)$ defines the function φ, which according to the above theorem has a least fixed point. The least fixed point (P, Q) of this function defines the the pessimistic semantics of the EPCs P and the optimistic semantics Q.

8 Conclusion

In this paper, we have proposed a semantics for EPCs, which is mathematically sound. We have argued that this semantics is as close to the informal semantics of [NR02] as can be. Nevertheless, we do not claim that this should be ‘the semantics’ of EPCs[10].

The main contribution of this paper is a sound mathematical theory for defining all kinds of non-local semantics for EPCs. A new semantics can be defined by giving a definition of $R(P)$; when $R(P)$ is monotonously decreasing, the semantics comes for free. For discussing different versions of semantics, we can concentrate on this definition. What is

[9]We did not chose natural numbers in our paper, because this introduces many choices in the definition of $R(P)$, which need careful investigation and is beyond the scope of this paper.
[10]The author is not yet convinced that EPCs should have a non-local semantics because of the ambiguities arising in some EPCs. In order to provide a sound foundation for a discussion of different semantics, however, he saw the need to provide a mathematical theory for defining non-local semantics in a uniform way.

more, for any semantics defined in this framework, the framework clearly identifies clean and unclean EPCs, where unclean EPCs are those that do not precisely meet the intended informal semantics. We hope that this framework helps to discuss different semantics and, ultimately, define 'the semantics' of EPCs.

Once 'the semantics' has been defined, the framework can be used for proving necessary and sufficient syntactical conditions characterizing unclean EPCs and to develop efficient algorithms for calculating the pessimistic and optimistic semantic for EPCs – whichever seems to be the more appropriate one. Of course, there are some obviously sufficient syntactical conditions for clean EPCs. For example, if no cycle of control flow of an EPCs has an XOR-join and OR-join connectors on it, the EPC has a clean semantics. But, stronger conditions are more difficult to find and strongly depend on the exact definition of $R(P)$. Likewise, fixed point approximation immediately gives us an algorithm for calculating the semantics of an EPC. Efficient algorithms, however, strongly depend on the exact definition of $R(P)$. Therefore, we need to define the 'the semantics' of EPCs first. Then we can start working on efficient simulation algorithms and efficient syntactical conditions for EPCs with a clean semantics.

Acknowledgements I would like to thank Markus Nüttgens and Frank Rump for many dicussions on the semantics of EPCs, which inspired me to provide a sound definition of a non-local semantics for EPCs. In particular, the arguments with Markus Nüttgens encouraged me not to stop with an impossibility result. Moreover, I would like to thank Wil van der Aalst and Björn Axenath for comments on earlier versions of this paper, which helped to improve the presentation of the ideas.

References

[KNS92] G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Heft 89, Universität des Saarlandes, January 1992.

[LA94] F. Leymann and W. Altenhuber. Managing business processes as an information resource. *IBM Systems Journal*, 33(2):326–348, 1994.

[LSW98] P. Langner, C. Schneider, and J. Wehler. Petri Net Based Certification of Event driven Process Chains. In J. Desel and M. Silva, editors, *Application and Theory of Petri Nets 1998, LNCS* 1420, 286–305. Springer, 1998.

[NR02] Markus Nüttgens and Frank J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In *PROMISE 2002, Prozessorientierte Methoden und Werkzeuge fürr die Entwicklung von Informationssystemen, GI Lecture Notes in Informatics* P-21, 64–77. Gesellschaft für Informatik, 2002.

[Rit00] Peter Rittgen. Quo vadis EPK in ARIS? *Wirtschaftsinformatik*, 42:27–35, 2000.

[Rum99] Frank J. Rump. *Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten*. Teubner-Reihe Wirtschaftsinformatik. B.G.Teubner, 1999.

[vdADK02] Wil van der Aalst, Jörg Desel, and Ekkart Kindler. On the semantics of EPCs: A vicious circle. In M. Nüttgens and F. J. Rump, editors, *EPK 2002, Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, 71–79, November 2002.

[vdAtH02] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. Technical Report QUT Technical report, FIT-TR-2002-06, Queensland University of Technology, Brisbane, 2002.