XOR Operator

From EPC Standard

XOR Operator

Contents

Short Description

The XOR 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 an exclusive choice (XOR-Split). On the other hand, it merges a split control flow, when the token of the previously activated branch arrives at it (XOR-Join).

Syntax

Similar to the OR and AND Operator, For the XOR Operator exist two subtypes of operators: a XOR-Split and a XOR-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 XOR-Split has exactly one ingoing and at least two or more outgoing arcs, while the XOR-Join has multiple incoming arcs and one outgoing arc.[1][2][3][4][5][6] The EPC syntax requires an XOR-Split to be triggered by a function and followed by events. Due to the semantics of events as passive elements, they lack the ability to determine the event that should follow and therefore cannot be the predecessor of a XOR-Spilt.[7][8][9] The XOR-Join merges alternative branches, which were generated due to the split of the control flow arc by the XOR-Split.[9] The process of joining is not associated with a decision. Therefore, it is possible for events to be the predecessor of a XOR-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 XOR-Split and XOR-Join operators. Following definitions exist:

- Cxor = {c ∈ C | l(c) = xor} being the set of XOR-connectors.[9]
- Cxs = {c ∈ C | l(c) = xor ∧ |nin| = 1} as the set of OR-Split operators.[10]

The XOR-Split represents an exclusive choice between one of several alternative branches within the process. As a result, an XOR-Split triggers exactly one of several possible following events. Which of the possible branches is activated depends on the process conditions.[11][12][13]

- Cxj = {c ∈ C | l(c) = xor ∧ |nout| = 1} as the set of XOR-Join operators. [10]

An XOR-Join waits for an arriving token on one of its incoming arcs, which is then propagated to the outgoing arc and activates the successor. But for an XOR-Join there is an additional condition: The XOR-Join must not activate its successor, if there is or there could arrive a token on one of the other incoming arcs. Whether it is possible that a token arrive on one of the other arcs or not depends on the overall behaviour of the EPC and cannot be checked locally. Therefore, the semantics of the XOR-Join connector is called non-local. [1]

The non-local semantic is a central characteristic of a XOR Operator add 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="typeXOR"> <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="xorFunctionEventSplit"/> <xs:enumeration value="xorFunctionEventJoin"/> <xs:enumeration value="xorEventFunctionJoin"/> </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] Mendling: Event Driven Process Chains - Metrics for Process Models, Volume 6 of the series Lecture Notes in Business Information Processing, 2009, pp. 17-57.
- [*10] 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.
- [*11] 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.
- [*12] R. Laue and J. Mendling, “Structuredness and its significance for correctness of process models,” Inf. Syst. Ebus. Manag., vol. 8, no. 3, pp. 287–307, Jun. 2009.
- [*13] V. Gruhn and R. Laue, “Forderungen an hierarchische EPK-Schemata,” EPK 2007 Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, no. November, pp. 59–76, 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