

OR Operator

From EPC Standard

The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.



Contents

[1 Short Description](#)

[2 Syntax](#)

[3 Semantics](#)

[3.1 Semantic Representation](#)

[4 XML Representation](#)

[5 References](#)

Short Description

The OR Operator is a subtype of the [operator](#) process element. On the one hand it can split the control flow in at least two different branches, due to a nonexclusive choice ([OR-Split](#)). On the other hand, it merges a split control flow, when the token(s) of the previously activated branch(es) arrive at it ([OR-Join](#)).

Syntax

Similar to the OR and [AND Operator](#), for the OR Operator exist two subtypes of operators: an [OR-Split](#) and a [OR-Join](#) operator. The split operator splits up the control flow in at least two branches, while the join operator reunites the split control flow. The OR-Split has exactly one ingoing and at least two or more outgoing arcs, while the OR-Join has multiple incoming arcs and one outgoing arc.[1]-[6] The EPC syntax requires an OR-Split to be triggered by a [function](#) and followed by multiple [events](#). Due to the semantics of events as passive elements, they lack the ability to determine the event(s) that should follow and therefore cannot be the predecessor of an OR-Split.[7]-[9] The OR-Join merges alternative branches, which were generated due to the split of the control flow arc by the OR-Split. The process of joining is not associated with a decision. Therefore it is possible for [events](#) to be the predecessor of an OR-Join.

Semantics

Operators express that a function can be started by one or more events, or a function can generate one

or more events as a result.

Semantic Representation

Generally, operators can be either split or join operators:

There are specific [OR-Split](#) and [OR-Join](#) operators. Following definitions exist:

- $Cor = \{c \in C \mid l(c) = or\}$ as the set of OR-connectors. [10]
- $Cos = \{c \in C \mid l(c) = or \wedge |nin| = 1\}$ as the set of OR-Split operators.[11]

The OR-Split represents a choice between one or more of several alternative branches within the process. It waits to get the [control flow](#) on its incoming arc before allowing it to continue on one or more of its outgoing arcs.[12]

- $Coj = \{c \in C \mid l(c) = or \wedge |nout| = 1\}$ as the set of OR-join operators.[11]

An [OR-Join](#) waits for an arriving token on each of its incoming arcs, propagates it to the outgoing arc and activates its successor. In contrast to a [XOR-Join](#) the OR-Join must wait until all tokens have arrived. The decision whether more tokens can arrive on the incoming arcs cannot be made locally at the OR-Join, it depends on the behavior of the corresponding [OR-Split](#). E.g. when the split has sent three tokens on different outgoing arcs, the join has to wait for exactly these three tokens to arrive in order to continue the process.[13] . Therefore, the semantics of the OR-Join connector is called non-local.[1]

The non-local semantic is a central characteristic of the OR Operator and leads to a tradeoff.[14] On the one hand the non-local semantic helps to simplify many models, but on the other hand it results in a lack of satisfactory formalization.[15] The non-local semantic is comprehensively discussed in the literature and there are different suggestions and approaches to overcome this problem.[15][16][17]

XML Representation

[Edit the XML Code](#)

```
<source lang="xml">
```

```
<xs:complexType name="typeOR"> <xs:sequence> <xs:element name="documentation" type="xs:anyType" minOccurs="0"/> <xs:element name="toolInfo" type="xs:anyType" minOccurs="0"/> <xs:element name="name" type="xs:string" minOccurs="0"/> <xs:element name="description" type="xs:string" minOccurs="0"/> <xs:choice minOccurs="0"> <xs:element name="graphics" type="epml:typeGraphics"/> </xs:choice> <xs:choice minOccurs="0"> <xs:element name="syntaxInfo"> <xs:complexType> <xs:attribute name="implicitType"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="orFunctionEventSplit"/> <xs:enumeration value="orFunctionEventJoin"/> <xs:enumeration value="orEventFunctionJoin"/> </xs:restriction> </xs:simpleType> </xs:attribute> </xs:complexType> </xs:element> </xs:choice> <xs:choice minOccurs="0" maxOccurs="unbounded"> <xs:element name="attribute" type="epml:typeAttribute"/> </xs:choice> </xs:sequence> <xs:attribute name="id" type="xs:positiveInteger" use="required"/> <xs:attribute name="defRef" type="xs:positiveInteger" use="optional"/> </xs:complexType> </source>
```

References

- [*1] N. Cuntz and E. Kindler, "On the Semantics of EPCs: Efficient Calculation and Simulation," *Bus. Process Manag.*, pp. 398–403, 2005.
- [*2] B. F. van Dongen, R. M. Dijkman, and J. Mendling, "Measuring Similarity between Business Process Models," *Adv. Inf. Syst. Eng.*, vol. 5074, pp. 450–464, 2008.
- [*3] J. Dehnert, J. Dehnert, P. Rittgen, and P. Rittgen, "Relaxed soundness of business processes," *Lect. notes Comput. Sci.*, pp. 157–170, 2001.
- [*4] E. Kindler, "On the semantics of EPCs: Resolving the vicious circle," *Data Knowl. Eng.*, vol. 56, no. 1, pp. 23–40, 2006.
- [*5] M. La Rosa, M. Dumas, A. H. M. ter Hofstede, and J. Mendling, *Configurable multi-perspective business process models*, vol. 36, no. 2. 2011.
- [*6] J. Mendling, M. La Rosa, and a H. M. Hofstede, "(2008) Soundness of EPC Process Models with Objects and Roles . Copyright 2008 (The authors) Soundness of EPC Process Models with Objects and Roles," vol. 2008, 2008.
- [*7] G. Keller, M. Nüttgens, and A.-W. Scheer, "Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“,“ *Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi)*, Univ. des Saarlandes, no. 89, 1992.
- [*8] M. Fellmann, S. Bittmann, A. Karhof, C. Stolze, and O. Thomas, "Do We Need a Standard for EPC Modelling? The State of Syntactic, Semantic and Pragmatic Quality," in *5th International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA)*, 2013, pp. 103–116.
- [*9] J. Mendling, H. M. W. Verbeek, B. F. van Dongen, W. M. P. van der Aalst, and G. Neumann, "Detection and prediction of errors in EPCs of the SAP reference model," *Data Knowl. Eng.*, vol. 64, no. 1, pp. 312–329, 2008.
- [*10] Mendling: *Event Driven Process Chains - Metrics for Process Models*, Volume 6 of the series *Lecture Notes in Business Information Processing*, 2009, pp. 17-57.
- [*11] E. Kindler "On the semantics of EPCs: resolving the vicious circle", *Data & Knowledge Engineering - Special issue: Business process management archive Volume 56 Issue 1*, 2006, pp.23-40.
- [*12] R. Dijkman, "Diagnosing differences between business process models," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5240 LNCS, pp. 261–277, 2008.
- [*13] V. Gruhn and R. Laue, "What business process modelers can learn from programmers," *Sci. Comput. Program.*, vol. 65, no. 1, pp. 4–13, 2007.
- [*14] N. Cuntz, J. Freiheit, and E. Kindler, "On the Semantics of EPCs: Faster Calculation for EPCs with Small State Spaces," *Proc. Fourth Work. Event-Driven Process Chain. (WI-EPK 2005)*, pp. 7–23, 2005.
- [*15] E. Kindler, "On the semantics of EPCs: A vicious circle," no. August, pp. 71–80, 2002.
- [*16] P. Rittgen, "Quo vadis EPK in ARIS? Ansätze zu syntaktischen Erweiterungen und einer formalen Semantik," *Wirtschaftsinformatik*, vol. 42, no. 1, pp. 27–35, 2000.
- [*17] P. Langner, C. Schneider, and J. Wehler, "Petri Net Based Certification of Event-Driven Process Chains," *19h Int. Conf. Appl. Theory Petri Nets*, pp. 286–305, 1998.

[Categories:](#)

[Meta Model](#)

This page was last edited on 15 January 2021, at 13:46.
Content is available under [Creative Commons „Zero“ \(Gemeinfreiheit\)](#) unless otherwise noted.

[Privacy policy](#)

[About EPC Standard](#)

[Disclaimers](#)

[Powered by MediaWiki](#)