Time Based Discovery of Parallel Business Processes

Riyanarto Sarno*, Kartini, Widyasari Ayu Wibowo, and Adhatus Solichah A
Informatics Department, Faculty of Information Technology, Institut Teknologi Sepuluh Nopember, Indonesia

***Abstract*— Process mining for discovering concurrent activities is important since there are many of them contained in business processes. The concurrency is formed by AND parallel or OR conditional. However, most of the existing process mining algorithm discover only concurrency formed by AND parallel. Substituting OR conditional with AND parallel does not always discover the real business processes. Also, the existing process mining algorithms use linear dependence principle; therefore, they require complete event logs which are difficult to be provided since there are many possible traces. In this regard, this paper proposes Time based Discovery algorithm which utilizes non-linear dependence principle. The proposed algorithm can effectively distinguish AND parallel and OR conditional. The experimental results show that the proposed algorithm can discover the concurrent business processes formed by AND parallel or OR conditional.**

Keywords—Parallel Bussiness Process, Incompleteness, Time Based Process Mining.

I. INTRODUCTION

Process discovery is one of the most challenging process mining task. It is a set of techniques that automatically construct a model of an organization`s current activities and its major activity variations. These techniques use event log of activities within an organization. The model is analyzed to show the complex activity problems and how to solve them. These problems exist in any field, e.g. business [1], environment [2, 3], smartphone [4], fraud [5], etc. Each techniques has different disadvantages. These techniques have disadvantages. Therefore, the effective organization`s activities can not be presented by the discovered model. Consequently, these techniques should be improved.

Process discovery comes up with many algorithm, e.g. alpha, alpha+, alpha++, genetic miner [6], and heuristic miner [7] algorithm. The alpha algorithm is the basic algorithm of process discovery. This algorithm has problem with the length one loop and length two loop. Hence, the alpha algorithm is improved to be alpha+ algorithm. However, the alpha+ algorithm also has problem with non-free choice. Therefore, the alpha+ is improved to be alpha++. Despite these improvements, the alpha++ algorithm still has several problems, i.e. completeness, noise, OR conditional. Then, the genetic miner and heuristic miner algorithms come up to solve the completeness and noise problem. However, there are no algorithm that focus on OR conditional. The algorithm usually discovers the OR conditional as AND parallel or XOR conditional. This way of discovering will change the result of activities [8].

In real life case on Fig 1, the OR conditional is the condition between "give water" and "give fertilizer" activities. The "give water" activity will be played if the soil is dry. The "give fertilizer" activity will be played every two weeks. If the OR conditional become AND parallel, then the plant will get the water every two weeks. This action will kill the plant. If the OR conditional become XOR conditional, then the plant will get only water or fertilizer. This action will unbalance the plant condition. Hence, it is important to distinguish OR conditional, XOR conditional, and AND parallel.

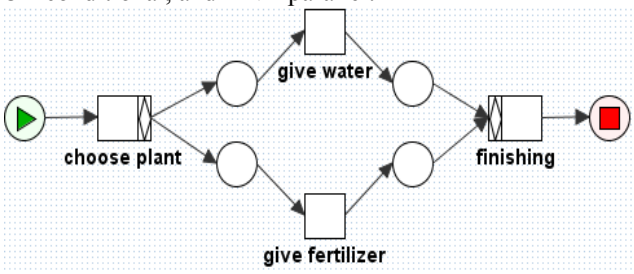

Fig 1. Activities of Plant Treatment

When it comes to process mining, the notion of completeness is also very important. It is related to noise. However, whereas noise refers to the problem of having "too much data" (describing rare behavior), completeness refers to the problem of having "too little data" [9]. The completeness problem occurs when the event log does not contain all of the possible activity`s flows. However, there are so many possible activity`s flows. Therefore, the event log is barely possible to be complete. The alpha, alpha+, and alpha++ algorithm discover the flow between activities using linear dependence principle [10]. However, the linear-dependence principle is less efficient compared to non-linear dependence principle using activity lifespan [10]. The non-linear principle need less data in the event log. Hence, it can be stated that the non-linear principle can reduce the need of complete event log.

In this paper, the proposed method focuses on discovering OR conditional and reducing the need of complete log. It defines several definitions and algorithms to discover OR conditional. And it uses the non-linear dependence principle using activity lifespan to reduce the need of complete log. The advantages of non-linear dependence is presented in mathematical approach.

II. RELATED WORK

A. Process Modeling

Aalst [9] introduces some basic process modeling notations, i.e. transition system, Petri nets, and YAWL. Transition system is the most basic process modeling notation. It is simple but have problem expressing concurrency succinctly. Suppose that there are n parallel activities, i.e. all n activities need to be executed but any order is allowed. There are n! possible execution sequences. Petri nets are the oldest and best investigated process modeling language allowing for modeling concurrency. However, Petri nets define only firing rules for AND-gate and XOR-gate without OR-gate. YAWL is

*Corresponding author.
Email address: riyanarto@if.its.ac.id

currently one of the most widely used open-source workflow system. YAWL offers direct support for mant patterns while keeping the language simple. These patterns are control-flow patterns, data patterns, resource patterns, etc. From the control-flow perspective, YAWL introduces some gate, i.e. AND-gate, XOR-gate, and OR-gate. The proposed method focuses on discovering OR conditional. Hence, the proposed method will uses YAWL since it defines OR-gate.

B. The Existing Algorithms

The alpha++ algorithm discovers model using linear dependence principle. It defines some rules to discover concurrency. However, it does not define rule to distinguish the concurrency occurred by AND-gate or OR-gate. The alpha++ algorithm uses Petri nets as process modeling notation. As said by Aalst, Petri nets does not firing rule for OR-gate. The OR-gate is used as OR conditional notation. Hence, the alpha++ algorithm can not discover OR conditional. The alpha++ algorithm defines its toleration of completeness in mathematical approach. The notion of completeness for event log is n(n-1) where normally n! with n as the number of parallel activities.

Rizka et. al. [10] view one process instance as one process variant. Hence, they use non linear dependence for process discovery to get parallel relation in one process instance. Moreover, the use of non linear dependence increase the precision of discovered process model from process discovery. The non-linear dependence uses activity lifespan control flow to enable the relations in process instances. And the control flow uses temporal causal relation to note the relation between activities in event log. Additionally, they introduce an algorithm from them. The main steps of the algorithm are listing all input and output activities and classifying sequence and parallel relations. However, they do not distinguish the parallel formed by AND or OR. Hence, the proposed method introduce the definitions of AND parallel and OR conditional.

The temporal causal relation and activity lifespan control flow is described in Definition 1 and Definition2. And the non-linear dependence is described in Definition 3.

Definition 1. Temporal Causal Relations. Given event log (L) and trace (σ) such that $\sigma \in L$. The causal relation between two activities $A(A_s, A_f)$ and $B(B_s, B_f)$, according to which $A, B \in L$ can be differentiated as follows:

$Before\ and\ meets, A > B\ iff\ A_f \leq B_s$
$Overlaps, A \square B\ \ iff\ A_f > B_s\ and\ A_f < B_f$
$Contains, A@B\ iff\ A_s < B_s\ and\ B_f > A_s\ and\ A_f > B_f$
$Is\ finish\ by, A_f B\ iff\ A_f = B_f\ and\ A_s < B_s\ and\ B_s < A_f$
$Equals, A \Diamond B\ iff\ A_s = B_s\ and\ A_f = B_f$
$Starts, A_p B\ iff\ A_s = B_s\ and\ A_f > B_f$

Definition 2. Activity Lifespan Control Flow. Given event log (L) and trace (σ) such that $\sigma \in L$. The control flow between two activities $A(A_s, A_f)$ and $B(B_s, B_f)$, according to which $A, B \in L$ can be differentiated as follows:

$Sequence, A \rightarrow B\ \ iff\ A > B$
$Parallel, A \parallel B\ \ iff\ A > B\ and\ B > A\ or$
$\{A \square B\ or\ A@B\ or\ A_f B\ or\ A \Diamond B\ or\ A_p B\}$

Definition 3. Non-Linear Dependency in Process Instance. Given event log L and trace σ such that $\sigma \in L$, a sequence relation a→b and a→c between activities $a(e_s,e_f)$, $b(e_s,e_f)$. and $c(e_s,e_f)$, such that a,b,c∈A if a>b, a>c, and b || c. As well as, a sequence relation b→d and c→d between activities $a(e_s,e_f)$, $b(e_s,e_f)$. and $c(e_s,e_f)$, such that a,b,c∈A iff b>d, c.d, and b || c.

III. THE PROPOSED METHOD

A. Definition and Formulation

The proposed method introduced some definitions to cope with the algorithm presented in this paper. The algorithm employs this definitions together with the definitions mentioned in Section 2. The definitions proposed in the methodology are described in Definition 4 and 5.

Definition 4. Classification of Parallel. Given event log (L) and trace (σ) such that $\sigma \in L$. The control flow between two activities $A(A_s, A_f)$ and $B(B_s, B_f)$, according to which $A, B \in L$ can be differentiated as follows:

$Conditional\ XOR, A \otimes B\ if\ there\ exist\ only\ A\ or\ B$
$in\ any\ traces\ in\ the\ log$
$Parallel\ AND, A \bullet B\ \ iff\ A$
$\parallel B\ and\ there\ not\ exists\ A \otimes B$
$in\ any\ traces\ in\ the\ \log(L)$
$Conditional\ OR, A \oplus B\ \ iff\ A \parallel B\ an\ there\ exists\ A \otimes B$
$in\ any\ traces\ in\ the\ \log(L)$

Definition 5. Notion of Completeness. Given event log (L), model (M), and activity *(A)*. Notion of completeness between the event log (L) and the model *(M)* are stated as follows:

- $A \in M$, *iff* $A \in L$.
- $(A_1 \rightarrow A_2) \in M$, *iff* $(A_1A_2)^1 \in L$ *where i in* A_i *is an identifier of an activity and n in* $(A_1A_2)^n$ *is* A_1 *and* A_2 *relation's frequency.*
- $(A_1 \,//\, A_2) \in M$, *iff* $[(A_1A_2) \in L$ *and* $(A_2A_1) \notin L]$ *or* $[(A_1A_2) \in L$ *and* $(A_2A_1) \in L]$.

The notion of completeness in Definition 5 is stating the required conditions for event log used in process discovery. These conditions inflicts the event log to have a certain amount of traces inside it. The amount of traces in event log is different between business process models. It is determined by the number of parallel relations, the number of parallel activities, and the number of parallel branches in the event log. Therefore, to calculate the right amount of traces shall be stored in event log, this paper introduced some formulations described in Formulation 1 to 5.

Formulation 1. AND parallel. Given a model *(M)*, number of parallel activities at i[th] branch *(n_i)* and number of parallel branches *(p)*. The model *(M)* consists of one AND parallel. The amount of traces needed in event log to discover this model is calculated using the following equation.

$$\boldsymbol{AND\_T = \sum_{i=1}^{p} \sum_{j=i+1}^{p} n_i \times n_j}$$

Formulation 2. OR Conditional. Given a model *(M)*, number of parallel activities at i[th] branch *(n_i)* and number of parallel branches *(p)*. The model *(M)* consists of one OR conditional. The amount of traces needed in event log to discover this model is calculated using the following equation.

$$\boldsymbol{OR\_T = \left(\sum_{i=1}^{p} \sum_{j=i+1}^{p} n_i \times n_j \right) + 1}$$

Formulation 3. XOR Conditional. Given a model *(M)*, relation on i[th] branch *(R_i)* and number of parallel branches *(p)*. The model *(M)* consists of one XOR conditional. The amount of traces needed in event log to discover this model is calculated using the following equation.

$$\boldsymbol{XOR\_T = \sum_{i=1}^{p} if\ R_i = seq, 1\ else\ 0}$$

Formulation 4. Set of Parallels. Given a model *(M)* consists of one set of parallels which is formed with one AND parallel other parallel inside it. The amount of traces needed in event log to discover this model is calculated using the following equation.

$$\boldsymbol{PT = AND\_T + OR\_T + XOR\_T}$$

Formulation 5. The Whole Process Model. Given a model *(M)* and the maximum number of traces for one set of parallels *(PT_{max})*. The model *(M)* consists of only sequence relation or together with parallel relation. The amount of traces needed in event log to discover this model is calculated using the following equation.

$$\boldsymbol{CT = If\ seq, 1\ else\ PT_{max}}$$

B. The Algorithm

The proposed algorithm uses Definition 1, 2, and 3 to discover relations from event log. Then, it uses Definition 4 to differentiate parallel relations to be AND parallel, OR conditional, or XOR conditional. Finally, it merges all relations using non-linear dependence in Definition 3. The detail steps of it are described as follows:

Algorithm 1.

Step 1. List input (I) and output(O) activities from every trace (I_i, O_i) in the event log.

Step 2. List the sequence relations (>) from every trace $(>_i)$ in the event log.

Step 3. List the parallel relations (||) from every trace $(\|_i)$ in the event log.

Step 4. Classify the parallel relations (||) to be AND parallel, OR conditional, and XOR conditional.

Step 5. Merge between members in the AND parallel relation.

$foreach\ R\ i \bullet, which\ (A,B) \wedge (C,D) \in R$
$iff\ A=(C \vee D)\ \ nd\ [(C \vee D),B] \vee [B,(C \vee D)] \in >\_L\ then$
$[A,(B,(C \vee D))]$

Step 6. Merge between members in the OR conditional relation.

$foreach\ R\ i \oplus, which\ (A,B) \wedge (C,D) \in R$
$iff\ A=(C \vee D)\ \ and\ [(C \vee D),B] \vee [B,(C \vee D)] \in >\_L\ then$
$[A,(B,(C \vee D))]$

Step 7. Form a graph using the relations of AND parallel, OR conditional, and XOR conditional.

Step 8. Add the sequence relations and input-output to the graph.

$foreach\ R\ in >\_L, which\ (A,B) \in R$
$iff\ (A,B) \notin G$
$iff\ (\bullet B)\ \ ot\ exist$
$G \leftarrow G \cup (A,B)$
$else\ iff\ A \bullet C, which\ (C,B) \in G$
$G \leftarrow G \cup [(A,C) \bullet B]$
$else\ iff\ A \oplus C, which\ (C,B) \in G$
$G \leftarrow G \cup [(A,C)\ \oplus B]$
$else\ G \leftarrow G \cup [(A,C) \otimes B]$

The algorithm is applied on the event log shown in Table 1 (L = {ABBC, ABDEC, ADBEC, ADEBC, ABDEDEC}).

TABLE 1. AN EVENT LOG

Trace	Task	Start	Finish
1	A	22:44	22:51
	B	23:02	23:08
	B	23:11	23:16
	C	23:22	23:32
2	A	24:50	24:58
	B	25:00	25:23
	D	25:05	25:18
	E	25:27	25:37
	C	25:43	25:51
3	A	25:46	26:21
	D	26:33	27:59
	B	27:08	28:27
	E	28:18	28:45
	C	28:52	29:20
4	A	32:00	32:35
	D	32:42	34:25
	B	33:40	35:50
	E	34:45	35:28
	C	36:02	36:31
5	A	37:55	38:15
	B	38:21	38:47
	D	38:31	38:45
	E	38:53	39:06
	D	39:12	39:25
	E	39:30	39:43
	C	39:45	40:01

The event log (L) is described on Gantt chart to show the different time execution. The first to third steps of the proposed algorithm discover the relation from each traces.

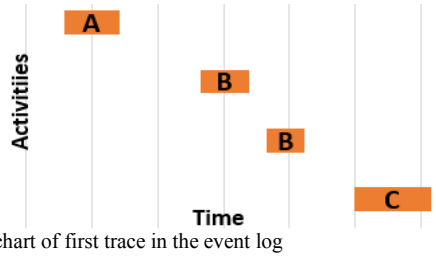

Fig 2. Gantt chart of first trace in the event log

The discovered relations in first trace are described as follows:

Step 1. $I_1 = A$, $O_1 = C$.

Step 2. $>_1 = \{\{\} \rightarrow A, A \rightarrow B, B \rightarrow B, B \rightarrow C, C \rightarrow \{\}\}$.

Step 3. $\|_1 = \{\}$.

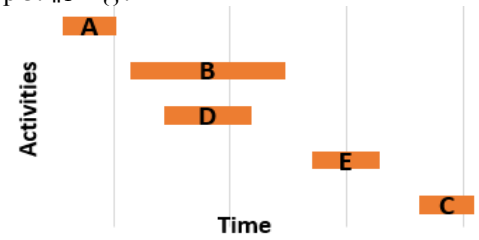

Fig 3. Gantt chart of second trace in the event log

The discovered relations in first trace are described as follows:

Step 1. $I_2 = A$, $O_2 = C$.

Step 2. $>_2 = \{\{\} \rightarrow A, A \rightarrow B, A \rightarrow D, D \rightarrow E, E \rightarrow C, C \rightarrow \{\}\}$.

Step 3. $\|_2 = \{ B \| D \}$.

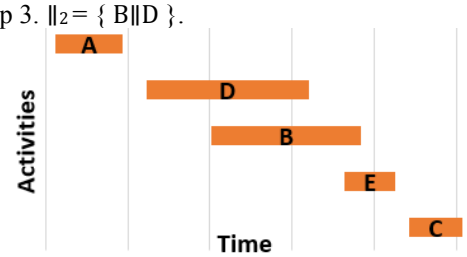

Fig 4. Gantt chart of third trace in the event log

The discovered relations in first trace are described as follows:

Step 1. $I_3 = A$, $O_3 = C$.

Step 2. $>_3 = \{\{\} \rightarrow A, A \rightarrow D, E \rightarrow C, C \rightarrow \{\}\}$.

Step 3. $\|_3 = \{ B \| D , B \| E \}$.

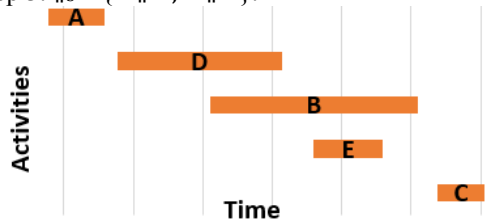

Fig 5. Gantt chart of fourth trace in the event log

The discovered relations in first trace are described as follows:

Step 1. $I_4 = A$, $O_4 = C$.

Step 2. $>_4 = \{\{\} \rightarrow A, A \rightarrow D, D \rightarrow E, B \rightarrow C, C \rightarrow \{\}\}$

Step 3. $\|_4 = \{ B \| E \}$.

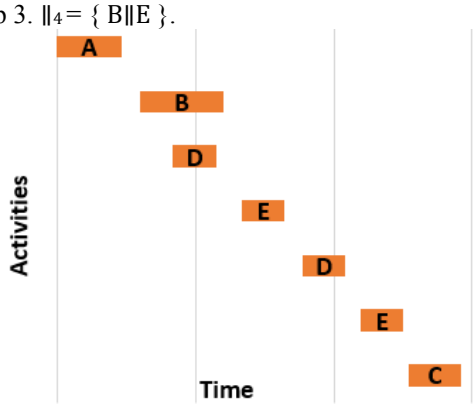

Fig 6. Gantt chart of fifth trace in the event log

The discovered relations in first trace are described as follows:

Step 1. $I_5 = A$, $O_5 = C$.

Step 2. $>_5 = \{\{\} \rightarrow A, A \rightarrow B, D \rightarrow E, E \rightarrow D, E \rightarrow C, C \rightarrow \{\}\}$.

Step 3. $\|_5 = \{ B \| D \}$.

The relations discovered in every trace will be united in each classifications.

Step 1. I = A, O = C.

Step 2. > = {{}→A, A→B, B→B, B→C, D→E, E→C, A→D, E→D, C→{}}

Step 3. ‖={B‖D, B‖E}

Step 4. • = {}

Step 5. ⊕ = {B⊕D, B⊕E}

Step 6. ⊕ = {B⊕(D,E)}

Step 7. G = {⊕, •}

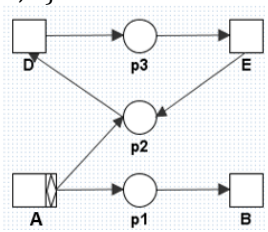

Fig 7. Graph composed by OR and AND relations

Step 8. $G \leftarrow G \cup (>, I, O)$

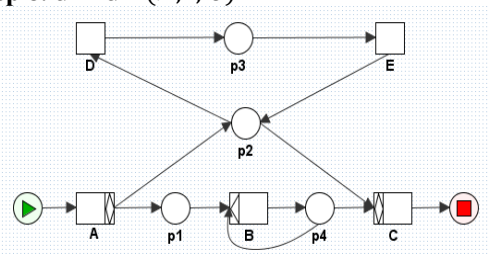

Fig 8. The final graph

IV. Evaluation

The proposed method can discover the OR conditional and length two loop shown by its demonstration in Section 3. The example in Section 3 is modified to show the proposed algorithm distinguishing the concurrency formed by AND parallel or OR conditional. The current event log (L) is

modified to be L`( L` = { ABDEC, ADBEC, ADEBC, ABDEDEC}). The first trace in the event log (L) does not include in the modified event log (L`). Then, the modified event log (L`) is discovered by the proposed method. The relations discovered from the modified event log (L`) are as follows:

Step 1. I = A, O = C.

Step 2. > = {{}→A, A→B, B→C, D→E, E→C, A→D, E→D, C→{}}

Step 3. ‖={B‖D, B‖E}

Step 4. • = { B•D, B•E }

Step 5. ⊕ = { }

Step 6. • = {B•(D,E)}

Step 7. G = {⊕, •}

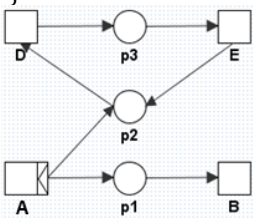

Fig 9. Graph composed by OR and AND relations

Step 8. $G \leftarrow G \cup (>, I, O)$

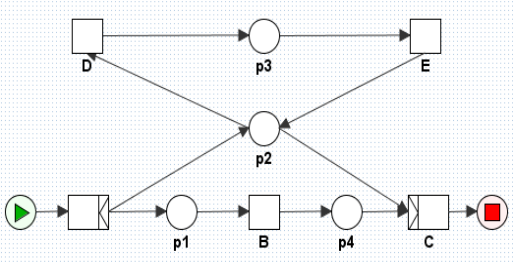

Fig 10. Final graph

From these examples, it can be stated that the proposed method can distinguish parallel as OR conditional or AND parallel while temporal activity based algorithm [10] cannot. At first example using event log L, the proposed method discover OR conditional and on the other hand, temporal activity based algorithm discover AND parallel. At second example using event log L', both proposed method and temporal activity based algorithm discover AND parallel. Furthermore, the proposed method can discover length one loop, length two loop, and XOR conditional.

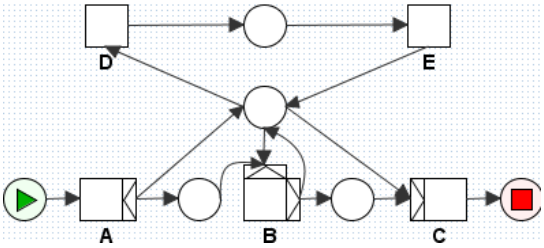

Fig 11. The Discovered Process Model Using Linear Dependence

The difference between linear and non-linear dependence is described using the proposed methodology. Therefore, the example in Section 3 is modified to show the proposed algorithm discovering more relations by using non-linear dependence principle rather than linear dependence principle. The event log (L) is modified to be L`` (L`` = { ABC, ADEBC, ABDEDEC}). The second and third traces in the event log (L) do not include in the modified event log(L``). Then, the modified event log (L``) is discovered by the proposed method using linear and non-linear dependence.

The result from the proposed method is produced such as the example in Section 3. The process model discovered from the modified event log (L``) are shown on Fig 11 and 12 representing linear and non-linear dependence.

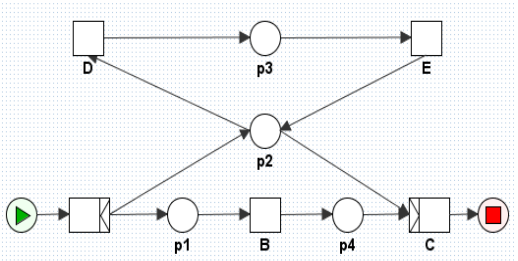

Fig 12. The Discovered Process Model Using Non-Linear Dependence

The discovered sequence relations are shown in Table 2. The discovered parallel relations are shown in Table 3. The different relations are presented in bold and italic style.

TABLE 1. THE DIFFERENT SEQUENCE RELATIONS

Trace	Non-Linear	Linear
1	{}→A, A→B, B→B, B→C, C→{}	{}→A, A→B, B→B, B→C, C→{}
2	{}→A, A→D, D→E, B→C, C→{}	{}→A, A→D, D→E, ***E→B***, B→C, C→{}
3	{}→A, A→B, D→E, E→D, E→C, C→{}	{}→A, A→B, ***B→D***, D→E, E→D, E→C, C→{}

TABLE 3. THE DIFFERENT PARALLEL RELATIONS

Trace	Non-Linear	Linear
1	-	-
2	B‖E	-
3	B‖D	-

The parallel relations in the event log (L``) cannot be found by using linear dependence principle. They are discovered as sequence relation at Step 2. The linear dependence principle regards the relation between activities as sequence relation unless it is reciprocal relation (AB, BA). The reciprocal relation is took as parallel relation. Therefore, the relation between B and E is discovered as sequence relation and so does the relation between B and D; whereas, the non-linear dependence principle discovers these relations as parallel relation. Because, it utilizes activity lifespan which can show concurrency in the log (L``) to be used with the definitions stated in this paper.

Process discovery relies on notion of completeness in Definition 5 to produced process model. Based on Formulation 1, the number of traces in event log which is fulfilled the notion of completeness to discover process model on Fig 13 using non-linear dependence is 4+3+2+1=10 traces; whereas the number of traces in event log which is fulfilled the notion of

completeness to discover process model on Fig 13 using linear dependence is (4 + 3 + 2 + 1) * 2 = 20 traces. Therefore, it can be stated that non-linear dependence is better than linear dependence in overcome incompleteness of event log. The needed number of traces in event log to discover business process by linear dependence is two times more than non-linear dependence. The difference number of traces in event log between linear and non-linear dependence is shown on Chart 1. The chart describes the difference of the number of traces to discover process model formed with n parallel activities with n as number of parallel activities. Hence, it can be stated that the more parallel activities in process model, the better process discovery using non-linear dependence than linear dependence in overcoming incompleteness event log.

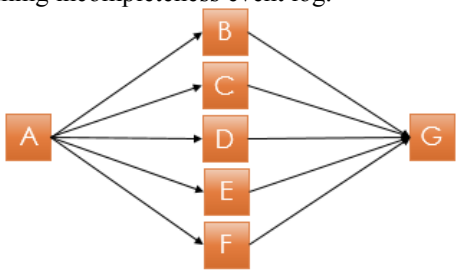

Fig 13. A model with 5 parallel activities.

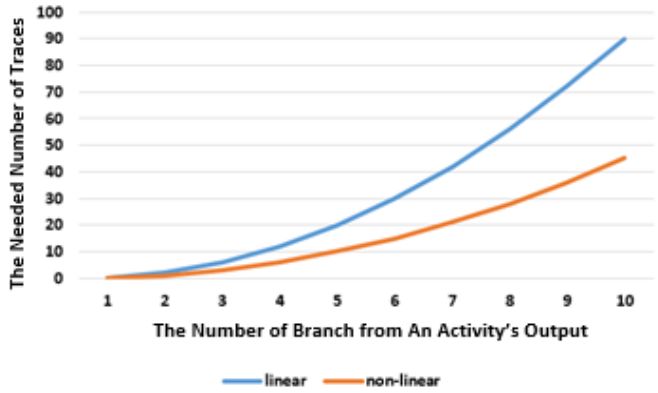

Chart 1. The difference number of traces between linear and non-linear dependence.

V. Conclusion

The paper proposes a method to discover concurrent business processes formed by AND parallel or OR conditional. The proposed method utilizes non-linear dependence principle to discover more relations contained in event logs. The event log is mined using the rules described in Definition 1 and 2 in Section 3. Then the discovered parallel relations are classified into AND parallel and OR conditional using Definition 3. Finally, all of the discovered relations are formed into a graph using the proposed algorithm. The result of the first example in Section 4 shows that the proposed method successes on distinguishing the concurrency formed by AND parallel or OR conditional. The result of the second example in Section 4 shows that the use of non-linear dependence principle reduces the need of complete logs. The needed number of traces in event log to discover business process by linear dependence is two times more than non-linear dependence. The noise problems caused by truncated event logs will be considered in future research.

VI. References

[1] Olatunde T. Baruwa, Miquel A. Piera, "Identifying FMS repetitive patterns for efficient search-based scheduling algorithm: A colored Petri net approah," *Journal of Manufacturing Systems,* vol. 35, pp. 120-135, 2015.

[2] Adnen Sanaa, Samir Ben Abid, Abdennacer Boulila, Chokri Messaoud, Mohammed Boussaid, Najeh Ben Fadhel, "Modelling hydrochory effects on the Tunisian island populations of Pancratium maritimum L. using colored Petri nets," *BioSystems,* vol. 129, pp. 19-24, 2015.

[3] Jie Yuan, David Oswald, Wei Li, "Autonomous traking of chemical plumes developed in both diffusive and turbulent airflow environment using Petri nets," *Expert Systems with Application,* vol. 42, pp. 527-538, 2015.

[4] Victor R.L. Shen, Horng-Yih Lai, Ah-Fur Lai, "The implementation of a smartphone-based fall detection system using a high-level fuzzy Petri net," *Applied Soft Computing,* vol. 26, pp. 390-400, 2015.

[5] Riyanarto Sarno, Rahadian Dustrial Dewandono, Tohari Ahmad, Mohammad Farid Naufal, and Fernandes Sinaga, "Hybrid Association Rule Learning and Process Mining for Fraud Detection," IAENG International Journal of Computer Science, vol. 42, no.2, pp59-72, 2015.

[6] Borja Vazquez-Barreiros, Manuel Mucientes, Manuel Lama, "ProDiGen: Mining complete, precise and minimal structure process models with a genetic algorithm," *Information Sciences,* vol. 294, pp. 315-333, 2015.

[7] Sofie De Cnudde, Jan Claes, Geert Poels, "Improving the quality of the Heuristics Miner in Prom 6.2," *Expert Systems with Applications,* vol. 41, pp. 7678-7690, 2014.

[8] Riyanarto Sarno, Putu Linda Indita Sari, Hari Ginardi, Dwi Sunaryono, Imam Mukhlash, "Decision Mining for Multi Choice Workflow Patterns," in *2013 International Conference on Computer, Control, Informatics ad Its Applications*, pp. 337-342, 2013.

[9] W. M. P. v. d. Aalst, Process Mining: Process Modelling and Analysis, Netherlands: Springer, pp. 31-42, 2010.

[10] Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Minsoo Kim, "Process Model Discovery Based On Activity Lifespan," in *International Conference on Technology Innovation and Industrial Management*, pp. 137-156, Seoul, 2014.