Supplemental Material: Optimising Event Pattern Matching using Business Process Models

Matthias Weidlich, *Member, IEEE,* Holger Ziekow, Avigdor Gal, *Senior Member, IEEE,* Jan Mendling, Mathias Weske, *Member, IEEE*

1 Overview

This report comprises supplemental material complementing the article *'Optimising Event Pattern Matching using Business Process Models'* [1] published in the IEEE Transactions on Knowledge and Data Engineering (TKDE).

First, Section 2 provides the formalization of rules that have not been discussed in detail in the submission. Second, Section 3 focuses on the formal properties of correctness, efficiency, and completeness of the proposed optimisation rules. Those rules relate to pattern transformation, plan selection, and plan transformation. Third, Section 4 provides details on additional experiments that we conducted as part of our evaluation. First, we present plots showing the savings achieved with rules for plan selection (rule 10) and plan transformation (rules 12 and 13) relative to the size of the processes. Second, we report on the results from an additional experiment with a dataset from a different domain, i.e., a publicly available event log that contains recorded execution sequences of a paper reviewing process. In the spirit of the experiments outlined in the submission, we have been able to determine how frequently the presented rules can be applied and measured the effects of the optimisation also for this dataset.

2 Rule Formalization

The submission does not include the formalisation for all rules. For rules for pattern transformation and plan selection that have not been discussed in the detail in the submission, the formalization is given in this section.

2.1 Pattern Transformation

Rule 2 adapts rule 1 for the reversed ordering of events in which event types E_1 and E_2 are related by reverse strict order. Hence, we have to change the order of event types of the elementary pattern, for which the operator type is updated.

Rule 2 :
for $(\mathcal{O}, \chi)$
if $\exists\ O = ((O_1, \ldots, O_n), \delta(O)) \in \mathcal{O}, \delta(O) = all$
and $\forall\ E_1, E_2 \in \mathbb{E}_P, E_1 \in \chi^*(O_i), E_2 \in \chi^*(O_j), 1 \leq i < j \leq n :$
$E_1 \rightsquigarrow^{-1} E_2$
then $(\mathcal{O} \cup \{O'\} \setminus \{O\}, \chi')$
$O' = ((O_n, \ldots, O_1), seq)$
$\chi' = \{(E, E') \in \chi \mid E' \neq O\} \cup \{(\rho(O), O')\}$

Rule 4 is similar to rule 3 and considers combinations of events that cannot occur. In contrast to rule 3, however, it exploits exclusiveness of event types. All compound event types that aim at the detection of a joint occurrence of events, may it be unordered (all) or ordered (seq), can be falsified if it holds for at least two of the referenced primitive event types: they are exclusive and respective events of those types are part of all sets in the matching set, i.e., the types are referenced by all and seq operators only.

Rule 4 :
for $(\mathcal{O}, \chi)$
if $\exists\ O = ((O_1, \ldots, O_n), \delta(O)) \in \mathcal{O}, \delta(O) \in \{all, seq\} \wedge$
and $\exists\ E_1, E_2 \in \mathbb{E}_P, E_1 \in \chi^*(O_i), E_2 \in \chi^*(O_j), 1 \leq i, j \leq n :$
$\forall\ O' \in \chi^*(O_i) \setminus \mathbb{E}_P, E_1 \in \chi^*(O') : \delta(O') \in \{all, seq\} \wedge$
$\forall\ O' \in \chi^*(O_j) \setminus \mathbb{E}_P, E_2 \in \chi^*(O') : \delta(O') \in \{all, seq\} \wedge$
$E_1 + E_2$
then $(\mathcal{O}', \chi')$
$\mathcal{O}' = \mathcal{O} \cup \{\perp\} \setminus \{O\} \setminus \{O' \in \mathcal{O} \mid \forall\ (E, E') \in \chi' : O' \neq E'\}$
$\chi' = \{(E, E') \in \chi \mid E \notin \chi^*(O)\} \cup \{(\rho(O), \perp)\}$

Rule 5 in the submission showed how a pattern may be transformed based on knowledge about order constraints about an event type that is part of the pattern definition and one event type that is independent of the pattern but still referenced by another pattern in the same pattern EPA definition.

We now consider the case of having a primitive event type E_1 used to define an any pattern ($E1$ is mandatory for one of the event types E of the any pattern), which is part of the definition of an all or seq pattern. Assume that there is a primitive event type E_2 that is mandatory for the latter pattern, but is exclusive to E_1. Then, we falsify the respective alternative, i.e., event type E, of the any pattern since matches of E will never contribute to matches of the complete pattern EPA.

Rule 6 :
for $(\mathcal{O}, \chi)$
if $\exists\ O = ((O_1, \ldots, O_n), \delta(O)) \in \mathcal{O}, \delta(O) = any \wedge$
$\exists\ O' = ((O'_1, \ldots, O'_m), \delta(O')) \in \mathcal{O}, \delta(O') \in \{all, seq\},$
$O \in \chi^*(O'_i), 1 \leq i \leq m$
and $\exists\ E_1, E_2 \in \mathbb{E}_P, E_1 \in \chi^*(O_j), E_2 \in \chi^*(O'_k),$
$1 \leq j \leq n, 1 \leq k \leq m :$
$\forall\ O'' \in \chi^*(O_j) \setminus \mathbb{E}_P, E_1 \in \chi^*(O'') : \delta(O'') \in \{all, seq\} \wedge$
$\forall\ O'' \in \chi^*(O'_k) \setminus \mathbb{E}_P, E_2 \in \chi^*(O'') : \delta(O'') \in \{all, seq\} \wedge$
$E_1 + E_2$
then $(\mathcal{O}', \chi')$
$\mathcal{O}' = \mathcal{O} \cup \{\perp\} \setminus \{O_j\} \setminus \{\hat{O} \in \mathcal{O} \mid \forall\ (E, E') \in \chi' : \hat{O} \neq E'\}$
$\chi' = \{(E, E') \in \chi \mid E \notin \chi^*(O_j)\} \cup \{(\rho(O_j), \perp)\}$

Rule 7 of the submission implemented pruning of falsified event types for the case of the all and seq operator. The following rules handle falsified event types for any patterns. The first prunes the falsified event type from the definition of an any pattern. The latter prunes complete any patterns that are defined over a single falsified event type. The first rule is defined for syntactical reasons: falsified event types are removed from a pattern EPA definition. Hence, the first may enable the second rule, but only the second one improves efficiency of event processing.

Rule 8 :
for $(\mathcal{O}, \chi)$
if $\exists\, O = ((O_1, \ldots, O_n), \delta(O)) \in \mathcal{O}, \delta(O) = any,$
$O_i = \bot, 1 \leq i \leq n$
then $(\mathcal{O}' \cup \{O'\} \setminus \{O\}, \chi')$
$O' = ((O_1, \ldots, O_{i-1}, O_{i+1}, \ldots, O_n), \delta(O))$
$\chi' = \{(E, E') \in \chi \mid E' \neq O\} \cup \{(\rho(O), O')\}$

Rule 9 :
for $(\mathcal{O}, \chi)$
if $\exists\, O = ((O_1), \delta(O)) \in \mathcal{O}, \delta(O) = any, O_1 = \bot$
then $(\mathcal{O}', \chi')$
$\mathcal{O}' = \mathcal{O} \setminus \{O\} \setminus \{O'' \in \mathcal{O} \mid \forall\, (E, E') \in \chi' : O'' \neq E'\}$
$\chi' = \{(E, E') \in \chi \mid E \notin \chi^*(O)\} \cup \{(\rho(O), \bot)\}$

2.2 Plan Selection

The mirrored case of rule 10 is captured as follows. Co-occurrence of the first event type (or set of event types) of a seq operator followed by the second event type (or set of event types) is used to guide the plan selection towards a push-strategy. Consider two event types E_1 and E_2, such that each event of type E_1 matches an event of type E_2, but not vice versa. Events of type E_2 may occur more often than events of type E_1. Pushing E_1 allows to efficiently filter out events of type E_2 – those for which there is no preceding event of type E_1.

Rule 11 :
for $(\mathcal{O}, \chi)$
if $\exists\, O = ((O_1, \ldots, O_n), \delta(O)) \in \mathcal{O}, \delta(O) = seq,$
and $\forall\, E_1, E_2 \in \mathbb{E}_P, E_1 \in \chi^*(O_i), E_2 \in \chi^*(O_{i+1}), 1 \leq i \leq n :$
$E_1 \gg E_2 \wedge E_2 \not\gg E_1$
then $O_i \rightarrow O_{i+1}$

3 Formal Properties

Below we list the proofs for correctness and efficiency of the optimisation rules. Also, if possible, we present results on their completeness given the model for event processing outlined in the submission.

3.1 Pattern Transformation

Proposition 1: We have the following results for correctness of the optimisation rules:
(1) Rule 1 is correct.
(2) Rule 2 is correct.
(3) Rule 3 is correct.
(4) Rule 4 is correct.
(5) Rule 5 is correct.
(6) Rule 6 is correct.
(7) Rule 7 is correct.
(8) Rule 8 is correct.
(9) Rule 9 is correct.

Proof:
(1) Assume that only the original pattern EPA Q, but not the rewritten one Q' matches $(e_1, \ldots, e_n) \in \mathcal{T}$. Then, there exist event types $O, E_1, E_2 \in \mathcal{E}$ with $O = ((O_1, \ldots, O_m), all)$, such that $E_1 = e_l.type \in \chi^*(O_i)$ and $E_2 = e_k.type \in \chi^*(C_j)$ for some $1 \leq i < j \leq m$ and $1 \leq k < l \leq n$. We arrive at a contradiction, since $E_1 \rightsquigarrow E_2$ implies $l < k$. Further, it trivially follows from the definition of all and seq that any trace matching Q' is matched by Q.
(2) The proof is obtained by mirroring the proof of case (1).
(3) Assume that the original pattern EPA Q comprises an event type $O = ((O_1, \ldots, O_m), seq)$ that qualifies for falsification, but matches $(e_1, \ldots, e_n) \in \mathcal{T}$. Then, there exist event types $E_1, E_2 \in \mathcal{E}$, such that $E_1 = e_k.type \in \chi^*(O_i)$ and $E_2 = e_l.type \in \chi^*(O_j)$ for some $1 \leq i < j \leq m$ and $1 \leq k < l \leq n$. $E_1 \rightsquigarrow^{-1} E_2$ implies $l < k$; a contradiction. Since $\bot$ does not match any event, any trace matching the rewritten EPA is matched by Q as well.
(4) The proof is obtained by mirroring the proof of case (3).
(5) The proof follows the one of case (3).
(6) The proof follows the one of case (3).
(7) Follows directly from the semantics of event type $\bot$ and the operators all and seq.
(8) Follows directly from the semantics of event type $\bot$ and the operators all and seq.
(9) Follows directly from the semantics of event type $\bot$ and the operators all and seq.

□

Proposition 2: We have the following results for efficiency of the optimisation rules:
(1) Rule 1 is improves pattern processing.
(2) Rule 2 is improves pattern processing.
(3) Rule 3 is improves pattern processing.
(4) Rule 4 is improves pattern processing.
(5) Rule 5 is improves pattern processing.
(6) Rule 6 is improves pattern processing.
(7) Rule 7 is improves pattern processing.
(8) Rule 9 is improves pattern processing.

Proof:
(1) Consider a pattern EPA Q that comprises a compound event type $O = ((O_1, \ldots, O_n), \delta(O))$. Since the plans for $\delta(O) = seq$ are always a subset of the plans for $\delta(O) = all$, the plans for $\delta(O) = all$ cannot show less plan instantiations. With $\delta(O) = all$ and two event types O_i and O_j, $1 \leq i < j \leq n$, either the execution plan $O_j \rightarrow O_i$ or the plan $O_j \rightarrow_{pull} O_i$ is instantiated for the event sequence (e) with $e.type = O_j$. With $\delta(O) = seq$, no plan is instantiated.
(2) The proof is obtained by mirroring the proof of case (1).
(3) Consider a pattern EPA Q that comprises a compound event type $O = ((O_1, \ldots, O_n), \delta(O))$ and two event types $E_1, E_2 \in \mathcal{E}$, such that $E_1 = e_k.type \in \chi^*(O_i)$ and $E_2 = e_l.type \in \chi^*(O_j)$ for some $1 \leq i < j \leq n$. The trace $(e) \in \mathcal{T}$ with $e.type = E_2$ leads to the instantiation of plan $E_2 \rightarrow E_1$ or plan $E_2 \rightarrow_{pull} E_1$, whereas no plan is instantiated for $\bot$.

TABLE 1: Overview of pattern transformations

Pattern Operator	Process Knowledge: Strict Order	Rev. Strict Order	Exclusiveness	Interleaving
Rules for event types of a single elementary pattern				
all	seqs (Rule 1)	seqs (Rule 2)	faln (Rule 4)	—
seq	faln (Rule 3)	faln (Rule 3)	faln (Rule 4)	—
any	—	—	—	—
Rules for event types of independent elementary patterns				
all	—	—	—	—
seq	—	—	—	—
any	faln (Rule 5)	faln (Rule 5)	faln (Rule 6)	—
Rules for pruning falsified event types				
all	prn (Rule 7)			
seq	prn (Rule 7)			
any	prn (Rules 8/9)			

(4) The proof is obtained by mirroring the proof of case (3).
(5) The proof follows the one of case (3).
(6) The proof follows the one of case (3).
(7) Follows directly from the removal of O from $\mathcal{O}$.
(8) Follows directly from the removal of O from $\mathcal{O}$.

□

Having discussed correctness and efficiency for the optimisation rules that relate to pattern transformation, we also consider their completeness, which gives rise to the following Theorem. For further reference, we also include the overview of pattern transformations as shown in Table 1.

Theorem 1: Rules 1-9 are correct, improve pattern processing, and are complete for all, seq, and any, under $\mathcal{B}_P$.

Proof: Correctness and efficiency of rules 1-9 follow from Proposition 1 and 2. Hence, we focus on completeness below. A pair (E_1, E_2) of event types is in exactly one of the relations in $\mathcal{B}_P$ [2]. Further, optimisation of an elementary pattern may be done in case only one of the event types E_1 and E_2, or both of them are required to (directly or indirectly) define a compound event type O. Since only local transformation rules are considered, it suffices to prove completeness for each of the eight cases (four order relations and either both event types, E_1 and E_2, or one of them is needed to define O) separately. Let $(\mathcal{O}, \chi)$ be a pattern EPA.

Consider the first case in the first row of Table 1, i.e., an elementary pattern $O = ((O_1, \ldots, O_n), \delta(O)) \in \mathcal{O}$, such that $\delta(O) = all$ and $E_1 \rightsquigarrow E_2$ for some $E_1 \in \chi^*(O_i)$, $E_2 \in \chi^*(O_j)$ with $1 \leq i < j \leq n$. Now, consider all possible transformation operations: changing the operator type to any could change the matching set of O. Changing it to seq is possible only if the respective order is observed for all event types E_1 and E_2 needed to define O. This case is captured by rule 1. Changing one of the event types, E_1 or E_2, or the whole pattern to $\perp$ is not possible since $E_1 \rightsquigarrow E_2$ witnesses that there exists a trace containing an event of type E_1 before an event of type E_2.

The same reasoning applies for all cases in which E_1 and E_2 are needed to define O (first three rows of Table 1). Then, completeness follows from the following observation: the application of rules 1 and 2 for a pattern O does not influence the applicability of any rule to for event types $E \in \mathcal{E}(O)$ and event types E with $O \in \chi^*(E)$. Application of rule 3 and 4 may disable the application of the other rules by introducing $\perp$. Though, maximal optimisation is ensured by rule 7 that propagates $\perp$.

Consider the first case in the fourth row of Table 1, i.e., an elementary pattern $O = ((O_1, \ldots, O_n), \delta(O)) \in \mathcal{O}$, such that $\delta(O) = all$ and $E_1 \rightsquigarrow E_2$ for some $E_1 \in \chi^*(O_j)$ with $1 \leq j \leq n$ and for all $1 \leq k \leq n$ it holds $E_2 \notin \chi^*(O_k)$. Again, consider all possible transformation operations: changing the operator type to any or seq could change the matching set of O. Neither of the event types O_j, $1 \leq j \leq n$, nor the whole pattern can be changed to $\perp$ since matching of O is independent of matching E_2.

The same reasoning applies for all cases in which E_1, but not E_2 is required to define O and the operator type is all or seq. Now consider $\delta(O) = any$ and $E_1 \rightsquigarrow E_2$. Let $O' = ((O'_1, \ldots, O'_m), \delta(O')) \in \mathcal{O}$, such that $O \in \chi^*(O'_i)$, $1 \leq i \leq m$, and $E_2 \in \chi^*(O'_k)$ with $1 \leq k \leq m$. Again, changing the operator type to all or seq could change the matching set of O. Event type O_j can be changed to $\perp$, only if events of type E_1 are in each matching set of O, events of type E_2 are in each matching set of O', but it holds $k < i$. This case is captured by rule 6. The whole pattern can be changed to $\perp$, only if all O_j have been changed to $\perp$. This is captured by rules 8 and 9. The same reasoning applies for all cases in which E_1, but not E_2 is required to define O and the operator type is any. □

3.2 Plan Selection

Rules for plan selection choose among valid plans and, therefore, correctness is not an issue here.

Proposition 3: We have the following results for efficiency of the optimisation rules:
(1) Rule 10 improves pattern processing.
(2) Rule 11 is improves pattern processing.

Proof:
(1) Let $O = ((O_1, \ldots, O_n), seq)$ be an event type and for all $E_1, E_2 \in \mathcal{E}_P$ with $E_1 \in \chi^*(O_i)$ and $E_2 \in \chi^*(O_j)$ for some $1 \leq i < j \leq n$ it holds $E_1 \not\gg E_2 \wedge E_2 \gg E_1$. Since $E_2 \gg E_1$, for every instantiation of plan $O_j \rightarrow pull\ O_i$ for some trace, there is also an instantiation of plan $O_i \rightarrow O_j$. Since $E_1 \not\gg E_2$, however, there may be a trace $(e) \in \mathcal{T}$ with $e.type = E_1$ that is not matched eventually. Hence, there are instantiations of plan $O_i \rightarrow O_j$ for which there are no instantiations of plan $O_j \rightarrow pull\ O_i$.
(2) The proof is obtained by mirroring the proof of case (1).

□

Turning to completeness of our optimisation rules with respect to plan selection, we obtain the following result. Again, we include the overview of plan selection rules, see Table 2 to discuss completeness.

Theorem 2: Rules 10-11 improve pattern processing and are complete for all, seq, and any under $\mathcal{B}_P^+$.

Proof: Since plans are generated for all primitive and compound event types in isolation, it suffices to consider the cases not addressed according to Table 2. If different event types show co-occurrence in either direction, pull-based plans and push-based plans are instantiated with the same frequency. If co-occurrence is not observed at all, no conclusion on the

TABLE 2: Overview of plan selections

Pattern Operator	Process Knowledge for Event Types E_1 and E_2 Co-occur. either	No co-occur.	Co-occur. E_2 to E_1	Co-occur. E_1 to E_2
all	—	—	—	—
seq	—	—	Pull (Rule 10)	Push (Rule 11)
any	—	—	—	—

number of occurrences of the events of both types can be drawn. For the all operator, plans (push- or pull-based) that are instantiated upon the arrival of either event are needed to guarantee correct execution. In case of the any operator, separate execution plans are instantiated for all event types of the definition of the compound event type. □

3.3 Plan Transformation

Proposition 4: We have the following results for correctness of the optimisation rules:
(1) Rule 12 is correct.
(2) Rule 13 is correct.

Proof:

(1) Assume that only the original plan $O_1 \rightarrow O_2$, but not the rewritten plan $O_1 \rightarrow \neg X \rightarrow O_2$ matches $(e_1, \ldots, e_n) \in \mathcal{T}$. Let $E_1, E_2 \in \mathcal{E}_P$ be event types such that $E_1 = e_i.type \in \chi^*(O_1)$, $X + E_2$, and $E_2 = e_j.type \in \chi^*(O_2)$ for some $1 \leq i < j \leq n$. Since the rewritten plan does not match, there exists an event e_k with $e_k.type = X$ and $i < k < j$, a contradiction with $X + E_2$.
(2) Assume that only the original plan $O_1 \rightarrow O_2$, but not the rewritten plan $X \rightarrow \ pull\ O_1 \rightarrow O_2$ matches $(e_1, \ldots, e_n) \in \mathcal{T}$. Let $E_1, E_2 \in \mathcal{E}_P$ be event types such that $E_1 = e_i.type \in \chi^*(O_1)$, $X+E_2$, and $E_2 = e_j.type \in \chi^*(O_2)$ for some $1 \leq i < j \leq n$. Since the rewritten plan does not match, there exists no event e_k with $e_k.type = X$ for $1 \leq k \leq n$. Since $O_1 \rightarrow O_2$ matches $(e_1, \ldots, e_n)$, this is a contradiction with $E_1 \gg X \wedge E_1 \rightsquigarrow X$ and $X \rightsquigarrow E_2$. □

Proposition 5: We have the following results for efficiency of the optimisation rules:
(1) Rule 12 improves memory consumption.
(2) Rule 13 improves memory consumption.

Proof:

(1) Consider a trace $(e_1, \ldots, e_n) \in \mathcal{T}$ containing an event e_i with $e_i.type = X$ for $1 \leq i < n$. For plan $O_1 \rightarrow O_2$, the memory consumption is reflected by the time $t_i = \sum_{1 \leq j \leq n, e_j.type \in \chi^*(O_1)} (e_n.t - e_j.t)$ that events stay in the system. For plan $O_1 \rightarrow \neg X \rightarrow O_2$, this time is $t'_i = \sum_{1 \leq j \leq n, e_j.type \in \chi^*(O_1)} (e_i.t - e_j.t)$. Since $i < n$, it holds $t'_i < t_i$. This holds for all traces showing an event e_i with $e_i.type = X$ for which either plans have been instantiated since $E_1 \rightsquigarrow X$ for all $E_1 \in \chi^*(O_1)$ implies that $e_j.type = E_1$ yields $j < i$.
(2) Consider a trace $(e_1, \ldots, e_n) \in \mathcal{T}$ containing an event e_i with $e_i.type = X$ for $1 \leq i < n$ that matches O_1, but not O_2. For plan $O_1 \rightarrow O_2$, events stay in the system for time $t_i = \sum_{1 \leq j \leq n, e_j.type \in \chi^*(O_1)} (e_n.t - e_j.t)$. For plan $X \rightarrow \ pull\ O_1 \rightarrow O_2$, this time is $t'_i = k \cdot (e_n.t - e_i.t)$ with k as the number of events e_j with $e_j.type \in \chi^*(O_1)$ for $1 \leq j \leq n$. Since $E_1 \rightsquigarrow X$ for $E_2 \in \chi^*(O_1)$, $e_j.type = E_1$ implies $j < i$ and, therefore, $e_j.t < e_i.t$. Hence, $t'_i < t_i$. □

4 Additional Evaluation Experiments

In this section, we report on further evaluation experiments. First, Section 4.1 discusses further results obtained for the industry case discussed in detail in the submission.

Second, in Section 4.2, we report on a series of experiments conducted for a process from a different domain. We took up a publicly available event log of a paper review process and also investigated applicability of our technique and the potential improvements for event pattern matching. As such, this experiment complements the industry case discussed in the submission by covering a different application domain and relying on a dataset that is publicly available and, thus, allows for reproducing the results.

4.1 Further Details on the Insurer Case

In the submission, we discussed the applicability of plan selection with rule 10 and plan transformation with rules 12 and 13 and also showed how applicability is related to the size of the processes. Further, we presented the distributions of the savings achieved with these rules. Here, we provide further insights on how these savings correlate with process size for the *full baseline patterns*.

For the three rules, Fig. 1 summarises the results. We observe that for rule 10, there appears to be no relation between the savings and the size of the processes. However, for the large majority of the processes, significant savings are achieved (45% on average).

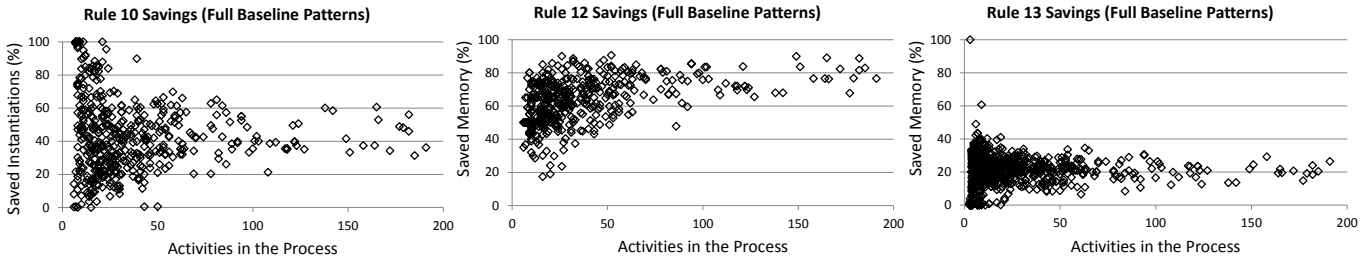
Plan transformation with rule 12 leads to increasing savings when processes get larger. For large processes (> 50 activities), we observe memory savings of at least 60% in virtually all cases. This highlights a large potential of this particular rule since for such large processes, optimisation is most required.

For plan transformation with rule 13, in turn, there is no obvious relation between the savings and the process size. However, the savings are still high, with 47% of the processes achieving savings over 20%. This is remarkable, since, again, we obtained these results using a worst case assumption regarding the strategy for adding an additional event type X to the execution plan.

4.2 Log of a Paper Review Process

Dataset. The event log used in this experiment relates to a paper review process. Both, the event log and a corresponding process model are publicly available[1] and have been used to demonstrate process mining techniques, cf., [3]. The model defines 20 activities. The log comprises 3730 events for 100 process instances. Each event is associated with a timestamp and a reference to an activity of the process model.

1. http://www.processmining.org/logs/start



(a) (b) (c)

Fig. 1: Savings obtained with rules 10, 12, and 13 relative to the size of the processes (full baseline patterns)

Setup. As discussed for the industry case in the submission, we focus on the potential applicability in terms of the exploitable process knowledge when evaluating rules for pattern transformation (rules 1 to 9).

For the evaluation of plan selection rules, again, we assume a push-based strategy by default. Hence, we focus on rule 10 for the evaluation. In the absence of a catalogue of event patterns, we rely on baseline queries that relate to sequential pairs of event types. Again, the motivation of these baseline queries stems from the domain of the process. That is, monitoring of the delay between the execution of two activities in the paper review process can be seen as an essential use case for event processing in this setting. As in the industry case, we extracted all event type pairs that can occur in sequence within the process and determined the pairs for which the rule is applicable. Further, we generated a baseline pattern and an optimized pattern for each pair where a rule applies. The comparison of the efficiency of both plans allows for judging the achieved savings.

Evaluation of plan transformation, namely rule 12 and rule 13, follows the approach outlined for plan selection, i.e., comparing a baseline pattern and an optimized pattern for all sequences of pairs of event types for which a rule is applicable.

Again, the analytic model introduced in the submission is used as a basis for quantifying processing efficiency. We therefore refer the reader to the discussion on how to quantify memory savings that can be found in the submission.

Results. Table 3 depicts the relation sizes of the behavioural profile obtained for the model of the paper review process.

TABLE 3: Relation Sizes (Review Process)

Strict Order	Rev. Strict Order	Exclusiveness	Interleaving
156 (39%)	156 (39%)	28 (7%)	60 (15%)

Only a small share of 15% of pairs of event types are interleaving, which precludes the possibility to apply any rule for pattern transformation. The large majority of pairs is related by one of the order relations, which may enable application of rules for sequentialization and falsification. Also, exclusiveness as observed for 7% of the pairs, allows for applying further falsification rules. Hence, the paper review process shows a large potential for optimization by pattern transformation.

The results for applicability and savings for plan selection and plan transformation rules are given in Table 4.

TABLE 4: Applicability and Savings (Review Process)

Rule	Applicability Over 156 Pairs	Savings Saved Instantiations or Saved Memory
Rule 10	76 (49%)	38%
Rule 12	52 (33%)	9%
Rule 13	27 (17%)	10%

Rule 10 is applicable for nearly every second execution plan. In comparison to the baseline pattern, 40% of plan instantiations are saved, which points at high effectiveness of this rule. Rule 12, early termination of execution plans, is applicable for a third of the event type pairs and yields memory savings of 9% in comparison to the baseline. For rule 13, which reduces memory consumption by considering additional event types X in the execution plan, the process can include more than one candidate for X. In our evaluation, we used a disjunction of all candidate events, which is the worst case assumption and yields the lower bound of achievable savings. We observe applicability for 17% of the pairs. In terms of memory savings, the effect of rule 13 is similar to the one observed for rule 12.

REFERENCES

[1] M. Weidlich, H. Ziekow, A. Gal, J. Mendling, and M. Weske, "Optimising Event Pattern Matching using Business Process Models," *IEEE Transactions on Knowledge and Data Engineering*, 2013. To appear.

[2] M. Weidlich, J. Mendling, and M. Weske, "Efficient consistency measurement based on behavioral profiles of process models," *IEEE Trans. Software Eng.*, vol. 37, no. 3, pp. 410–429, 2011.

[3] W. V. der Aalst, K. V. Hee, J. V. der Werf, and M. Verdonk, "Auditing 2.0: Using process mining to support tomorrow's auditor," *IEEE Computer*, vol. 43, no. 3, pp. 90–93, 2010.