

Timed automaton RVT-grammar for workflow translating

Alexander Afanasyev, Nikolay Voit^[0000-0002-4363-4420], Sergey Kirillov

Ulyanovsk State Technical University, Ulyanovsk, Russia
{a.afanasev,n.voit}@ulstu.ru, kirillovsyu@gmail.com

Abstract. The paper studies grammar for workflow translating including semantic analysis. The main purpose of the translation is to expand the methods of semantic analysis of the grammatical model of distributed workflows due to the capabilities of the translation language. The article describes grammar, algorithm of its construction, differences from usual RV-grammar and author's modifications. At the end of the work the result of the experiment of translating the BPMN language diagrams into a temporary Petri net is presented.

Keywords: Automated Systems, Diagrammatic Models, Workflows, Automaton Grammars, Timed petri nets, Diagram Translation

1 Introduction

Large manufacturing enterprises have complex business processes. Management of business processes while providing customers with services and products has become key for such enterprises [9]. As a rule, the analysis of a large amount of information is necessary for decision-making by the manager. It uses data mining, data warehousing, on-line analytical processing to obtain unbiased useful information [5]. In most cases, the presentation format of business processes (for example, BPMN, IDEF3, eEPC) should be translated into a modeling format that is understandable for modeling tools [11]. When analyzing business processes for errors, designers are forced to translate the internal representation of the system work flows into a view that is suitable for modeling [8]. Diagrammatic model of visual modeling languages (e.g. UML [1], IDEF [2], eEPC [3], BPMN [4], SharePoint, ER, DFD) are widely used in the practice of designing complex automated systems (as) especially at the conceptual stage [5]. Such languages are flexible and allow you to build diagrams that can be applied to different subject areas. The flexibility of languages is due to the incompleteness or informality of their description, as a result of which the resulting diagrams can be interpreted ambiguously. Machine processing of such graphic diagrams is difficult [11]. Language flexibility can lead to a family of languages, i.e. many languages that conceptually have a common basis but different interpretation specific to the subject area of their application. Most of the existing approaches consider such languages in isolation, although it is sufficient to determine the generalizing semantics for the language family (perhaps

for some elements it will be abstract) and to specialize the semantic component of the individual elements of the language and (or) the diagram before its interpretation.

The paper has the following structure. Section 2 has related work that short describes major relevant research studies. Section 3 provides a brief overview of the semantics of graphical languages. Section 4 contains RVTITg-grammar. Section 5 presents the example RVTITg-grammar for basic BPMN. Conclusions and further research directions are presented in Conclusion.

2 Related Work

The authors investigate some works that consider the specification of document flow, verification and translation. Several papers focused on the definition of formal semantics and validation methods for workflows using Petri nets, process algebra, abstract state machine, see for example [12-22]. In [18, 19], Decker and Weske propose a Petri net-based formalism for determining choreographies, properties as realizability and local applicability, and a method for verifying these two properties. However, they consider only synchronous communication and does not explore the association with languages modeling of interaction of a high-level BPMN. Bultan and Fu [23] determine a sufficient condition for analyzing the feasibility of choreographies defined using UML collaboration diagrams (CD). In [24], Salaün and Bultan modify and extend this work with the feasibility analysis method by adding a synchronization message among peers. This method controls the realizability of CDs for bounded asynchronous communication. The feasibility problem for Message sequence diagrams (MSCs) has also been studied (e.g. [25, 26]). In [26], the authors offer bounded MSCS graphs which are bounded by BPMN 2.0 because branching and looping behavior are not supported by CDs and MSCs (there is no selection in CDs, there are no some looping behaviors in MSCs, and only Self-loops in CDs). In [27] BPMN behavior is studied from the semantic point of view and several BPMN patterns are proposed. This work is not theoretically justified and is not complete, it discusses only some of the laws. Lohmann and Wolf [28] propose to analyze existing patterns and control them with compatible patterns. In [29], the authors focused on the translation of BPMN into the algebra of processes for the analysis of choreography using model checking and equivalence. The main limitation of these methods is that they do not work when there are different types of diagrams at the same time, which means that in some cases the input diagrams cannot be analyzed.

3 The semantics of graphical languages

All existing graphic languages can be divided into the following types according to the language formality:

1. Formal. The syntax and semantics of such languages are formally defined.
2. Semi-formal. The syntax of the language is formal, and semantics can have different interpretations.

3. Informal. The syntax and semantics of the language are informal [6].

The vast majority of popular graphic languages are semi-formal. For them, it is worth investigating the methods of formalization. In [5, 8, 9, 11, 12-29] mainly offer to give a semantic technicality language in the following ways:

1. To specialize the language. To give it some capabilities of the language to simplify it, to give new opportunities that will positively affect formalization.
2. To determine the semantics dynamically. Dynamic semantics involves the transformation of the diagrams of the basic graphic language into a target language.

The second method is more promising, because it makes it possible to implement diagrams of different graphic languages on the basis of one universal tool. The method develops the ideas of primitive libraries. In this case, libraries store the interpretation of the graphical image in terms of the target graphic or text language. And there can be several interpretations for the same image, each of them is assigned a unique name to avoid ambiguous interpretation and incorrect use. The target language is a more formal language relative to the base language.

4 RVTITg-grammar

RVTITg-grammar (Timed RV-grammar for graphic translation) is an RVTI-grammar (Timed RV-grammar [7]) development in which the grammar scheme products are expanded to store the correspondence in terms of the target formal description, and the internal memory stores the information necessary for the translation process [7, 10]. Temporal RVT-grammar of a language L (G) is an ordered n-tuple of eleventh non-empty sets

$$G = (V, U, \Sigma, \tilde{\Sigma}, M, F, C, E, R, T, r_0) \quad (1)$$

where $V = \{v_e, e = \overline{1, L}\}$ is auxiliary alphabet; $U = \{u_e, e = \overline{1, K}\}$ is auxiliary alphabet of the target language; $\Sigma = \{a_t, t = \overline{1, T}\}$ is terminal alphabet graphic language; $\tilde{\Sigma} = \{\tilde{a}_t, t = \overline{1, T}\}$ is quasi terminal alphabet; $M = TT \cup TN$ is combining terminal (TT) and non-terminal (TN) characters of the target language; $F = \{generate_input(), generate_output(), select_output(), stick_connection_points()\}$ is many translation functions for working with elements of the set; C is set of clock identifiers; E is the set of temporal relations “Before”, “During”, “After” (initialization of the clock $\{c := 0\}$, relations of the form $\{c \sim x\}$, where x the variable (the identifier of the clock), c is a constant, $\sim \in \{=, <, \leq, >, \geq\}$); $R = \{r_i, i = \overline{0, I}\}$ is grammar G schema (set of names of complexes of products, each complex r_i consists of a subset P_{ij} of products $r_i = \{P_{ij}, j = \overline{1, J}\}$); $T \in \{t_1, t_2, t_3, \dots, t_n\}$ is a set of time stamps; $r_0 \in R$ is RV-axiom grammar.

Graphical objects containing more than one input or output are loaded with translation functions from the set to ensure that the inputs and outputs of such objects match the base and target languages. Assignment of translation functions and operations performed by them.

1. `generate_input()` is a formation of a set of input connection points, except for the one on which this graphic object was reached. It is performed in the primary analysis of graphical objects containing more than one input.
2. `generate_output()` is a formation of a set of outgoing connection points. It is performed in the primary analysis of graphical objects containing more than one output, or when the only output is supposed to be used as a link – label, i.e. it is necessary to change the direction of analysis.
3. `select_output()` is a select the outgoing connection point of the element as the continuer of the target language chain. An event function that is executed for graphical circuits with a dynamically changing number of outgoing connection points after the formation of a set of connection points for outgoing connections, it is selected from such points. In General, the selection algorithm is not regulated, i.e. the choice is random.
4. `stick_connection_points()` is a link connection points and object. An event function that is performed when secondary graphical objects that contain more than one input are analyzed. Binds the incoming link to the connection point of the object, information about which is stored in the internal memory.

The presence of these functions allows you to create an algorithm for building the output chain, the main operations in which are the selection of the point – continuer analysis for objects containing more than one output, and the layout of a complete sequence of already analyzed objects containing more than one input, and associated with them analyzed objects. The order of application of functions is shown in Fig. 1.

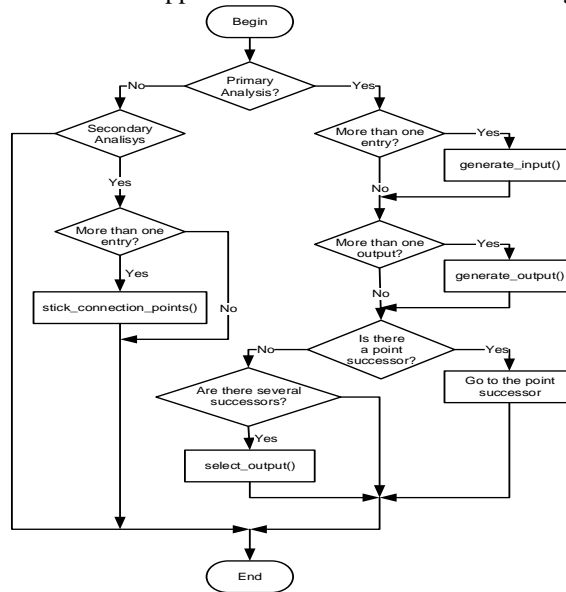


Fig. 1. The main algorithm of translation.

For Fig. 2 all possible display options are presented. Table 1 describes the corresponding types of the choice of the successor and the application of f function. On the maps

(see Fig. 2) hollow points show the guide points, shaded – the points through which the analyzer reaches the graphical object, and filled with points – all other incoming and outgoing points. In the following tables, the encoding of translation functions is accepted. The value f_{gi} in the table cells corresponds to the function call `generate_input()`, f_{go} is a `generate_output()`, f_{so} is a `select_output()`, f_{scp} is a `stick_connection_point()`.

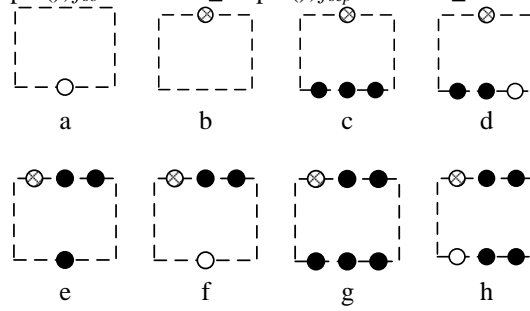


Fig. 2. Possible display types.

Table 1. A description of the possible representation types.


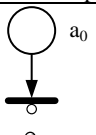

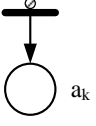

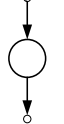
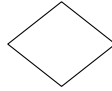
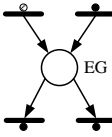





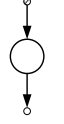
Representation description	Translating functions	Successors
with one directional output (Fig. 2, a)	no	directed output
With one input (Fig. 2, b)	no	no, selected from the number of label links
with one input and several non-directional outputs (Fig. 2, c)	f_{go} is a selects all outputs	f_{so} or selected from the number of label links
with one input and several outputs, including the directional (Fig. 2, d)	f_{go} is a selects all outputs except directional	directed output
with multiple inputs and one non-directional output (Fig. 2, e)	f_{gi}, f_{go} are selects all outputs	no, selected from the number of label links
with multiple inputs and one directional output (Fig. 2, f)	f_{gi}	directed output
with multiple inputs and multiple non-directional outputs (Fig. 2, g)	f_{gi}, f_{go} are selects all outputs	f_{so} or selected from the number of label links
with multiple inputs and multiple outputs, including directional (Fig. 2, h)	f_{gi}, f_{go} are selects all outputs except directional	directed output

5 Example RVTITg-grammar for basic BPMN

The base language from which the broadcast will be produced is a well-known BPMN. A standard Business Process Model and Notation (BPMN) will provide businesses with

the capability of understanding their internal business procedures in a graphical notation and will give organizations the ability to communicate these procedures in a standard manner [4]. The BPMN specification also provides a mapping between the graphics of the notation and the underlying constructs of execution languages, particularly Business Process Execution Language (BPEL) [4]. As a translation language, a timed Petri net was chosen. Representation in terms of the timed Petri network for the language elements of the BPMN presented in Table 2.

Table 2. Representation in terms of the timed Petri network for the language elements of the BPMN.

	Element name	BPMN representation	Timed Petri nets representation
A	Start event		
B	End event		
C	Action		
D	Exclusive gateway		
E	Parallel gateway		
F	Intermediate event "Timer"		
H	Link flow		

Final tabular form RVTg-grammar for the languages BPMN and timed Petri nets are presented in Table 3.

Table 3. Translation grammar example.

N	State	Quasi term	Next state	Operation with memory	
				Base language	Target language
1	r_0	A_0	r_1	\emptyset	\emptyset
2	r_1	rel	r_3	\emptyset	\emptyset
3	r_2	$label_{EG}$	r_3	$W_2(b^{1m}, b^{t(6)})$	$W_2(b^{3m})$
4		$label_{PG}$	r_3	$W_2(b^{2m}, b^{t(6)})$	$W_2(b^{4m})$
5	r_3	A_i	r_1	\emptyset	\emptyset
6		A_{im}	r_1	\emptyset	\emptyset
7		A_{it}	r_1	$W_1(t_s^{t(6)})$	\emptyset
8		A_{kl}	r_2	$\emptyset / W_3(!e^{1m}, !e^{2m})$	$\emptyset / W_3(!e^{3m}, !e^{4m})$
9		A_k	r_4	\emptyset	\emptyset
10		A	r_1	$W_1(t_s^{t(6)})$	\emptyset
11		A_{it}	r_3	$W_1(t_s^{t(6)})$	\emptyset
12		EG_c	r_1	$W_1(t^{1m^{(n-1)}}) / W_3(k = 1)$	$W_1(t^{3m^{(n-1)}}) / W_3(k = 1)$
13		EG	r_2	$W_1(I^{t(1)}, k^{t(2)}) / W_3(e^{t(2)}, k \neq 1)$	$W_1(I^{t(7)}, k^{t(8)}) / W_3(e^{t(8)}, k \neq 1)$
14		$_EG$	r_2	$W_1(inc(m^{t(1)}) / W_3(m^{t(1)} < k^{t(2)}))$	$W_1(inc(m^{t(7)}) / W_3(m^{t(7)} < k^{t(8)}))$
15		$_EG_e$	r_1	$W_1(t^{1m^{(n-1)}}) / W_3(m^{t(1)} = k^{t(2)}, p \neq 1)$	$W_1(t^{3m^{(n-1)}}) / W_3(m^{t(7)} = k^{t(8)}, p \neq 1)$
16		$_EG_{me}$	r_1	$o / W_3(m^{t(1)} = k^{t(2)}, p = 1)$	$o / W_3(m^{t(7)} = k^{t(8)}, p = 1)$
17		PG_f	r_1	$W_1(t^{2m^{(n-1)}}) / W_3(k = 1)$	$W_1(t^{4m^{(n-1)}}) / W_3(k = 1)$
18		PG	r_2	$W_1(I^{t(3)}, k^{t(4)}) / W_3(e^{t(3)}, k \neq 1)$	$W_1(I^{t(9)}, k^{t(10)}) / W_3(e^{t(9)}, k \neq 1)$
19		$_PG$	r_2	$W_1(inc(m^{t(3)}) / W_3(m^{t(3)} < k^{t(4)}))$	$W_1(inc(m^{t(9)}) / W_3(m^{t(9)} < k^{t(10)}))$
20		$_PG_e$	r_1	$W_1(t^{2m^{(n-1)}}) / W_3(m^{t(3)} = k^{t(4)}, p \neq 1)$	$W_1(t^{4m^{(n-1)}}) / W_3(m^{t(9)} = k^{t(10)}, p \neq 1)$
21		$_PG_{je}$	r_1	$W_1(t^{2m^{(n-1)}}) / W_3(m^{t(3)} = k^{t(4)}, p = 1)$	$W_1(t^{4m^{(n-1)}}) / W_3(m^{t(9)} = k^{t(10)}, p = 1)$
22	r_4	no_label	r_5	*	*
23	r_5				

When the analysis is complete, the stores in the internal memory of the target language must be empty and all tape cells must contain the “0” character. Checking the memory status is described by the operation indicated by the “/” symbol. Take an example of an abstract diagram in BPMN. It is depicted in the Fig. 3.

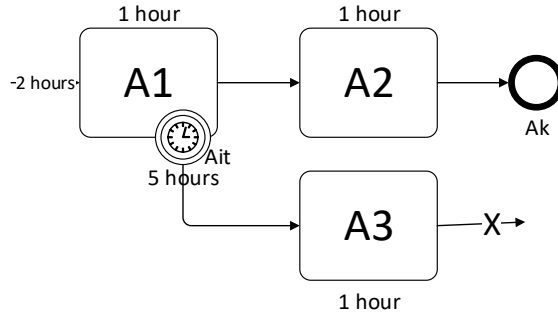


Fig. 3. Temporal BPMN diagram example.

Representation of temporal BPMN diagram example in timed Petri nets presented in Fig. 4.

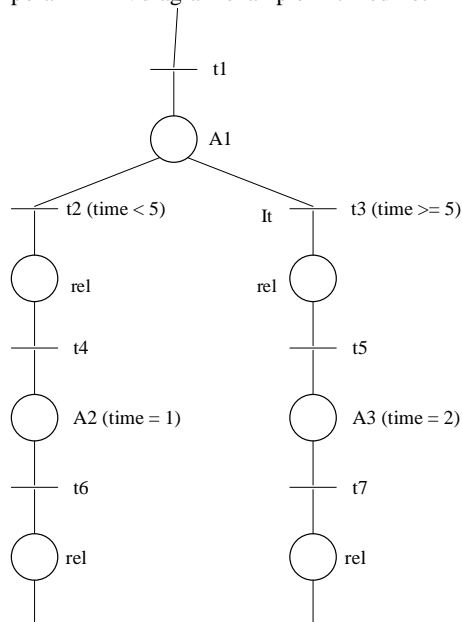


Fig. 4. Representation of temporal BPMN diagram example in timed Petri nets.

Conclusions

Presented RVTITg-grammar based on RVTI-grammar, which takes into account the temporal characteristics and broadcast diagrammatically models in different graphical languages. An example of the translation of BPMN diagrams into a timed Petri net. Further directions of work are the expansion of the possibilities of semantic analysis of diagrammatic models from the point of view of coordinating text attributes of diagrams with project documentation.

Acknowledgment

The reported study was funded by RFBR according to the research project № 17-07-01417 and Russian Foundation for Basic Research and the government of the region of the Russian Federation, grant № 18-47-730032.

References

1. Booch, G., Jacobson, I., Rumbaugh, J.: The Unified Modeling Language User Guide. Addison-Wesley (1998).
2. Mayer, R. J., Painter, M. K., de Witte, P. S.: IDEF family of methods for concurrent engineering and business re-engineering applications. College Station, Tex, USA: Knowledge Based Systems (1994).
3. Santos, P. S., Almeida, J. P. A., Pianissolla, T. L.: Uncovering the organisational modelling and business process modelling languages in the ARIS method. *International Journal of Business Process Integration and Management*, 5(2), 130-143 (2011).
4. Model, B. P. Notation (BPMN), v. 2.0, 2011. OMG, www.omg.org/spec/BPMN/2.0, last accessed 2018/09/01.
5. Van Der Aalst, W., Van Hee, K. M., van Hee, K.: *Workflow management: models, methods, and systems*. MIT press (2004).
6. Sharov, O., Afanasyev, A.: Syntax error recovery in graphical languages. *Programming and Computer Software* 34, 44-48 (2008). doi: 10.1134/S0361768808010052
7. Afanasyev, A. N., Voit, N. N., Kirillov, S. Y.: Development of RYT-grammar for analysis and control dynamic workflows. *International Conference on Computing Networking and Informatics (ICCNI)*, pp. 1-4. Lagos (2017). doi: 10.1109/ICCNI.2017.8123797
8. Zur Muehlen, M.: *Workflow-based process controlling: foundation, design, and application of workflow-driven process information systems*. vol. 6. Michael zur Muehlen (2004).
9. Becker, J., Rosemann, M., Von Uthmann, C.: *Guidelines of business process modeling*. *Business Process Management*. Springer, Berlin, Heidelberg, pp. 30-49 (2000).
10. Maurer, P.M.: The design and implementation of a grammar-based data generator. *Software: Practice and Experience*, vol. 22, no. 3, pp. 223–244 (1992).
11. Reijers, H. A.: *Design and control of workflow processes: business process management for the service industry*. Springer-Verlag (2003).
12. Poizat, P., Salaün, G., Krishna, A.: Checking business process evolution. In *International Workshop on Formal Aspects of Component Software*, pp. 36-53. Springer, Cham (2016), <https://hal.inria.fr/hal-01366641>, last accessed 2018/09/01.
13. Martens, A.: Analyzing web service based business processes. In *International Conference on Fundamental Approaches to Software Engineering*, pp. 19-33. Springer, Berlin, Heidelberg (2005). doi: 10.1007/978-3-540-31984-9_3
14. Raedts, I., Petkovic, M., Usenko, Y. S., van der Werf, J. M. E., Groote, J. F., Somers, L. J.: Transformation of BPMN Models for Behaviour Analysis. *MSVVEIS*, 2007, 126-137 (2007).
15. Dijkman, R. M., Dumas, M., & Ouyang, C.: Semantics and analysis of business process models in BPMN. *Information and Software technology*, 50(12), 1281-1294 (2008). doi: 10.1016/j.infsof.2008.02.006

16. Wong, P. Y., & Gibbons, J.: A process semantics for BPMN. In International Conference on Formal Engineering Methods, pp. 355-374. Springer, Berlin, Heidelberg (2008). doi: 10.1007/978-3-540-88194-0_22
17. Wong, P. Y., & Gibbons, J.: Verifying business process compatibility (short paper). In Quality Software, 2008. QSIC'08. The Eighth International Conference on, pp. 126-131. IEEE (2008, August). doi: 10.1109/QSIC.2008.6
18. Decker, G., & Weske, M.: Interaction-centric modeling of process choreographies. Information Systems, 36(2), 292-312 (2011). doi: 10.1016/j.is.2010.06.005
19. Decker, G., & Weske, M.: Local enforceability in interaction petri nets. In International Conference on Business Process Management, pp. 305-319. Springer, Berlin, Heidelberg (2007, September). doi: 10.1007/978-3-540-75183-0_22
20. Güdemann, M., Poizat, P., Salaün, G., & Dumont, A.: Verchor: A framework for verifying choreographies. In International Conference on Fundamental Approaches to Software Engineering, pp. 226-230. Springer, Berlin, Heidelberg (2013, March). doi: 10.1007/978-3-642-37057-1_16
21. Mateescu, R., Salaün, G., & Ye, L.: Quantifying the parallelism in BPMN processes using model checking. In Proceedings of the 17th international ACM Sigsoft symposium on Component-based software engineering, pp. 159-168. ACM (2014, June). doi: 10.1145/2602458.2602473
22. Kossak, F., Illibauer, C., Geist, V., Kubovy, J., Natschläger, C., Ziebermayr, T., ... & Schewe, K. D.: A Rigorous Semantics for BPMN 2.0 Process Diagrams. In A Rigorous Semantics for BPMN 2.0 Process Diagrams, pp. 29-152. Springer, Cham (2014). doi: 10.1007/978-3-319-09931-6_4
23. Bultan, T., & Fu, X.: Specification of realizable service conversations using collaboration diagrams. Service Oriented Computing and Applications, 2(1), 27-39 (2008). doi: 10.1109/SOCA.2007.41
24. Salaün, G., & Bultan, T.: Realizability of choreographies using process algebra encodings. In International Conference on Integrated Formal Methods, pp. 167-182. Springer, Berlin, Heidelberg (2009, February).
25. VBPMN Framework, <https://pascalpoizat.github.io/vbpmn/>, last accessed 2018/09/01.
26. Alur, R., Etesami, K., & Yannakakis, M.: Realizability and verification of MSC graphs. Theoretical Computer Science: Automata, Languages and Programming, 331(1), 97 (2005). doi: 10.1016/j.tcs.2004.09.034
27. Lotos, I. S. O.: A formal description technique based on the temporal ordering of observational behaviour. ISO8807, 1XS989 (1989).
28. Lohmann, N., & Wolf, K.: Realizability is controllability. In International Workshop on Web Services and Formal Methods, pp. 110-127. Springer, Berlin, Heidelberg (2009, September). doi: 10.1007/978-3-642-14458-5_7
29. Poizat, P., & Salaün, G.: Checking the realizability of BPMN 2.0 choreographies. In Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 1927-1934. ACM (2012, March). doi: 10.1145/2245276.2232095