Defining meaningful Local Process Models

Mitchel Brunings, Dirk Fahland, and Boudewijn van Dongen

Eindhoven University of Technology, Eindhoven, The Netherlands
{m.d.brunings, d.fahland, b.f.v.dongen}@tue.nl

Abstract. Current process discovery techniques are unable to produce high quality models that describe all observed behavior in semi-structured processes in a meaningful way. Local process model (LPM) discovery has been proposed to discover meaningful patterns in event logs from unstructured processes. In this paper, we explore the use of LPM discovery on event logs from semi-structured processes and find several drawbacks: it finds many small patterns but doesn't find patterns larger than 4-5 events, it produces too many models, and the discovered models describe some events from the log multiple times while leaving others unexplained. Despite these drawbacks, we observe that a set of LPMs taken together can yield interesting insights. From these observations we distill several requirements for meaningful sets of LPMs: we want (1) a limited set of models that (2) have high accuracy measures such as fitness and precision while (3) they together cover the whole event log and (4) do not cover parts of the log multiple times unnecessarily. We show that it is possible to manually construct sets of LPMs that satisfy all these requirements on the well-known BPIC12 event log. We then apply and evaluate the existing quality measures for individual LPMs. We propose to disregard support, confidence, and determinism as measures for meaningfulness of LPMs and we propose new ways to evaluate sets of LPMs based existing methods.

Keywords: Process discovery · Local Process Models · Coverage

1 Introduction

Semi-structured behavior exists plentiful in practice such as in hospitals and purchasing processes. An example of such behavior is captured in the log of BPIC12 [12]. Discovering process models from this log has proven to be a big challenge for traditional start-to-end process discovery techniques such as Inductive Miner [4] (produces fitting but imprecise models) and SplitMiner [1] (produces precise but non-fitting models). Other methods to model such behavior include Declare [5], DCR graphs [2], trace clustering [10], patterns [6], and Local Process Models (LPMs) [9].

Using declare to discover a model from semi-structured behavior results in constraint explosion. Declare is not suitable for otherwise imperative processes that can be described with flow-based techniques [3]. Clustering of traces cannot capture situations where trace variants share parts of behavior with other variants, so either they merge or we get cluster explosion [7]. Visual exploration using the Log Pattern Explorer [6] works well, but is a manual task which does not automatically result in models. So far, LPMs [9] have been applied on finding patterns in highly unstructured behavior (like

smarthome environments), but we are interested in finding models for slightly more structured processes.

In this paper, we explore what it takes to discover accurate process models from semi-structured behavior. We show what LPM discovery [9] can do on a reduced version of the BPIC12 log which contains more structured behavior than what LPM discovery was designed for, but is still not very structured. We find that the output consists of too many models which together cover less than two-thirds of the log, while covering nearly a third of the log multiple times. We show that to explain this log, we need only a few models instead of hundreds, our models describe the whole behavior instead of only a part, and we provide a high-level description of how our models coexist.

In Section 2 we introduce the fragment of the BPIC12 log that we use as running example and show that start-to-end mining techniques fall short, and then explain how LPMs work, and explore the LPMs that LPM discovery produces, from which we distill some requirements for meaningful sets of LPMs. In Section 3 we show a set of LPMs that we produced by hand, and explore how they coexist. In Section 4 we compare how existing quality measures for LPMs score our manually constructed models versus the automatically discovered LPMs. We then propose new measures that better reward models for describing more behavior, and measures that help build a set of models that do not overlap.

2 Process model discovery on semi-structured behavior

In this section, we introduce a running example of semi-structured behavior and apply some start-to-end model discovery techniques on this example and describe their shortcomings. We recall the idea of local process models (LPMs) and their purpose compared to start-to-end models in process discovery. We then apply existing LPM discovery techniques and discuss the quality of the resulting LPMs for describing semi-structured behavior in a meaningful way.

2.1 Semi-structured behavior

To show where current techniques fall short, we take a section of the log from the Business Process Intelligence Challenge from 2012 [12] (BPIC12). The BPIC12 log is a well-known log that comes from a loan application process in a bank. We filter this log to keep only events in which resource 10939 was involved and call this filtered log $L_{\text{BPIC12}}^{10939}$. We choose resource 10939 in particular to be the same as the resource chosen in [9] for easier comparisons with that work. However, in [9] they used the date as their case notion, instead of the usual loan application ID. We filter to the same events, but we keep using the loan application ID as our case notion.

A log with resource+caseID as case notion represents the behavior of an individual resource/actor in a process within each case. This shows how a resource interacts with a case. This particular view on the BPIC12 log makes $L_{\text{BPIC12}}^{10939}$ semi-structured: We look at the same person working along a case, but each case is different and they are not always involved in the same way in each case, as they are sharing work with other people. This makes $L_{\text{BPIC12}}^{10939}$ less structured than the full log in which all steps by all

people are recorded: A trace in our log may start and end anywhere in the lifetime of the full case, and there may be gaps.

$L^{10939}_{\text{BPIC12}}$ consists of 2763 events in 647 traces. There are 23 event classes and 84 trace variants. In Table 1 we count the event classes observed in $L^{10939}_{\text{BPIC12}}$. This table also serves as a legend for the shorthand (in parentheses) that we use for the event classes in this log.

Table 1: Event occurrence matrix for $L^{10939}_{\text{BPIC12}}$(with shorthand notation in parentheses)

		lifecycle:transition		
concept:name		SCHEDULE (0)	START (1)	COMPLETE (2)
A_ACCEPTED	(A_ACC)	–	–	104
A_CANCELLED	(A_CAN)	–	–	27
A_DECLINED	(A_DEC)	–	–	78
A_FINALIZED	(A_FIN)	–	–	104
A_PREACCEPTED	(A_PRE)	–	–	73
O_CANCELLED	(O_CAN)	–	–	34
O_CREATED	(O_CRE)	–	–	124
O_SELECTED	(O_SEL)	–	–	124
O_SENT	(O_SEN)	–	–	124
W_Afhandelen leads	(W_Afh)	–	154	154
W_Completeren aanvraag	(W_Com)	73	389	396
W_Nabellen incomplete dossiers	(W_Nid)	4	103	103
W_Nabellen offertes	(W_Nof)	120	227	235
W_Valideren aanvraag	(W_Val)	9	2	2

2.2 Discovering start-to-end process models from semi-structured behavior

Applying the Inductive Miner - infrequent [4] (at 20% threshold) on $L^{10939}_{\text{BPIC12}}$ results in the model shown in Fig. 1. The Inductive Miner produces a Petri net that allows behavior that we know does not exist in the process. For instance, it allows W_Afh+1 and W_Afh+2 to occur or be skipped independently of each other, while we know that every W_Afh+1 is (eventually) followed by W_Afh+2. Several other behavior patterns that are known from the BPIC12 log and also appear in $L^{10939}_{\text{BPIC12}}$ have been missed in a similar fashion.

Applying the SplitMiner (using standard settings) on a modified[1] $L^{10939}_{\text{BPIC12}}$ results in the model shown in Fig. 2. The SplitMiner produces a BPMN model that disallows behavior that we know exists. For instance, this model claims that W_Afh+1, ..., W_Afh+2 is never followed by W_Com+1, ..., W_Com+2. Several other behavior patterns that are known from the BPIC12 log and also appear in $L^{10939}_{\text{BPIC12}}$ have been missed in a similar fashion.

[1] The SplitMiner [1] doesn't explicitly consider lifecycle events, therefore we include the lifecycle data in the event name with the "Bring Lifecycle to Event Name" plugin in ProM 6.9.

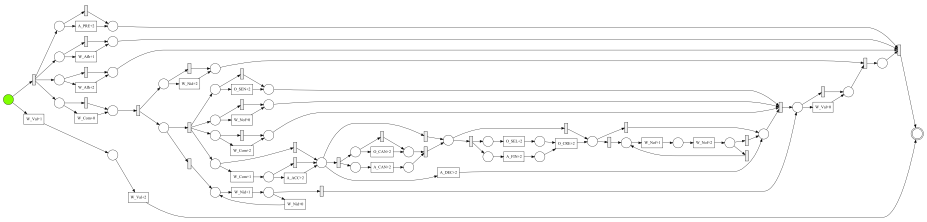


Fig. 1: The resulting Petri net of IMf [4] on $L^{10939}_{\text{BPIC12}}$

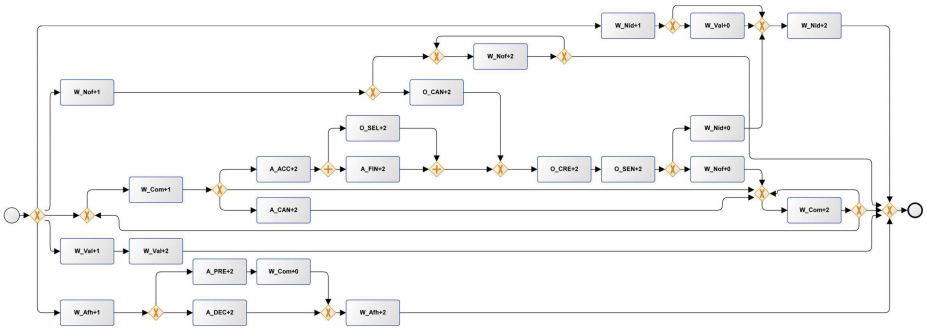


Fig. 2: The resulting Petri net of SplitMiner [1] on $L^{10939}_{\text{BPIC12}}$

These examples of state-of-the-art process discovery tools for start-to-end processes show that they are unable to deal with the structure of $L^{10939}_{\text{BPIC12}}$. They result in models that may have high fitness or high precision scores, but they never score well on both measures. These models are also rather complex, with Fig. 1 containing tau-skips for almost all transitions, and Fig. 2 containing many loops and several jumps between paths. We want models that are accurate (i.e. have high fitness and precision scores), as inaccurate models do not describe the process we are trying to understand. We also want models that are simple, because incomprehensible models will also not help our understanding of the processes they describe. The models in Fig. 1 and Fig. 2 show that start-to-end model discovery techniques are not able to discover models from semi-structured behavior that are both accurate and simple.

2.3 Local Process Models

Tax et al. [9] also show that existing process model discovery techniques are unable to deal with event logs that contain a lot of repetition within traces. They introduce *local process models* (LPMs) to describe patterns that are smaller than the full observed behavior, but can still capture more behavior than simple sequences, such as concurrency and choice.

As an example, they give the event log shown in Fig. 3a. They describe the log as events from a fictional sales department, where each trace describes the activities of a particular sales person on a particular day. A sales person may work on multiple cases in

a day, and multiple sales persons may be working on the same case. However, despite this chaotic behavior, a frequent pattern still emerges: When a sales person performs activity 'A', they often perform both activities 'B' and 'C' shortly after that on the same day.

Applying the Inductive Miner - infrequent [4] (at 20% threshold) on this example log produces the model shown in Fig. 3b, which allows nearly all behavior. Tax et al. show that there is a model that describes the frequent pattern from Fig. 3a in Fig. 3c. In this model we see that 13 times out of 21 total occurrences of 'A', it is followed by a 'B' and a 'C' in any order.

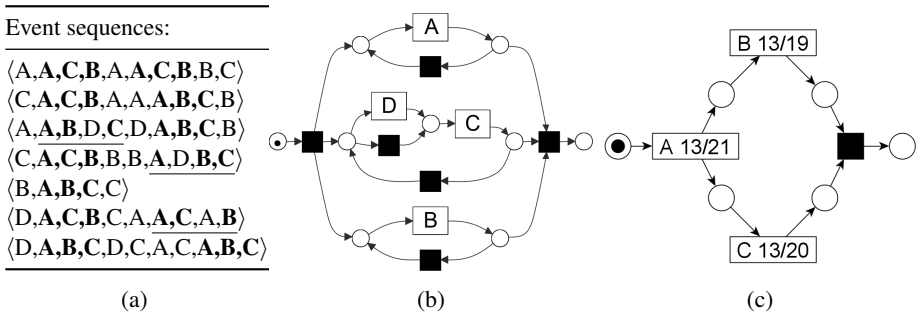


Fig. 3: (a) A log of event sequences with **highlighted** instances of a frequent pattern. Gapped instances are underlined. Source: [9]. (b) The resulting Petri net of IMf [4] on the event sequences shown in Fig. 3a. (c) An LPM showing the frequent behavior pattern from Fig. 3a. Source: [9].

Tax et al. managed to discover an LPM that describes a pattern that could not be discovered or displayed before. This suggests that LPMs are worth exploring as an alternative to traditional start-to-end model discovery and as an improvement on sequential pattern mining.

2.4 Discovering LPMs from semi-structured behavior

Applying "Search for Local Process Models" on a modified[2] $L^{10939}_{\text{BPIC12}}$ with all the default settings yields 100 LPMs divided into 13 groups. Each LPM has 2 to 4 labeled transitions and all LPMs together describe events of 13 event classes out of the 23 event classes that appear in the log. Table 2 shows which event classes occur in which group(s) of LPMs. In this table, we can also see that LPM groups 2 and 5-12 explain different combinations of the same 5 event classes.

[2] The "Search for Local Process Models" plugin in ProM 6.9 only looks at the event names, therefore we include the lifecycle data in the event name with the "Bring Lifecycle to Event Name" plugin.

Table 2: Event class occurrence for the 13 groups of LPMs

Group	1	2	3	4	5	6	7	8	9	10	11	12	13
W_Afh+1	x												
W_Afh+2	x												
O_SEL+2		x						x		x	x	x	
O_CRE+2		x				x	x	x	x	x		x	
O_SEN+2		x			x		x	x	x	x	x		
W_Nid+1			x										
W_Nid+2			x										
A_PRE+2				x									
W_Com+0				x									
A_FIN+2					x	x	x	x	x		x	x	
W_Com+1									x	x	x	x	
W_Nof+0													x
W_Com+2													x

In Fig. 4, we show the highest ranked LPM from 3 of the 13 groups of LPMs. The two LPMs on the right show significant overlap (highlighted), where they both explain all 104 occurrences of A_FIN in $L^{10939}_{\text{BPIC12}}$, and 104 out of 124 occurrences of both O_SEL and O_CRE, which implies an overlap of at least 84 occurrences each.

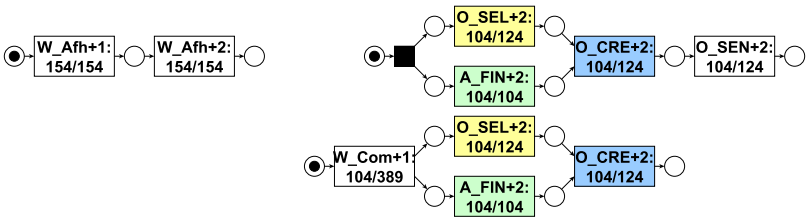


Fig. 4: A selection of LPMs discovered from $L^{10939}_{\text{BPIC12}}$ (same labels highlighted)

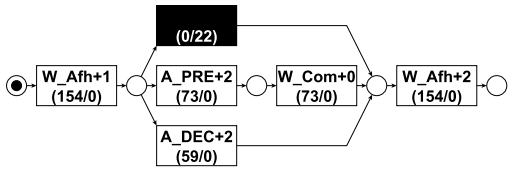
We conclude that these LPMs are not meaningful for three reasons: (1) We get too many of them. (2) They leave large parts of the log unexplained: the 10 missing event classes alone already represent 722 out of the 2763 events in $L^{10939}_{\text{BPIC12}}$. And (3) they explain some events multiple times: these events are described by multiple transitions in multiple different LPMs. We show in Section 4 as we evaluate "coverage" of events that this is indeed the case.

3 The potential of Local Process Models

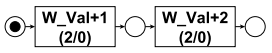
In Section 2 we introduced $L^{10939}_{\text{BPIC12}}$ and saw that there are no good process discovery techniques for this particular log. In this section, we show a manually created set of

LPMs to describe the behavior from this log. We then discuss why our set of LPMs is preferable over the models we saw in Section 2 both in terms of accuracy and understandability.

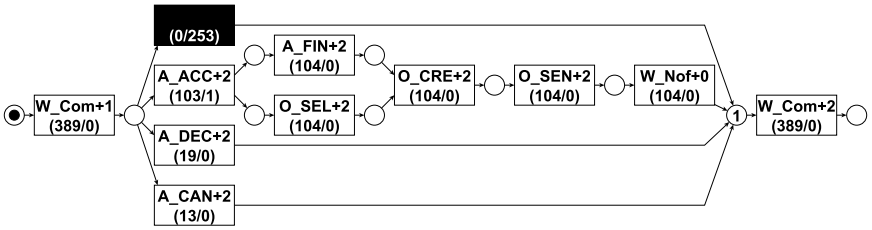
3.1 Local Process Models that could be



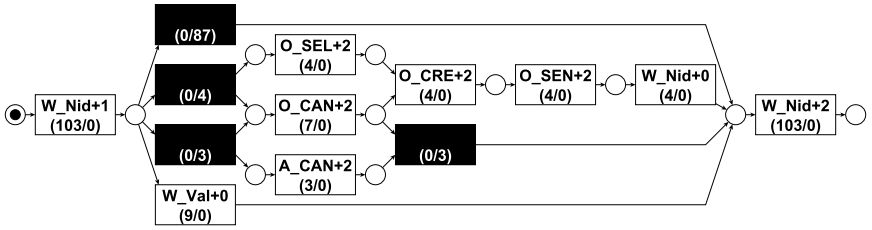
(a) Manual LPM 1 for $L^{10939}_{\text{BPIC12}}$



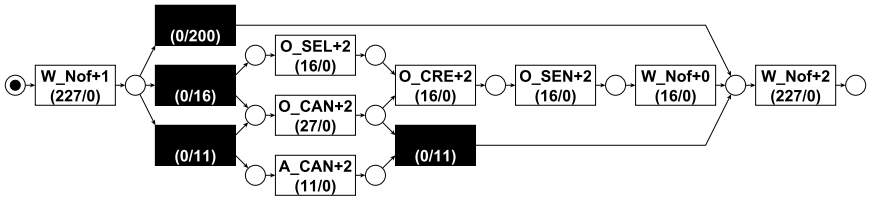
(b) Manual LPM 5 for $L^{10939}_{\text{BPIC12}}$



(c) Manual LPM 2 for $L^{10939}_{\text{BPIC12}}$



(d) Manual LPM 3 for $L^{10939}_{\text{BPIC12}}$



(e) Manual LPM 4 for $L^{10939}_{\text{BPIC12}}$

Fig. 5: Manually contructed set of LPMs for $L^{10939}_{\text{BPIC12}}$

In $L^{10939}_{\text{BPIC12}}$, all A_... and O_.... events occur between pairs of W_...+1 (start) and W_...+2 (complete) events and these start and complete event pairs are never nested. So, we

split the log into trace fragments that start with a W_...+1 event and end with their corresponding W_...+2 event and created a log for each start event class containing the fragments with the corresponding start events. We analyzed the 5 resulting logs and constructed the set of LPMs shown in Fig. 5 to model the behavior from each individual log. By aligning each of the logs to the corresponding LPM [11], we obtained the number of synchronous moves and model moves per transition as indicated in the figures. A single move on log is indicated with a number in its corresponding place.

Each of our LPMs represents a chunk of behavior carried out by resource 10939 per case, just like the LPMs in Section 2.4, but in contrast to those LPMs, 4 of our 5 LPMs are significantly larger, as we have found much larger patterns. All top LPMs (best of their group) from Section 2.4 that were found with the plugin by Tax et al. appear as parts of our larger LPMs. We also see that in our set of LPMs all event classes are represented at least once, and 7 event classes are represented in more than one LPM. Not only do our LPMs represent large chunks of observed behavior, but we also only need 5 of them to describe nearly all observed behavior. We explain this quality criterion we call *coverage* in the next section.

3.2 Coverage

We want a set of LPMs to explain as much of the behavior in the log as possible with minimal redundancy. To this end, we introduce the terms *coverage* and *duplicate coverage*. We first define coverage on an individual LPM as the fraction of events from the log that are actually explained by the given LPM. To measure this, we calculate the alignment of relevant trace fragments from the log on the LPM and then take the number of synchronous moves in the alignment and divide by the total number of events in the log. We say that an LPM *covers* an event if and only if this event is part of a synchronous move in the alignment. This differs from the coverage metric in [9] which measures the fraction of events from the log that *might* be explained by the LPM because the LPM has a transition with a matching label.

We define coverage for a set of LPMs as follows: We consider an event covered if there is at least one LPM in the set that covers it. The coverage is then computed by counting all covered events and dividing by the total number of events in the log. The result is the fraction of the observed behavior that is explained by the set of LPMs.

To calculate the coverage of our set of LPMs from Fig. 5 we can simply sum up the number of synchronous moves for all models and see that our 5 models together cover 2747 out of 2763 events recorded in the log. The sum is valid because we know that we did not duplicate any event because of the way we split $L^{10939}_{\text{BPIC12}}$.

At the end of Section 2.4 we state that all LPMs from that section together fail to explain at least 722 events because they do not have corresponding transitions, this means that those models together cover at most 2041 out of 2763 events from the log. From the perspective of coverage, our manually created set of LPMs is a major improvement.

To measure duplicate coverage of a set of LPMs, we count the number of events that are covered by more than one LPM and divide by the total number of events in the log. The result is the fraction of the observed behavior that is explained multiple times by the set of LPMs. By keeping this number low, we keep redundancy in our set of LPMs low.

Because we did not duplicate any event in splitting $L^{10939}_{\text{BPIC12}}$, we know that our manually constructed set of LPMs doesn't cover any event more than once. We showed in Section 2.4 that the models we discovered there had overlap resulting in significant duplicate coverage. Our manually constructed set of LPMs is therefore also an improvement from the perspective of duplicate coverage.

3.3 Requirements for meaningful sets of LPMs

The aim of this paper is not to introduce a new LPM discovery technique, but to share our vision on what a set of LPMs should look like. To more formally define our vision, we define the following requirements for meaningful sets of LPMs:

R1 The set of LPMs should consist of individual LPMs that are accurate, i.e. have high fitness and precision scores, because we want to describe the observed behavior and nothing else;
R2 The set of LPMs should be limited in size, because with too many models it will be hard to comprehend the set as a whole;
R3 The set of LPMs should maximize coverage, because we want to describe all observed behavior;
R4 The set of LPMs should minimize duplicate coverage, because we want to limit redundancy.

When we check our manually constructed set of LPMs against these requirements, we see that all requirements are satisfied. R2 is satisfied as we only have 5 LPMs, R3 is satisfied because only 17 out of 2763 events are not covered, and R4 is satisfied because no event is explained more than once. For R1 we show in Section 4 how to calculate accuracy measures for LPMs, and that these scores are indeed good for our set of LPMs.

In Section 4, we suggest quality measurement techniques based on existing measures in literature that measure the degree to which these requirements are met.

4 Evaluating quality of LPMs

Tax et al. [9] suggest a list of quality criteria (support, confidence, language fit, determinism, and coverage) to measure the quality of individual LPMs. In this section, we explain these measures and use them on both the LPMs discovered with the plugin by Tax et al. and on the LPMs we constructed manually. We then compare these results, and discuss for each measure if it is useful and why. We then introduce some new measures that we believe help us find meaningful sets of LPMs.

4.1 Quality metrics designed by Tax et al.

First, we provide a short description of each quality criterion developed by Tax et al. For the exact definition, we refer to [9].

The **support** of an LPM measures how often the pattern described by the LPM occurs (a.k.a. *frequency*) on a scale from 0 to 1.

The **confidence** of an LPM measures the likelihood that an event whose class appears as one of the transitions in the LPM is part of a pattern that this LPM describes on a scale from 0 to 1.

The **language fit** of an LPM is the ratio of the behavior that is allowed by the LPM that is observed in the log.

The **determinism** of an LPM measures how well the LPM can predict the next event of a fitting trace on a scale from 0 to 1.

The **coverage** of an LPM is the ratio of events in the log of types that occur in the LPM. (Note that this is not the same as the coverage measure we define in Section 3.2.)

The **score** on an LPM is a weighted average of its support, confidence, language fit, determinism, and coverage.

In tables 3 and 4 we show the scores on these measures for both the top discovered LPM of each group from Section 2.4 and the manual LPMs from Section 3.1. We use the "Rescore Local Process Model ranking to Log" plugin in ProM 6.9 to calculate these scores. Note that instead of reporting the support, this plugin reports the frequency of LPMs.

Table 3: The scores attained by the discovered LPMs from Section 2.4

group	score	frequency	confidence	determinism	language fit	coverage
1	0.869	154	1.000	1.000	1.000	0.111
2	0.868	124	1.000	1.000	1.000	0.135
3	0.867	103	1.000	1.000	1.000	0.075
4	0.865	73	1.000	1.000	1.000	0.053
5	0.832	104	0.912	1.000	1.000	0.083
6	0.832	104	0.912	1.000	1.000	0.083
7	0.821	104	0.886	1.000	1.000	0.127
8	0.756	104	0.874	0.800	1.000	0.172
9	0.691	104	0.561	1.000	1.000	0.268
10	0.686	104	0.547	1.000	1.000	0.275
11	0.631	104	0.561	0.800	1.000	0.268
12	0.631	104	0.561	0.800	1.000	0.268
13	0.628	104	0.403	1.000	1.000	0.187

Table 4: The scores attained by the manual LPMs from Section 3.1

model	score	frequency	confidence	determinism	language fit	coverage
1	0.742	154	0.939	0.660	1.000	0.193
2	0.661	388	0.670	0.737	1.000	0.575
3	0.446	103	0.081	0.876	0.833	0.236
4	0.544	227	0.219	0.954	1.000	0.367
5	0.823	2	1.000	1.000	1.000	0.001

Comparing these results, we see that the manual LPMs have lower scores than the discovered LPMs because the manual LPMs have lower confidence, determinism, and

language fit. The manual LPMs have lower confidence because they feature activities which only cover a fraction of the events of their class. This is by design, as the same activities may occur in different phases of the process, and thus occur in different LPMs. The manual LPMs have lower determinism, as we constructed larger LPMs with more choice than those discovered in Section 2.4. Finally, we score lower on language fit on a single LPM, because in that LPM we allow O_CAN+2 and A_CAN+2 to occur in parallel between W_Nid+1 and W_Nid+2, when there is no evidence for this in $L^{10939}_{\text{BPIC12}}$. However, we did observe 11 occurrences of O_CAN+2 and A_CAN+2 in parallel between W_Nof+1 and W_Nof+2, and with only 3 occurrences between W_Nid+1 and W_Nid+2, we assumed parallelism there as well. Though on $L^{10939}_{\text{BPIC12}}$we score lower on language fit because of this, when we check the original log from BPIC12, we see that these activities do indeed occur in parallel between W_Nid+1 and W_Nid+2.

The manual LPMs have higher frequency and coverage than the discovered LPMs, except for manual LPM 5 (Fig. 5b) which covers only 4 events. One could argue for its removal, but it explains 4 events perfectly that no other LPM does.

We observe that the manual LPMs have lower confidence and determinism by design and even frequency, language fit, and coverage don't seem to be important enough to accept or reject an LPM from a set. In the rest of this section, we explore measures that better evaluate sets of LPMs.

4.2 New quality measures for individual LPMs

A model should not leave large parts of the log unexplained. For traditional start-to-end models, there exist fitness measures that try to measure how much of a log is explained by a model. In this paper, we use the replay method explained in [11]. However, replaying $L^{10939}_{\text{BPIC12}}$ on the LPMs in this paper yields very low fitness scores for each LPM, as these LPMs do not represent the whole log by design. Instead, we limit our replays to relevant trace fragments for each LPM as follows.

As we have split $L^{10939}_{\text{BPIC12}}$ up by the W_...+1 and W_...+2 events to construct our manual LPMs, it makes sense to split the log the same way for their evaluation. Because we want to evaluate the automatically discovered LPMs using the same techniques, we need a way to find appropriate trace fragments from a log based only on a given LPM. To that end, we use the following method: Trace fragments should not include events for which there is no matching activity in the LPM, as these are clearly not events that the LPM is describing, so we filter out any such events. An LPM describes behavior that starts with its first activity, so we should make sure all of our trace fragments start with an event that matches the LPM's first activity. If an LPM starts with a tau AND-split, a trace fragment may start with any combination of its first labeled activities. Therefore, we start recording trace fragments when encountering such start events. For similar reasons, and using similar methods, we stop recording trace fragments when we encounter final events. Should we encounter a start event before we encountered a final event, we simply finish recording the previous trace fragment at that point, and start a new one. The resulting set of trace fragments can then be used as a log, which should have as many traces as there are occurrences of the first activity in the original log. This log of trace fragments can then be replayed on the LPM and the resulting alignment yields the fitness.

A model should not allow much more behavior than what is observed. For traditional start-to-end models, there exist precision measures that try to measure how much unobserved behavior is allowed by a model. In this paper, we use the escaping-edge based method explained in [8]. We calculate precision for an LPM using the same log of trace fragments and resulting alignment as we use for calculating fitness.

Applying these fitness and precision measuring techniques on the same sets of LPMs as used in Section 4.1 yields the results shown in tables 5a and 5b. These tables also include the coverage measured as described in Section 3.2.

Table 5: The scores attained by the LPMs from:

(a) Section 2.4

group	fitness	precision	coverage
1	1.000	1.000	0.111
2	1.000	1.000	0.135
3	1.000	1.000	0.075
4	1.000	1.000	0.053
5	1.000	1.000	0.075
6	1.000	1.000	0.075
7	1.000	1.000	0.113
8	0.986	1.000	0.172
9	0.662	1.000	0.254
10	0.662	1.000	0.254
11	0.662	1.000	0.254
12	0.662	1.000	0.254
13	0.970	1.000	0.081

(b) Section 3.1

model	fitness	precision	coverage
1	1.000	1.000	0.186
2	0.999	1.000	0.519
3	1.000	0.995	0.087
4	1.000	1.000	0.201
5	1.000	1.000	0.001

With these results, we conclude that the manual LPMs satisfy R1 from Section 3.3. We also see that some of the discovered LPMs have low fitness, meaning they don't satisfy R1.

4.3 Quality measures for sets of LPMs

For R2 it is easy to see that we have far fewer manual LPMs than automatically discovered LPMs. Clearly neither set consists of a single model, but our manual LPMs do better on this requirement than the automatically discovered LPMs.

To determine how well a set of LPMs satisfies R3 and R4, we project the coverage of the individual LPMs back on the original log. To do so, we check for each event by which models it is covered, and we record if it is covered at least once, and if it is covered more than once. Satisfaction of R3 is measured by the fraction of events in the log that is covered at least once. Satisfaction of R4 is measured by the fraction of events in the log that is covered more than once.

Calculating these numbers for the LPMs discovered in Section 2.4 yields a total coverage of 0.633 and a duplicate coverage of 0.313. This means that these 13 models explain less than two-thirds of the log, and they explain nearly half of the events that they do explain more than once. In contrast, the LPMs constructed in Section 3.1 have a total coverage of 0.994 with a duplicate coverage of precisely 0. Our manual LPMs explain nearly the entire log, without explaining any event more than once.

4.4 Measures versus requirements

Though we have seen unfit LPMs, they have all been fairly precise. Their simplicity prevents unexpected traces to be considered fitting, ensuring good precision scores. Should an LPM allow unobserved behavior, the precision score will still warn us of this inaccuracy, just as it does with traditional start-to-end models. Both our fitness and precision measures help in finding accurate models, which means they help us satisfy R1.

We don't have a measure for R2 other than counting the number of LPMs, and deciding on a per-case basis if that is too many. We should, however, minimize the number of LPMs without sacrificing any of the other requirements.

For R3 and R4 we have shown that measuring coverage and duplicate coverage can easily tell us how well a set of LPMs meets these requirements.

5 Conclusion

On semi-structured behavior, state-of-the-art start-to-end process model discovery techniques yield models that are too complex, not fitting and/or imprecise. New LPM discovery techniques yield too many models that repeatedly describe the same small fractions of behavior, and are not able to explain all observed behavior. However, it is possible to use sets of larger LPMs that do not have these problems. Therefore, the following requirements need to be met: (1) the individual LPMs should be accurate, (2) there should not be too many LPMs in the set, (3) the set of LPMs together should maximize coverage, and (4) the set of LPMs should minimize duplicate coverage.

It is possible to create a set of LPMs for $L_{\text{BPIC12}}^{10939}$ that satisfies these requirements, but they do not score well on the existing quality measures for LPMs. This indicates that these quality measures do not measure the qualities that our requirements demand. Adapting existing fitness and precision measurement techniques of start-to-end models for use on LPMs yields results that better describe the accuracy of individual LPMs. Accurately determining which events are described by which LPM shows both how well a set of models covers a log, and how much of that log is covered multiple times. These new techniques help distinguish good sets of LPMs from bad.

A major limitation of this paper is that it has only been shown to work on $L_{\text{BPIC12}}^{10939}$, and has not been tested on other datasets. However, the goal of this paper was to show a new way of thinking about LPMs: not as models of small pieces of an unstructured process, but rather as models of larger chunks of behavior in a semi-structured process.

Future work in this area should focus mainly on automatic discovery of sets of LPMs that meet the requirements mentioned above, measuring the simplicity of LPMs, and further improving quality measures for individual and sets of LPMs.

References

1. Adriano Augusto, Raffaele Conforti, Marlon Dumas, and Marcello La Rosa. Split miner: Discovering accurate and simple business process models from event logs. 2017.
2. Søren Debois, Thomas T. Hildebrandt, Paw Høvsgaard Laursen, and Kenneth Ry Ulrik. Declarative process mining for DCR graphs. In *Proceedings of the SAC 2017*, 2017.
3. Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo A. Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative versus imperative process modeling languages: The issue of understandability. volume 29 of *Lecture Notes in Business Information Processing*, 2009.
4. Sander J.J. Leemans, Dirk Fahland, and Wil M.P. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *International conference on business process management*, 2013.
5. Volodymyr Leno, Marlon Dumas, Fabrizio Maria Maggi, Marcello La Rosa, and Artem Polyvyanyy. Automated discovery of declarative process models with correlated data conditions. *Inf. Syst.*, 89, 2020.
6. Xixi Lu, Dirk Fahland, Robert Andrews, Suriadi Suriadi, Moe T. Wynn, Arthur H.M. ter Hofstede, and Wil M.P. van der Aalst. Semi-supervised log pattern detection and exploration using event concurrence and contextual information. In *OTM 2017 Conferences*, 2017.
7. Xixi Lu, Seyed Amin Tabatabaei, Mark Hoogendoorn, and Hajo A. Reijers. Trace clustering on very large event data in healthcare using frequent sequence patterns. *ArXiv*, abs/2001.03411.
8. Jorge Munoz-Gama et al. *Conformance checking and diagnosis in process mining*. 2016.
9. Niek Tax, Natalia Sidorova, Reinder Haakma, and Wil M.P. van der Aalst. Mining local process models. *Journal of Innovation in Digital Ecosystems*, 3(2), 2016.
10. Tom Thaler, Simon Felix Ternis, Peter Fettke, and Peter Loos. A comparative analysis of process instance cluster techniques. In *Wirtschaftsinformatik*, 2015.
11. Wil M.P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2), 2012.
12. Boudewijn F. van Dongen. BPI Challenge 2012, 2012.