Graph-based Process Discovery containing Invisible Non-Prime Task in Procurement of Animal-Based Ingredient of Halal Restaurants

Kelly Rossa Sungkono
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya Indonesia
kelly@its.ac.id

Adhatus Solichah Ahmadiyah
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya Indonesia
adhatus@if.its.ac.id

Riyanarto Sarno
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya Indonesia
riyanarto@if.its.ac.id

Muhammad Farhan Haykal
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya Indonesia
farhan.ikul@gmail.com

Muhammad Rayhan Hakim
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya Indonesia
rayhan.hakim17@gmail.com

Bagas Juwono Priambodo
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya Indonesia
jwnbagas@gmail.com

Muhammad Amir Fauzan
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya Indonesia
amrfauzan.af@gmail.com

Muhammad Kiantaqwa Farhan
Department of Informatics
Institut Teknologi Sepuluh Nopember
Surabaya Indonesia
enzozizou28@yahoo.com

***Abstract*— A process model based on animal-based ingredients' running procurement will help halal level examiners check halal implementation based on Halal Critical Control Points (HCCP). Process discovery is a study for automatically forming a process model based on a log of running processes. There are several algorithms of process discovery, such as Alpha++ and Alpha#. However, the procurement of animal-based ingredient processes has an invisible non-prime task that has not been discussed in the existing algorithms. This research proposes a graph-based process discovery algorithm to form a process model containing invisible non-prime tasks based on the procurement processes. The log of procurement processes is obtained by using a business process management application, i.e., ProcessMaker. This research evaluates the proposed graph-based algorithm by comparing it with Alpha++ and Alpha# based on fitness and precision. The evaluation verifies that the proposed graph-based algorithm has a better quality of the obtained process model than Alpha++ and Alpha#. The fitness and precision of the graph-based algorithm are 1 and 1. On the other hand, the precisions of Alpha++ and Alpha# are 0.43 and 0.43, respectively.**

Keywords—Halal, Process Discovery, Animal-based, Invisible Non-Prime Task

I. INTRODUCTION

The process discovery of animal-based ingredients' running procurement is an essential task to be considered. The goal is that we must always be aware of the foods we eat, especially those made of animal origin. Compared to conventional food management systems, halal food operations have special requirements, including pork-free ingredients, no alcohol, and meat must be slaughtered in an Islamic way, among others [1].

To provided checking processes by the process discovery, the process model of checking animals' halalness must be formed. The processes are based on Halal Critical Control Points (HCCP) [2]. HCCP are rules where haram substances are mentioned and removed from the processes [3]. This study proposed the process models divided into ten sections: meat, fat, milk, offal, blood, hair, eggs, trachea, bones.

Based on those proposed process models, the process model of offal has invisible non-prime tasks. The invisible non-prime tasks occur in this offal process model. Invisible non-prime task occurs because checking the internal digestive enzyme content can simultaneously examine the intestinal organs in the external digestive [4]. The existing process discovery algorithms, i.e., Alpha ++ [5], [6] and Alpha # [7], cannot handle this issue.

Besides proposed process models, this research develops the proposed graph-based process discovery using Neo4j with Cypher language query for handling invisible non-prime tasks. This study will also compare quality measurements of process models, i.e., fitness and precision, using Alpha ++ and Alpha#. This study can be a reference for halal examiners to check their procurement in total [8].

II. LITERATURE REVIEW

A. Existing Process Discovery Algorithms

The analysis uses a business process model to determine the latest business processes as a reference for future

development. Modification of the business process model is often overlooked in systems development, making process model changes incompatible with system changes. A set of algorithms called process discovery is required for business process models to detect automation algorithms [9]. The discovery process is also arguably one of the most challenging mining processes, which generates a process model from event log data [10].

Process discovery can be defined as a technique for defining, mapping, and analyzing organizational processes. The process discovery tool is a machine learning-based tool intended to help organizations identify business processes, record all possible uses of machine learning algorithms, and make Automation recommendations. Process Discovery tools distinguish the business processes that can be automated and help automate workflows, map, plan, and implement automation more effectively and efficiently. In this case, we use tools like Neo4j and ProM [11].

The process of examining animal halalness is divided into ten parts: meat, fat, milk, offal, blood, hair, eggs, trachea, bones. We use three algorithmic methods in this process. The first one uses the Alpha ++ algorithm. Second, using the Alpha # algorithm. Moreover, the latter uses the proposed graph-based process discovery method. All three algorithms are compares based on the quality measurements of process models, i.e., fitness and precision. They are equal compared to one another, using graphical algorithms—graph-based methods of processes containing non-free choices and graph-based methods of processes containing invisible tasks [8].

Alpha ++ [5], [6] algorithm can be used for non-free choice relation by describing the selected relationship in activities that depends on other activities. However, in hybrid non-free choice [12], [13] and invisible tasks [14], Alpha ++ cannot detect invisible tasks, so this method cannot discover a non-free choice relationship [13].

Different from Alpha ++, Alpha# can detect invisible task and describe the process entirely. SKIP, REDO, and SWITCH can detect Alpha #, but Alpha # cannot overlapping patterns and cannot detect the non-free choice process [7].

B. Neo4j

Neo4j is a NoSQL database that follows the mathematical tree theory for the category of graph databases. Neo4j is used for the mining process to show the process model's relationship in the form of a directed graph [15], [16]. The nodes and the relationship between them are the basis of the graph concept. Neo4j uses Cypher query language for transactional and analytical graphic data processing [17].

III. PROPOSED METHODS

This study proposes process models of checking animals' halalness and graph-based discovery algorithm to depict invisible prime tasks, mainly invisible tasks for a buildup of XOR relationships and AND relationships.

A. Process Models of Checking Animals' Halalness

Processes of checking animals' halalness are divided into ten sections: meat, fat, milk, offal, blood, skin hair, eggs, trachea, and bones. Fig. 1 until Fig. 10 shows developed process models by the authors of this study. Those process models are formed in the ProcessMaker application [18].

- Meat: Invisible Prime Task Skip (Fig.1)

  First, the checker needs to check the meat. If the meat comes from haram animals, then the checker can conclude from there. However, if the meat comes from halal animals, the checker can continue to the next process. Second, the checker needs to check the slaughter process. If the slaughter process not following Islamic law, then the checker can conclude from there. However, if the slaughter process follows Islamic law, the checker can continue to the next process. Third, the checker needs to check the meat content. After all that, the checker can conclude.

- Fat: Invisible Prime Task (Fig.2)

  First, the checker needs to check where the fat comes from. If the fat comes from mammals/poultry, the checker needs to check the fat fraction. If the fat comes from fish, the checker needs to check its acid content. Second, if the fat comes from mammals/poultry, the checker needs to check the fat fraction. However, if the fat comes from fish, the checker needs to check the fat's flavoring. Third, if the fat comes from mammals/poultry and has a fraction, the checker needs to check the fat's glycerol-glycerin content. If the fat comes from mammals/poultry and does not have a fraction, then the checker needs to check the fat's flavoring. Fourth, the checkers need to check the fat's E number. After all that, the checker can conclude.

- Milk: Non-Free Choice (Fig.3)

  First, the checker needs to identify the milk ingredients. If the milk ingredients are cheese, then the checker needs to check the enzymes in cheese. However, if the milk ingredients are whey, the checker needs to check the whey enzymes. Second, the checker needs to check rennet content. Third, if the milk ingredients are cheese, the checkers need to check the cheese's acidity. However, if the milk ingredients are whey, the checker needs to check the whey's liquid source. Fourth, the checker needs to check for lumps. After all that, the checker can conclude.

- Offal: Invisible Non-Prime Task XOR-AND (Fig.4)

  First, the checker needs to check where the offal comes from. Second, if the offal comes from an animal digestion system, the checker needs to check the enzyme content AND intestinal organs. However, if the offal comes from an animal's pancreas, the checker needs to check the hormone insulin in the pancreatic innards. Third, the checker needs to check rennet content in the offal. After all that, the checker can conclude.

- Blood: XOR relationship(Fig.5)

  First, the checker needs to check the globin concentrate content on the blood. Second, the checker needs to match the fibrinogen concentrate content on the blood. After all that, the checker can conclude.

- Skin: XOR relationship (Fig.6)

  First, the checker needs to check the gelatin content on the skin. Second, the checker needs to match the collagen content in the skin. After all that, the checker can conclude.

- Fur/Hair: XOR relationship (Fig.7)

  First, the checker needs to check the cysteine content on the fur/hair. Second, the checker needs to check phenylamine content on the fur/hair. After all that, the checker can conclude.

- Egg: XOR relationship (Fig.8)

  First, the checker needs to check the glycol oxidase enzyme content on the egg. After that, the checker can conclude.

- Trachea: XOR relationship (Fig.9)

  First, the checker needs to check the chondroitin content on the trachea. After that, the checker can conclude.

- Bone: XOR relationship (Fig.10)

  First, the checker needs to check the gelatine content on the bone. After that, the checker can conclude.

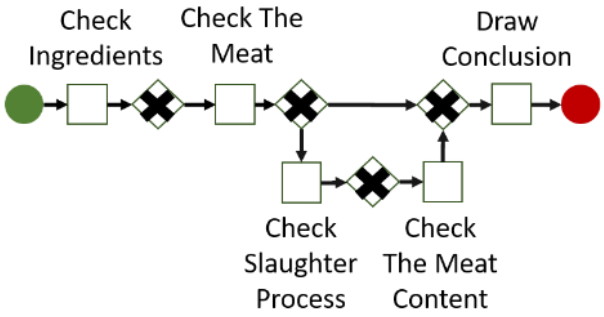

Fig 1. Meat BPMN

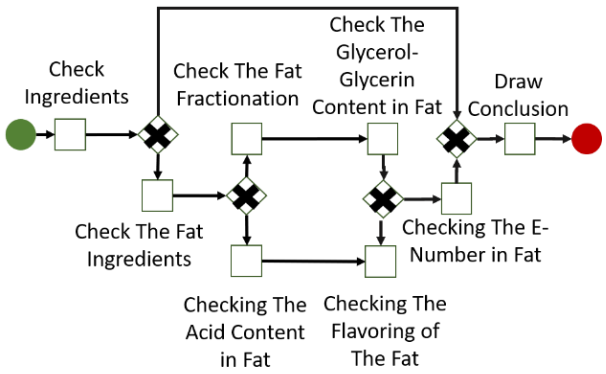

Fig 2. Fat BPMN

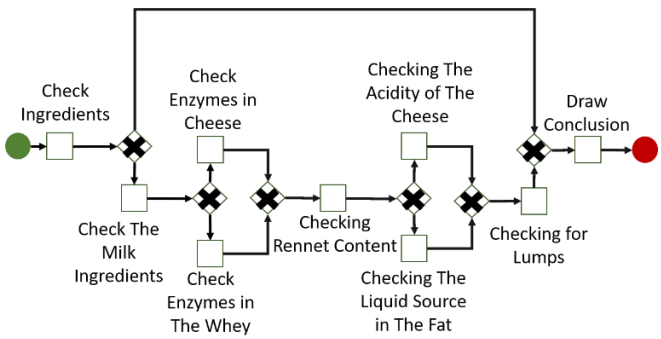

Fig 3. Milk BPMN

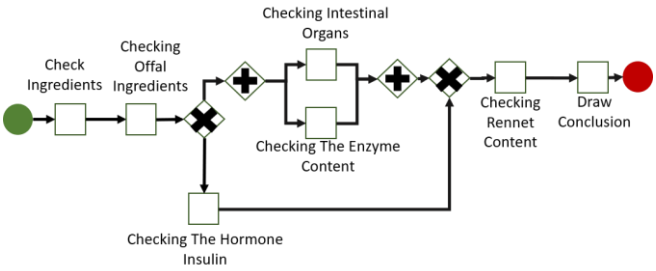

Fig 4. Offal BPMN

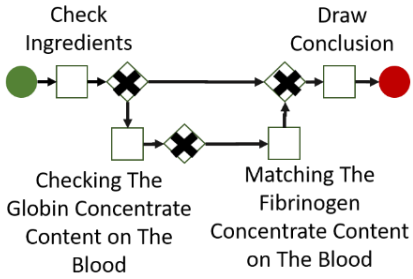

Fig 5. Blood BPMN

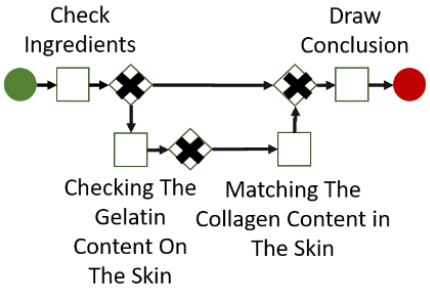

Fig 6. Skin BPMN

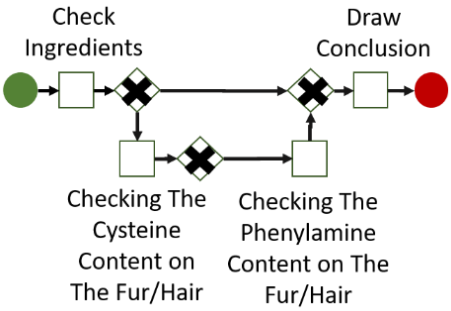

Fig 7. Fur/Hair BPMN

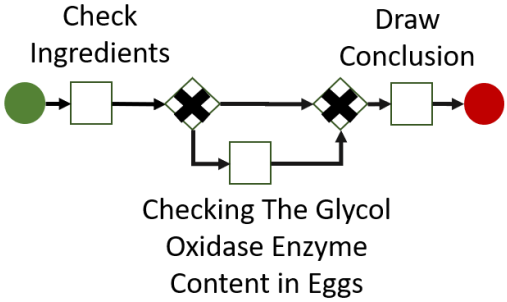

Fig 8. Egg BPMN

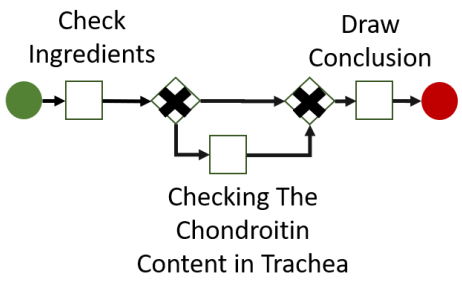

Fig 9. Trachea BPMN

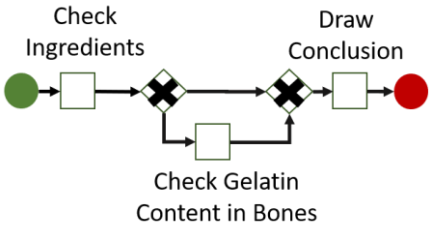

Fig 10. Bone BPMN

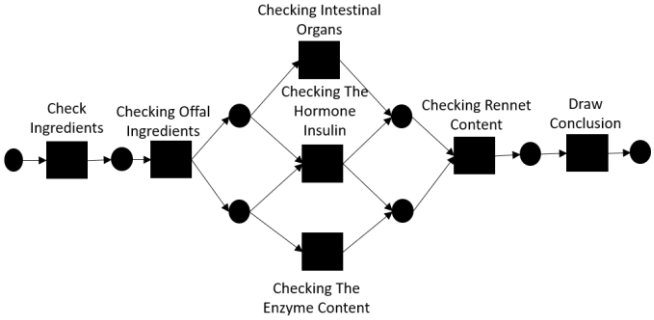

Fig 11. Alpha++ Offal: Invisible Non-Prime Task

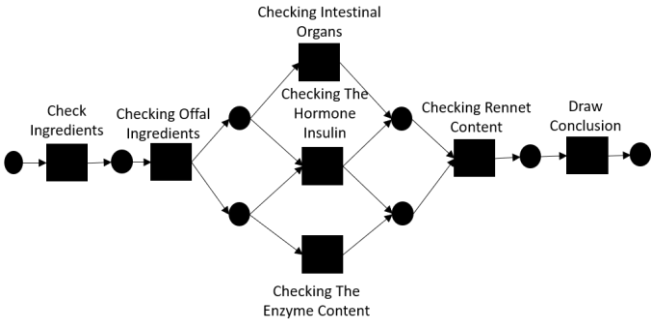

Fig 12. Alpha# Offal: Invisible Non-Prime Task

B. Graph-Based Process Discovery Method

From the event log generated by ProcessMaker, authors discover processes by using a proposed graph-based process discovery method. The proposed method, described in TABLE I, is built-in Neo4j by using Cypher query. An input data of the proposed method is an event log containing activities which have information, such as CaseID, Name and Timestamps. All activities are modified as a graph: the activities are stored as nodes. That information is saved as attributes of the nodes (can be seen in Code Activity and CaseActivity of TABLE I). After that, sequence relationships are performed using the Activity and CaseActivity node. Then, parallel relationships, i.e., XOR, AND, and OR, are created using the proposed queries in TABLE I. After all, relationships are made, detecting invisible tasks or non-free choices using the rules of invisible task, non-free choice, and invisible non-prime task in TABLE I.

TABLE I. CYPHER QUERY OF GRAPH-BASED PROCESS DISCOVERY

Code	**Activity**
Activity	LOAD CSV with headers FROM "file:///agile.csv" AS line Merge (:Activity{CaseId:line.CaseID, Name:line.Activity, Time:line.Timestamp})
CaseActivity	LOAD CSV with headers FROM "file:///agile.csv" AS line Merge (:CaseActivity {Name:line.Activity })
Sequence	MATCH (c:Activity) WITH COLLECT(c) AS Caselist UNWIND RANGE(0,Size(Caselist) - 2) as idx WITH Caselist[idx] AS s1, Caselist[idx+1] AS s2 MATCH (b:CaseActivity),(a:CaseActivity) WHERE s1.CaseId = s2.CaseId AND s1.Name = a.Name AND s2.Name = b.Name MERGE (a)-[r:SEQUENCE]->(b)
XOR Split	MATCH (bef)-[r]->(aft) WHERE size((bef)-->())>1 AND size((aft)<--())=1 AND (size((aft)-->())=1 OR size((aft)-->())>1) CREATE (bef)-[:XORSPLIT]->(aft) DELETE r
XOR Join	MATCH (bef)-[r]->(aft) WHERE (size((bef)-->())=1 OR size((bef)-->())>1) AND size((aft)<--())>1 CREATE (bef)-[:XORJOIN]->(aft) DELETE r
AND Split	MATCH (aft1)<-[r]-(bef)-[s]->(aft2) WHERE size((bef)-->())>1 AND size((aft2)-->())=size((bef)-->()) AND size((aft1)-->())=size((bef)-->()) AND not (aft1)-[:SEQUENCE]->(bef) AND not (aft2)-[:SEQUENCE]->(bef) MERGE (aft1)<-[:ANDSPLIT]-(bef)-[:ANDSPLIT]->(aft2) DELETE r,s
AND Join	MATCH (aft1)-[r]->(bef)<-[s]-(aft2) WHERE size((bef)<--())>1 AND size((aft2)-->())=size((bef)<--()) AND size((aft1)-->())=size((bef)<--()) AND not ()-[:ANDSPLIT]->(bef) MERGE (aft1)-[:ANDJOIN]->(bef)<-[:ANDJOIN]-(aft2) DELETE r,s

Code	Activity
OR Split	MATCH (bef1)-[r]->(aft)<-[s]-(bef2), (bef2)<-[v]-(bef4)-[w]->(bef1) WHERE size((bef4)-->()) >1 AND size(()--> (aft)) >1 AND size(()-->(bef2)) < size(()-->(bef1)) AND NOT((size(()-->(bef1)) = size (()-->(bef2)) = size((bef4)-->())) OR(size(()-->(bef1)) = size (()-->(bef2)) = 1)) AND NOT (size(()-->(bef4)) + size((bef4)-->()) = size(()-->(bef1)) + size((bef1)-->()) OR size(()-->(bef4)) + size((bef4)-->()) = size(()-->(bef2)) + size((bef2)-->())) AND NOT (bef1)-[:SEQUENCE]->(bef4) AND NOT (bef2)-[:SEQUENCE]->(bef4) MERGE (bef1)<-[:ORSPLIT]-(bef4)-[:ORSPLIT]->(bef2) DELETE w,v
OR Join	MATCH (aft1)-[r]->(bef)<-[s]-(aft2) WHERE size(()-->(bef)) > 1 AND NOT (bef)-[:SEQUENCE]->(aft1) AND NOT (bef)-[:SEQUENCE]->(aft2) AND size((aft2)-->()) > 1 AND size((aft2)-->()) < size(()-->(bef)) AND NOT size((aft2)-->())= 1 AND NOT size((aft1)-->())=1 AND NOT ()-[:ORSPLIT]->(bef) MERGE (aft1)-[:ORJOIN]->(bef)<-[:ORJOIN]-(aft2) DELETE r,s
Invisible Task	MATCH (aft2)<-[r]-(bef)-[s]->(aft1) WHERE (TYPE(s)='XORSPLIT' AND TYPE(r)='XORJOIN') CREATE (i:Invisible_Task {Name: "Inv_Task1"}) WITH bef, i, aft2, r, s CALL apoc.create.relationship(bef, type(s), {}, i) YIELD rel as relation1 CALL apoc.create.relationship(i, type(r), {}, aft2) YIELD rel as relation2 DELETE r MATCH (aft2)-[r]->(bef)<-[s]-(aft1) WHERE (TYPE(s)='XORJOIN' AND TYPE(r)='XORSPLIT') CREATE (i:Invisible_Task {Name: "Inv_Task2"}) WITH bef, i, aft2, r, s CALL apoc.create.relationship(bef, type(s), {}, i) YIELD rel as relation1 CALL apoc.create.relationship(i, type(r), {}, aft2) YIELD rel as relation2 DELETE r
Non-Free Choice	MATCH (a)-[b:XORJOIN]->() MATCH ()-[c:XORSPLIT]->(n) MATCH (k:Activity),(l:Activity) WHERE a.Name<>n.Name AND k.Name=a.Name AND l.Name=n.Name AND k.CaseId=l.CaseId AND k.Time<l.Time MERGE (a)-[:NONFREECHOICE]->(n)
Invisible Non-Prime Task (XOR-AND)	MATCH (a)-[b]->(c:Invisible_Task)-[d]->(e) MATCH (a)-[f]->(g:Invisible_Task)-[h]->(i) MATCH (e)-[j]->(i) MATCH (i)-[k]->(e) WHERE e.Name<>i.Name AND id(c)>id(g) DETACH DELETE g DELETE d,j,k MERGE (c)-[:ANDSPLIT]->(e) MERGE (c)-[:ANDSPLIT]->(i) MATCH (a)<-[b]-(c:Invisible_Task)<-[d]-(e) MATCH (a)<-[f]-(g:Invisible_Task)<-[h]-(i) WHERE e.Name<>i.Name AND id(c)>id(g) DETACH DELETE g DELETE d MERGE (c)<-[:ANDJOIN]-(e) MERGE (c)<-[:ANDJOIN]-(i)

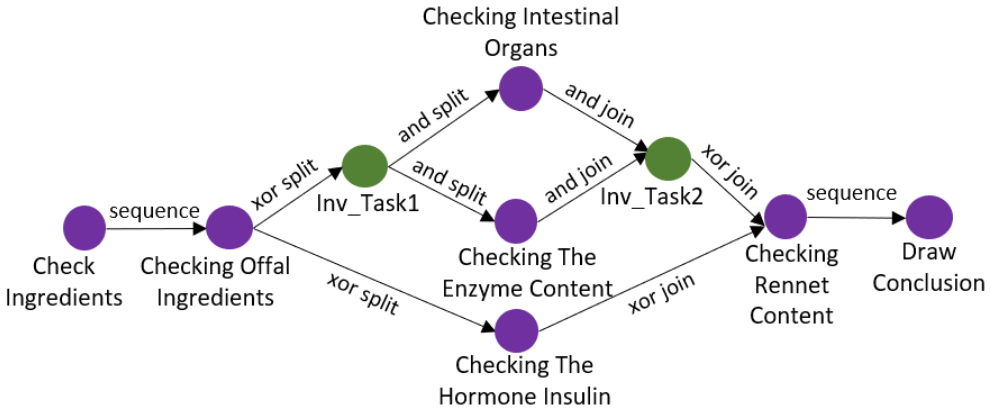

Fig 13. Neo4j Offal: Invisible Non-Prime Task

IV. Result and Discussion

A. Result

This study uses simulation event logs which are generated based on processes of checking animals' halalness. The authors of this study develop models of those processes. Out of all processes, only a process of checking offal has a condition providing an invisible non-prime task. This study utilizes the event logs from ten process models of checking animals' halalness to evaluate the proposed graph-based process discovery algorithm. This study compares the proposed algorithm with other algorithms, which are Alpha ++ and Alpha #.

The used process of checking offal has 30 cases and three traces. Traces are variations of processes captured in the event log. Traces of checking offal are [*Check Ingredients, Checking Offal Ingredients, Checking The Enzyme Content, Checking Intestinal Organs, Checking Rennet Content, Draw Conclusion], [Check Ingredients, Checking Offal Ingredients, Checking Intestinal Organs, Checking The Enzyme Content, Checking Rennet Content, Draw Conclusion], [Check Ingredients, Checking Offal Ingredients, Checking The Hormone Insulin, Checking Rennet Content, Draw Conclusion]*.

The obtained process models based on the event log of offal processes are mentioned in Fig. 11, Fig. 12 and Fig. 13. Fig. 11 is obtained using Alpha ++, Fig. 12 is depicted using Alpha #, and Fig. 13 is discovered using the graph-based process discovery method.

B. Discussion

For comparison, this study compares the result discovered by the graph-based method with those depicted by Alpha ++ and Alpha #. ProM applications perform both Alpha ++ and Alpha #. The results of those two algorithms are formed in Petri-net. The development of the graph-based method is shown in Fig. 13, while those of Alpha ++ and Alpha # are described in Fig. 11 and Fig. 12, respectively.

Based on the performance task in TABLE II, Alpha ++ can depict a non-free choice, while Alpha # does not. On the other hand, Alpha # can discover an invisible task, while Alpha ++ does not. Alpha ++ and Alpha # cannot describe invisible non-prime tasks, while the proposed graph-based method can depict them. The ability to discover invisible non-prime task is the contribution of this study.

This study compares those algorithms using ten event logs based on proposed process models of checking animals' halalness. The quality of obtained process models of those algorithms is measured based on fitness value (Eq. 1) and precision value (Eq. 2). The graph-based process discovery method has one fitness value and one precision value for all event logs based on the evaluation. The evaluation results verify the proposed graph-based method provides the best quality of the process model (can be seen in TABLE III).

TABLE II. COMPARISON BETWEEN ALPHA++, ALPHA#, AND GRAPH-BASED PROCESS DISOVERY

Task	**Alpha++**	**Alpha#**	**Graph-based Process Discovery**
Invisible [switch]	failed	succeed	succeed
Invisible Task [skip]	failed	succeed	succeed
Invisible Non-Prime Task	failed	failed	succeed
Non-Free Choice	succeed	failed	succeed

TABLE III. COMPARISON FITNESS AND PRECISION BETWEEN ALPHA++, ALPHA#, AND GRAPH-BASED PROCESS DICSOVERY

	Alpha++		Alpha#		Graph-based Process Discovery	
	Fitness	*Precision*	*Fitness*	*Precision*	*Fitness*	*Precision*
Meat	1.00	1.00	1.00	1.00	1.00	1.00
Fat	1.00	0.75	1.00	1.00	1.00	1.00
Milk	1.00	0.333	1.00	0.5	1.00	1.00
Offal	1.00	0.43	1.00	0.43	1.00	1.00
Blood	1.00	1.00	1.00	1.00	1.00	1.00
Skin	1.00	1.00	1.00	1.00	1.00	1.00
Fur/ Hair	1.00	1.00	1.00	1.00	1.00	1.00
Egg	1.00	1.00	1.00	1.00	1.00	1.00
Trachea	1.00	1.00	1.00	1.00	1.00	1.00
Bone	1.00	1.00	1.00	1.00	1.00	1.00

$$Fitness(x) = \frac{N(Captured\_Cases)}{N(Cases\_In\_Event\_Log)} \quad (1)$$

where:

$N(Captured\_Cases)$: the total of cases which are depicted in the discovered process model

$N(Cases\_In\_Event\_Log)$: the total of cases in the event log

$$Precision(x) = \frac{N(Capture\_Trace)}{N(Traces\_In\_Event\_Log)} \quad (2)$$

where:

$N(Captured\_Traces)$: the total of traces of the discovered process model which are captured in the event log

$N(Traces\_In\_Event\_Log)$: the total of traces in the event log

V. Conclusion

This study proposes process models to check animals' halalness and a graph-based process discovery method to discover invisible non-prime tasks. The condition of the invisible non-prime task is a buildup of XOR relationships and AND relationships.

This study obtains ten process models for checking animals' halalness. Those process models for checking meat, fat, milk, offal, blood, skin hair, eggs, trachea, and bones. Then, this study creates a graph-based method, which is compared with existing process discovery algorithms, such as Alpha ++ and Alpha #. Based on event logs provided based on the proposed process models, the graph-based method provides one fitness value and one precision value for all event logs, including an event log of the offal process that contains invisible non-prime tasks. Both Alpha ++ and Alpha # have one fitness value for the event log of the offal process; however, those two algorithms have a 0.43 precision value for that event log. Alpha ++ and Alpha # obtain the low precision values because process models discovered by those algorithms create additional traces which are not captured in the event log. The additional traces are raised because Alpha ++ and Alpha # cannot depict the invisible non-prime task.

There are several drawbacks of the proposed graph-based method. First, this method can be applied to the noise-free event log. Second, this method has not been implemented in the vast event log. Those drawbacks will be a challenge for future research.

Acknowledgment

This research was funded by AUN/SEED-Net under Special Program for Research Against COVID-19 (SPRAC) and the Ministry of Research and Technology/National Research and Innovation Agency Republic of Indonesia (Ristek-BRIN) and the Indonesian Ministry of Education and Culture under Newton Institutional Links - Kerjasama Luar Negeri (KLN) Program and under Penelitian Terapan Unggulan Perguruan Tinggi (PTUPT) Program managed by Institut Teknologi Sepuluh Nopember (ITS).

References

[1] K. Randeree, "Challenges in halal food ecosystems: the case of the United Arab Emirates," *British Food Journal*, 2019.

[2] R. Kamaruddin, H. Iberahim, and A. Shabudin, "Halal compliance critical control point (HCCCP) analysis of processed food," In: *2012 IEEE Business, Engineering & Industrial Applications Colloquium (BEIAC)*, 2012, pp. 383–387.

[3] W. Zzaman, N. A. Febrianto, N. S. Zakariya, W. N. W. Abdullah, T. A. Yang, and others, "Embedding Islamic dietary requirements into HACCP approach," *Food control*, Vol. 34, No. 2, pp. 607–612, 2013.

[4] K. R. Sungkono and R. Sarno, "Constructing Control-Flow Patterns Containing Invisible Task and Non-Free Choice Based on Declarative Model," *International Journal of Innovative Computing, Information and Control (IJICIC)*, Vol. 14, No. 4, 2018.

[5] Q. Guo, L. W. B, J. Wang, Z. Yan, and P. S. Yu, "Mining Invisible Tasks in Non-free-choice Constructs," In: *International Conference on Business Process Management. Springer*, 2015, pp. 109–110, doi: 10.1007/978-3-319-23063-4.

[6] N. Mehdiyev and P. Fettke, "Explainable Artificial Intelligence for Process Mining: A General Overview and Application of a Novel Local Explanation Approach for Predictive Process Monitoring," *arXiv preprint arXiv:2009.02098*, 2020.

[7] L. Wen, J. Wang, W. M. P. van der Aalst, B. Huang, and J. Sun, "Mining process models with prime invisible tasks," *Data & Knowledge Engineering*, Vol. 69, No. 10, pp. 999–1021, 2010, doi: 10.1016/j.datak.2010.06.001.

[8] R. Sarno, K. R. Sungkono, R. Johanes, and D. Sunaryono, "Graph-Based Algorithms for Discovering a Process Model Containing Invisible Tasks," *International Journal of Intelligent Engineering and Systems (IJIES)*, Vol. 12, No. 2, pp. 85–94, 2019.

[9] R. Sarno and K. R. Sungkono, "A survey of graph-based algorithms for discovering business processes," *International Journal of Advances in Intelligent Informatics*, Vol. 5, No. 2, pp. 137–149, 2019.

[10] W. M. P. van der Aalst, *Process Mining: Data Science in Action*. Springer, 2016.

[11] H. W. Sun, W. Liu, L. Qi, Y. Y. Du, X. Ren, and X. Y. Liu, "A process mining algorithm to mixed multiple-concurrency short-loop structures," *Information Sciences*, Vol. 542, pp. 453–475, 2021, doi: 10.1016/j.ins.2020.07.003.

[12] L. Hakim, R. Sarno, and K. R. Sungkono, "Modified Alpha++ Algorithm for Discovering the Hybrid of Non-Free Choice and Invisible Task of Business Processes," *International Journal of Intelligent Engineering and Systems (IJIES)*, Vol. 12, No. 3, pp. 31–40, 2019.

[13] R. Sarno and K. R. Sungkono, "Coupled Hidden Markov Model for Process Discovery of Non-Free Choice and Invisible Prime Tasks," *Procedia Computer Science*, Vol. 124, pp. 134–141, 2018, doi: 10.1016/j.procs.2017.12.139.

[14] K. R. Sungkono, R. Sarno, and N. F. Ariyani, "Refining business process ontology model with invisible prime tasks using SWRL rules," In: *2017 11th International Conference on Information Communication Technology and System (ICTS)*, 2017, pp. 215–220, doi: 10.1109/ICTS.2017.8265673.

[15] J. Joishi and A. Sureka, "Graph or Relational Databases: A Speed Comparison for Process Mining Algorithm," *arXiv preprint arXiv:1701.00072*, pp. 1–22, 2016.

[16] J. Pokorný, M. Valenta, and J. Kovačič, "Integrity constraints in graph databases," *Procedia Computer Science*, Vol. 109C, pp. 975–981, 2017, doi: 10.1016/j.procs.2017.05.456.

[17] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor, "Cypher: An Evolving Query Language for Property Graphs," 2018.

[18] E. P. Kechagias, S. P. Gayialis, G. D. Konstantakopoulos, and G. A. Papadopoulos, "An application of a multi-criteria approach for the development of a process reference model for supply chain operations," *Sustainability*, Vol. 12, No. 14, p. 5791, 2020.