Abductive Workflow Mining using Binary Resolution on Task Successor Rules

Buffett, Scott

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*Abductive Workflow Mining using Binary Resolution on Task Successor Rules**

Buffett, S.
October 2008

* published at The International RuleML Symposium on Rule Interchange and Applications (RuleML 2008). Orlando, Florida, USA. October 30, 2008. NRC 50392.

Canada

Abductive Workflow Mining using Binary Resolution on Task Successor Rules

Scott Buffett

National Research Council Canada, Fredericton, NB, E3B 9W4
Scott.Buffett@nrc.gc.ca

Abstract. The notion of *abductive workflow mining* is introduced, which refers to the process of discovering important workflows from event logs that are believed to cause or explain certain behaviour. The approach is based on the notion of abductive reasoning, where hypotheses are found that, if added to a rule base, would necessarily cause an observation to be true. We focus on the instance of workflow mining where there are critical tasks in the underlying process that, if observed, must be scrutinized more diligently to ensure that they are sufficiently motivated and executed under acceptable circumstances. Abductive workflow mining is then the process of determining activity that would necessarily imply that the critical activity should take place. Whenever critical activity is observed, one can then inspect the abductive workflow to ascertain whether there was sufficient reason for the critical activity to occur. To determine such workflows, we mine recorded log activity for task successor rules, which indicate which tasks succeed other tasks in the underlying process. Binary resolution is then applied to find the abductive explanations for a given activity. Preliminary experiments show that relatively small and concise abductive workflow models can be constructed, in comparison with constructing a complete model for the entire log.

1 Introduction

The diverse sets of advantages and disadvantages that various existing workflow mining algorithms have to offer demonstrate that there is seemingly a solution to fit almost any need. Techniques such as van der Aalst's α-algorithm [8], α++ algorithm [11] and heuristic miner [10] offer mining techniques that produce concise, simplified workflows that are easy to visualize, but sacrifice accuracy. Still others such as Petrify [7] and Region Miner [9] achieve high fitness and precision, but as a result can produce workflows that are too large or complicated to be reasonably understood by a human viewer. In this case, reducing the size and complexity of a workflow model by eliminating components that are not of interest to a particular observer with a specific purpose for viewing the model, while retaining the desirable qualities of fitness and precision, can be quite advantageous. Accomplishing this is the focus of this paper.

Situations where an observer needs to view and understand very specific components of a workflow model for a particular purpose are prevalent in the

domain of compliance. In this case, the primary focus commonly seen in workflow mining of understanding how processes work so that they can be improved upon, made more efficient, or simplified, is not as crucial. Instead, the goal here lies in ensuring that individuals' behaviour is deemed appropriate within the context of the overall observed activity. Thus the workflow only needs to be analyzed to the extent that it can be determined whether a particular event is compliant with the rules, given the observed activity within which the event is situated. This greatly simplifies the compliance checking process. It is then not necessary to collect all tasks in a particular case and check against the entire workflow model to ensure consistency. Instead only the simplified model, which shows the legal traces of activity with which a particular action may be associated, needs to be analyzed. Using the simplified model will improve accuracy by reducing false positive and false negative cases, since reducing the model should naturally reduce the number of structures in the model that degrade fitness and precision, such as loops. Just as important, this reduction will also greatly simplify the cognitive burden imposed on the viewer of the workflow model. If inappropriate activity is detected, the graphical representation of the simplified model can be displayed to a compliance officer, who will more easily be able to understand why a particular action was deemed inappropriate, and quickly make a decision on the subsequent course of action.

In this regard, the task is not as simple as merely isolating a specified region or subgraph of the workflow model. Given a particular activity for which compliance checking is desirable, it needs to be determined exactly what part of the overall workflow model is needed. To accomplish this task, we introduce the concept of *abductive workflow mining*. This concept comes from the particular field of logical reasoning known as *abductive reasoning*. Abductive reasoning refers to the problem of of finding *explanations* for a particular set of facts. These explanations are found to be sets of facts and axioms that, if true, will logically imply that the original set of facts are true. We apply this concept to workflow mining by discovering workflows that, if followed in a particular trace, would offer an explanation for why a particular task was executed. Such a workflow is known as an *abductive workflow*.

In this paper, we present a method that finds abductive workflows by mining and constructing propositional task successor rules. These rules, which are represented in Kowalski form, indicate which tasks follow which other tasks in the underlying process that is being mined. Given a simple action or a complex set of activity, binary resolution is applied to the rules to determine abductive explanations for the activity in question. The set of explanations can then be simplified to form a minimal set. We choose the stronger act of identifying abductive explanations, rather than simply identifying consistent activity, due to the possibly critical nature of the activity in question. For example, if the critical activity was the act of a bank employee accessing sensitive customer data, such as a credit report, the idea is to identify activity that would necessarily imply that the critical activity would have to take place (such as a loan application being processed), rather than to identify activity that is merely consistent with

the activity in question (such as the credit report being printed). Compliance software or a privacy officer could then do a simple check to ensure that at least one such causal activity (or sequence of activities, i.e. workflow) was indeed executed.

The remainder of the paper is presented as follows. In section 2 we discuss required background theory on workflow mining and abductive reasoning that we employ in our techniques. Section 3 then defines abductive workflow, and introduces some necessary theory surrounding the concept. In section 4 we outline our logic-based technique for discovering abductive workflows, and in section 5 we presents a few results demonstrating the performance of our method. Section 6 then discusses conclusions and offers a few directions for future work.

2 Background

2.1 Workflow Mining

Workflow mining [2] refers to the process of autonomously examining a transaction log of system events, and extracting a model of the underlying process being executed by the events. Generally speaking, an log consists of a number of events, each of which being associated with a *task* and a *case*. An event's task refers to the actual activity the event represents, while the event's case refers to the instance of the underlying business process to which the event belongs. Each case in the log consists of a sequence of tasks (often referred to as a *trace*) that represent a complete and ordered set of actions that are executed in an instance of the business process. Workflow mining techniques are then used to build a model of the business process by representing the different ways a case in the process can be executed. A number of different representations have been used in the literature, perhaps the most common of which being the Petri net [3, 1]. Figure 1 represents a small example log, as well as the resulting Petri net representing the mined workflow. Any legal sequence of transitions that takes a token from the start (leftmost) place to the end (rightmost) place represents a different way of executing the business process. Thus by a quick inspection of the graphical model, one can infer different characteristics of the process, such as the fact that C and D can be executed interchangeably, due to the preceding "OR-split" where one place points to the two transactions, or that both B and either C or D can be executed in parallel after A, due to the preceding "AND-split" where A points to two places.

2.2 Abductive Reasoning

Abduction is an instance of assumption-based reasoning where the focus is to determine a hypothesis (or a set of hypotheses) that, if true, would necessarily explain an observation or evidence in question [4]. This type of reasoning is particularly useful when one observes an interesting or curious event or happening, and may want to know what could possibly cause the event to be true. That is,

(*Case, Task*)		
1. (1, A)	9. (4, B)	17. (3, E)
2. (1, B)	0. (3, D)	18. (6, A)
3. (2, A)	11. (2, B)	19. (5, B)
4. (2, C)	12. (2, E)	20. (4, E)
5. (1, C)	13. (3, B)	21. (6, D)
6. (3, A)	14. (4, D)	22. (6, B)
7. (1, E)	15. (5, A)	23. (5, E)
8. (4, A)	16. (5, C)	24. (6, E)

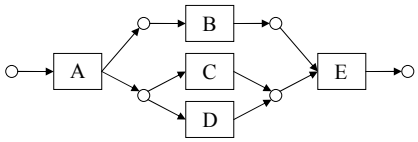

Fig. 1. Example log and corresponding workflow diagram.

given evidence E and rule base R, abductive reasoning attempts to find facts that, when added to R, would necessarily imply E. Such facts are then likely to have a causal effect on E in the real-world application being modeled by the rule base.

Abductive reasoning is of particular importance in fault diagnosis in complex systems. Here the rule base represents the operational characteristics of the system, and an observation is typically some fault or error in system function. Abductive reasoning is then used to determine what sort of activities would necessarily cause the fault to occur. These activities are then considered to be the primary causes of the error, and are thus investigated first. So abduction is applied to diagnosis with the idea that actions that would necessarily cause a fault to occur are more likely to be the true cause than actions that are merely consistent with the fault occurring.

We consider the application of abductive reasoning to workflow analysis in a similar way. Here the complex system to be modeled is the business process being analyzed, and the activities in question are the more serious or potentially harmful tasks that are executed within the process. Such critical tasks should not be executed freely; rather there should be an impelling reason for such tasks to be performed. Abductive reasoning can then be applied to determine what sort of activities would be sufficient for validating the critical behaviour in question, allowing one to easily verify whether the behaviour was warranted in any particular situation.

3 Abductive Workflow

Let W be a workflow model constructed based on a set T of tasks and a set C of cases of executions of T, as defined in section 2.1 above. Additionally, let W' be a sub-workflow of W, for which an explanation is desired. Note that W' may often simply consist of a single transition, representing the event to be explained, but in general could be any workflow structure. We consider a workflow W_a to be an *abductive workflow* of W for W' if and only if any activity observed in W_a necessarily implies that there is associated activity in W'. This means that observing activity in W_a necessarily implies that activity in W' will occur, and thus W_a is an explanation for W'.

Note that one might think that an easy way to find an abductive workflow that satisfies this condition is to simply set $W_a = W'$. So, for example, one could say that an explanation for a task A be executed is to observe that A is executed. Clearly, something stronger is needed. We need to find what sort of extra activity, not including A, would cause A to be executed. So, generally speaking, an explanation for W' is not allowed to include W'. W' may reside within the abductive model, but the activity surrounding W' must be sufficient for W' to occur. In this section we lay out formal definitions for the properties such as this for a workflow model to be considered an abductive workflow. In the following section we then offer general ideas for how an abductive workflow model can be constructed.

Definition 1 (consistent) *Let C be a set of cases and let W be a workflow model (not necessarily built based on C). A case $c \in C$ is said to be consistent with W if there is a valid trace through W that is a (non-empty) substring of c. Moreover, any sequence t of events is consistent with W if there is a valid trace through W that is a (non-empty) substring of t.*

So a case c is consistent with W' if observing c would necessarily cause a trace through W' to be observed, perhaps with extra activity from c occurring before and/or after W'. Let $C^+ \subseteq C$ then be the set of cases in C that are consistent with W', and accordingly let $C \setminus C^+$ be denoted by C^-. Any case in C^+ being observed will then necessarily cause activity in W' to occur. We use this partition over the set as the basis for the abductive model. First we define an *abductive trace*, which is the key concept in modeling abductive workflow.

Definition 2 (abductive trace) *Let W' be a workflow model and let C be a set of cases partitioned into C^+ and C^- as above. A sequence t_a of events is an abductive trace in C for W' if and only if each of the following hold:*

- *t_a is a substring of some $c^+ \in C^+$*
- *For any $c^- \in C^-$, (1) t_a is not a substring of c^-, and (2) if any possible trace $t_{W'}$ through W' is a substring of t_a, then there is no string S such that the result of replacing $t_{W'}$ with S makes t_a a substring of c^-.*

So, not only does an abductive trace t_a cause activity in W', the part of t_a without the activity from W' necessarily causes activity in W'. This means that the activity in t_a that resides in W' cannot be replaced, and thus the activity in W' *must* occur. To illustrate the necessity for this restriction, consider a simple example with two cases $ABDE$ and $ACDF$, with W' simply being B. Then $ABDE$ is a positive case and $ACDF$ is a negative case. The trace ABD would not be considered an abductive trace, since executing B was not necessary in this situation. Another choice was possible, namely D. Put another way, for any event x, if A is performed immediately before x and D is performed immediately after, it is not necessarily the case that x would be B. However, BDE on the other hand would be an abductive trace (as would $ABDE$, DE, or even simply E), since it is necessarily the case that an event x followed immediately by DE

would have to be B. Using the notion of abductive traces, we define abductive workflow:

Definition 3 (abductive workflow) *Let C be a set of cases yielding workflow W. A workflow W_a is an abductive workflow of W for workflow W' if and only if every valid trace through W_a is an abductive trace in C for W'.*

Thus any activity that is consistent with W_a, minus any activity that forms a valid trace through W', will necessarily cause a valid trace through W' to be executed.

While any abductive workflow will specify activity that will explain the activity in question, there may be additional explanations that are not captured by the model. That is, it is possible that activity in W' may occur without any explanation from W_a. Ideally, we would prefer an abductive workflow that would capture explanations for every case. That is not to say that we would expect W_a to capture every possible explanation; rather, it would be equally useful to generate abductive workflows that, given any activity in W', will contain *some* explanation for that activity. To give an indication of how well an abductive workflow performs at offering such an explanation, we define the completeness measure for an abductive workflow.

Definition 4 (completeness measure) *Let $C = C^+ \cup C^-$ be a set of cases yielding workflow W and let W_a be an abductive workflow of W for W' as defined above. The completeness measure for W_a is the probability that a case in C^+ is consistent with W_a.*

The completeness measure is thus a measure of how likely it is that an occurrence of activity in W' will be explained by W_a, which is equal to the probability that a case containing a trace of activity through W' will also contain a trace of activity through W_a.

Definition 5 (complete) *An abductive workflow with completeness measure equal to 1 is said to be complete.*

Another desirable characteristic of abductive workflow is that the model be small and concise, making it easy to check, as well as less likely to introduce errors. There are two ways to measure the size of a workflow model: (1) by considering the length of the traces through the model, and (2) by considering the number of traces through the model. We consider each of these separately.

Definition 6 (task-minimal abductive trace) *An abductive trace t_a is task-minimal if there is no substring of t_a that is an abductive trace.*

Definition 7 (task-minimal abductive workflow) *An abductive workflow is task-minimal if all abductive traces making up the workflow are task-minimal.*

So a task-minimal abductive workflow allows for only small traces, but may allow for more traces than necessary. We tackle this dimension next.

Definition 8 (trace-minimal abductive workflow) *An abductive workflow is trace-minimal if there is no subgraph of the workflow with the same completeness measure.*

Thus an abductive workflow is minimal if it cannot be reduced in any way, thus reducing the number of allowable traces, without giving up some of the explanations it offers.

Putting it all together, a best possible abductive workflow is one that is:

- complete
- task-minimal
- trace-minimal

and perhaps ideally, has the fewest nodes out of all workflows that meet the above conditions.

4 Discovering Abductive Workflow using Task Successor Rules

In an effort to find abductive workflows that meet the criteria described in the previous section, we mine the log data for rules, referred to as *task successor rules*, that indicate which activity immediately follows certain tasks in the log. These rules are represented in Kowalski form [6]. A non-horn clause is represented as a rule in Kowalski form if the tail of the rule consists of the conjunction of the atoms of the negative literals of the clause, and the head consists of the disjunction of the positive literals. Thus we construct rules of the form:

$$p_1 \wedge \ldots \wedge p_m \rightarrow q_1 \vee \ldots \vee q_n$$

Such a rule is interpreted as "the set of tasks $\{p_1, \ldots, p_m\}$, when observed as a whole, is always immediately followed by one of the tasks in $\{q_1, \ldots, q_n\}$". So the conjunction of tasks $p_1, \ldots, p_m$ being observed implies the disjunction $q_1, \ldots, q_n$. For convenience, such rules are henceforth denoted as $p_1 \ldots p_m \rightarrow q_1, \ldots, q_n$.

The set of rules is constructed as follows. Let W' represent the activity for which an explanation is desired, and let $C^+ \subseteq C$ be the set of cases consistent with W' and $C^- = C \setminus C^+$, as defined above. Let $C^+_{-W'}$ be a transformation of C^+ where, for every c in C^+, there is a corresponding c' in $C^+_{-W'}$ where c' is equivalent to c with two differences: (1) the specific tasks in c that represent a trace through W' (i.e. the part that makes it consistent with W') are removed, and (2) a dummy task (denoted by "w'"), which replaces the removed trace from W', is appended to the end. For example, let W' be represented by the execution of task A followed by task B. Then the case $KABD$ in C^+ would be represented by KDw' in $C^+_{-W'}$. This then represents that a case with K executed earlier than D, with no other activity (save for activity in W'), can cause activity in W' to occur (since the case includes w'). Also, let $C^-_{W'_o}$ be the cases in C^- with dummy tasks (denoted by "w'_o") appended to the end. Given

these transformations, a task successor rule is then created for every subsentence of activity (minus the dummy task w') that appears in a case in $C^+_{-W'}$, with the activities in the subsentence making up the tail of the rule. The head of the rule is then the disjunction of all tasks that immediately follow that set of activity in any case in $C^+_{-W'} \cup C^-_{W'_o}$.

Example 1 (Task Successor Rules). Let the set of cases be $\{PQR, PRS, RMN, TQV\}$, and let W' simply represent the execution of task R. Then $C^+ = \{PQR, PRS, RMN\}$, $C^- = \{TVQ\}$ and thus $C^+_{-W'} = \{PQw', PSw', MNw'\}$ and $C^-_{W'_o} = \{TVQw'_o\}$. The set of task successor rules is then:

$$\begin{array}{ll} P \rightarrow Q, S & Q \rightarrow w', w'_o \\ S \rightarrow w' & M \rightarrow N \\ N \rightarrow w' & PQ \rightarrow w' \\ PS \rightarrow w' & MN \rightarrow w' \end{array}$$

Once the rules are constructed, they are converted to clauses in conjunctive normal form. Binary resolution is then applied to find new clauses where w' is the only positive literal. Atoms from the negative literals must then necessarily imply w', where w' represents a trace of activity from W'. Thus the tasks represented by these atoms make up an abductive explanation. One could also make use of other mechanisms, such as assumption-based truth maintenance systems, to generate the abductive explanations.

Example 2 (Abductive Traces). Given the task successor rules generated in Example 1, one could perform the following sequence of resolutions and conclude that the presence of task P implies W':

$$\begin{array}{llll} \bar{P}QS, & \bar{P}\bar{Q}w' & \rightarrow & \bar{P}Sw' \\ \bar{P}Sw', & \bar{S}w' & \rightarrow & \bar{P}w' \end{array}$$

The set of abductive traces that would ultimately be found for this example is $\{P, S, M, N, PS, MN, PQ\}$. Thus the observation of any of these seven sequences implies that R will be performed. Note that the inclusion of w'_o in $C^-_{W'_o}$ (and subsequently in the task successor rule $Q \rightarrow w', w'_o$) ensures that Q is not chosen as an abductive explanation.

Even from a small example such as this, we see that an unnecessarily large number of explanations can be generated. As mentioned above, it is desirable to obtain abductive traces that are task-minimal. That is, it is more simplified to present either M or N as abductive explanations of R, rather than specify that both M and N are needed. Moreover, it also quite desirable to obtain workflow representations that depict only enough information that will minimally but completely explain W', and are thus complete and trace-minimal. In the above example, both P and S are explanations for R; however, it would be redundant to report both explanations, since P would explain the appearance of R in any case that S would.

Example 3 (Task-Minimal Traces). The set of abductive traces from Example 2 that are task-minimal is $\{P, S, M, N\}$. This is obtained by simply removing the supersets.

Example 4 (Complete Workflows). Using only the task-minimal traces in Example 3, there are six complete abductive workflows. Since each trace in this example consists of only a single event, each abductive workflow representation will consist of a number of possible single-event sequences, and thus are denoted here simply as a list of these events. The complete workflows are PM, PN, PMN, $PSM, PSN, PSMN$. This means that, if event R occurs, each of the six complete workflows will contain an explanation that occurred (i.e. either P or M will have occurred, and either P or N will have occurred, and either P or M or N will have occurred, etc.)

Example 5 (Trace-Minimal Workflows). Of the six complete workflows discussed in Example 4, only PM and PN are trace-minimal.

One can see that workflows that are complete, task-minimal and trace-minimal are quite desirable, not only because they are very concise and thus can be verified easily, but also because one can be assured that the workflow will necessarily contain an explanation that is triggering the critical behaviour. In the examples above, if the workflow PM was chosen, a compliance officer observing that activity R took place could easily check to see whether either P or M was executed. If not, one could conclude that there was not a sufficient motivation for R to occur, and could choose to investigate the matter further.

5 Results

As a first step in the investigation of abductive workflow mining, we perform a few simple tests to get an idea of how the general idea can work by reducing the size of the workflow representation. To accomplish this, we employ a naive method for finding abductive workflows where we simply search the set of cases for strings of tasks that imply W'. This will help to accomplish the main goal of the paper, which is to investigate the potential of abductive mining in terms of its effectiveness in simplifying complex workflows.

We start by building a workflow model W for the entire set C. W is then used to discover traces in the model that correspond to a path through W'. An activity a_i that directly precedes this path and an activity a_o that directly follows it are then chosen to form a larger path that contains the path through W'. We then search the rest of the model to determine if there is a sequence that contains a_i and a_o, with activity not consistent with W' in between. If one is found, then a new a_i or a_o is selected, or perhaps a longer sequence of preceding or following activities is selected. If this is not the case, however, then executing a_i and a_o necessarily causes activity in W' to be performed in between, and thus this is an abductive trace. This process continues until a complete abductive model is found.

To get an idea of by how much the size of the workflow graph is reduced by simply considering the abductive model, we ran tests on five example log files that are packaged with the ProM process management software [5]. We used our own miner to build the workflow, and then we chose a single task to be the target and found the corresponding abductive model that explains the target. The average number of transitions and arcs produced for each of the original and abductive model are presented in Table 1. The data shows that, in these tests the abductive model was less than one quarter the size of the original in terms of the number of both transitions and arcs.

	Original Workflow	Abductive Workflow
Number of transitions	156.2	37.2
Number of arcs	318.6	77.0

Table 1. Average size of the original and abductive workflow models for our workflow miner

To show that the abductive model can score a significant reduction on workflows for other miners, we ran the tests on the α miner as well, a particularly tough one to reduce, since very few transitions are used in α-algorithm-mined workflows. The results for these tests are depicted in Table 2. The data shows that significant reduction took place, as the workflows were cut to less than half the size.

	Original Workflow	Abductive Workflow
Number of transitions	12.0	5.6
Number of arcs	33.2	15.6

Table 2. Average size of the original and abductive workflow models for the α miner

6 Conclusions and Future Work

In this paper we discuss a new approach to modeling workflow, referred to as abductive workflow mining. With this approach, small concise workflows can be modeled that are found to necessarily cause some activity in question to occur. This is especially useful in the area of compliance checking where there is a small number of critical tasks that need to be checked. The resulting smaller workflow means that errors are less likely, and can also be more easily understood by a human analyst that might be trying to make sense of what went wrong. We employ a rule mining approach that searches the log in an effort to construct task successor rules, which indicate which tasks follow which other tasks in the underlying process. These rules, which are initially represented in Kowalski form, are then converted to CNF, and binary resolution is used to determine

the abductive explanations for the given critical activity. Tests on a very naive method for discovering abductive workflows show that the size of workflows can be significantly reduced.

While the technique performed well on a simple example presented in the paper, it is not guaranteed to find all abductive explanations. First, consider for example the transformed sets $C^{+}_{-W'} = \{PQw', QPw'\}$ and $C^{-}_{W'_o} = \{QPXw'_o\}$. Writing the clauses by preserving the order of tasks, we can see that P being executed before Q should necessarily imply W'. However, when considering these as clauses in binary resolution, PQ is treated no different from QP. And since QP does not necessarily imply W', no abductive explanations will be found. A second example of the technique's shortcomings arises in the transformed sets $C^{+}_{-W'} = \{PQRw', SQTw'\}$ and $C^{-}_{W'_o} = \{VTWw'_o\}$. Clearly, Q is an explanation for W'. However, the way the rules are written, Q is only specified to imply that R or T will be observed, and the observation of T does not imply W'. So Q will not be deemed an explanation by the resolution engine. Clearly there is much to be done to ensure that more (or all) explanations can be found. Accomplishing this task is the major focus of future work.

References

1. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow mining: A survey of issues and approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
2. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
3. James L. Peterson. Petri nets. *ACM Comput. Surv.*, 9(3):223–252, 1977.
4. D. Poole, A. Mackworth, and R. Goebel. *Computational Intelligence: A Logical Approach*. Oxford University Press, Inc., New York NY, USA, 1998.
5. ProM. The ProM framework. "http://is.tm.tue.nl/∼cgunther/dev/prom/", 2007.
6. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
7. Wil M.P. van der Aalst, V. Rubin, B.F. van Dongen1, E. Kindler, and C.W. Gunther1. Process mining: A two-step approach using transition systems and regions. Technical report, Eindhoven University of Technology, 2006.
8. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow mining: Discovering process mining models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
9. B.F. van Dongen, N. Busi, G.M. Pinna, and W.M.P. van der Aalst. An iterative algorithm for applying the theory of regions in process mining. Technical report, Department of Technology Management, Eindhoven University of Technology, 2006.
10. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
11. L. Wen, J. Wang, and J.G. Sun. Detecting implicit dependencies between tasks from event logs. In *Frontiers of WWW Research and Development - APWeb 2006*, volume 3841, pages 591–603. Springer Berlin / Heidelberg, 2006.