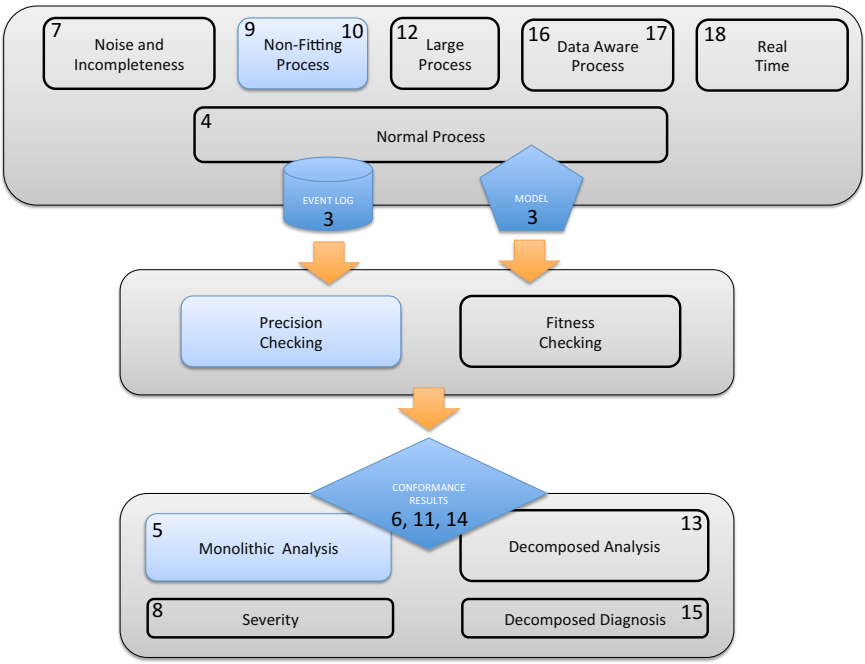
Chapter 9
Handling non-Fitness



In the first part of the book, we presented a precision checking technique based on escaping arcs. However, it is not uncommon to have a certain degree of unfitness when we measure precision, e.g., invisible or duplicate activities, or simply produced by small known mismatches between model and process. This chapter provides an overview on the use of alignment techniques as a pre-processing step to

J. Munoz-Gama: Conf. Check. ... in Process Mining, LNBIP 270, pp. 85–96, 2016.
DOI: 10.1007/978-3-319-49451-7_9

provide meaningful precision results in unfitting scenarios. In the next chapter, we will present some variants of this technique more suitable for some cases.

9.1 Introduction

As it has been illustrated in previous chapters, replaying observed behavior over the modeled behavior is an effective and efficient way to detect escaping arcs, and with them, detect potential imprecise points to be fixed. However, there are situations where the observed behavior cannot be replayed on the modeled behavior, i.e., the model cannot mimic the ”moves” observed in the log. These problems are produced by the presence of *unfitting* traces, or *undeterministic* situations on the model. For example, let us consider the model M_1 at the top of Figure 9.1, and the state $\sigma_1 = \langle a, b, c \rangle$ observed in the log. Given the choice between b and c in M_1, the state σ_1 cannot be reached by the model. A similar situation occurs for the models M_2 and M_3, in the middle and bottom of Figure 9.1, respectively. However, in this case the problems are not produced due to the unreachability of the observed state, but because there is a non-bijective relation between activity sequences and tasks sequences on the model – due to invisible and duplicate activities. Let us consider the model M_2 and the observed state $\sigma_2 = \langle a, b \rangle$. What are the set of modeled activities for the state σ_2? c, if we consider the upper b? Or d, if we consider the lower b?. The use of duplicate activities may introduce – not always – undeterminitic situations such as the one illustrated. A similar situation happens with the presence of invisible activities. For example, let us consider the model M_3 and the observed state $\sigma_3 = \langle a \rangle$. Because invisible activities are not reflected in the log, there is no way to determine if the set of modeled activities in σ_3 are b, c, or the empty set.

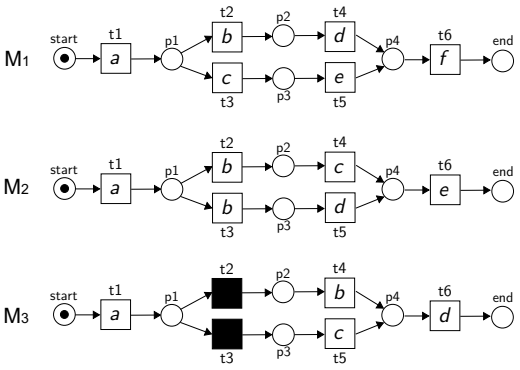


Fig. 9.1: Models to illustrate the problems with the replay produced by unfitting traces (top), the indeterminism of duplicate activities (middle), and the indeterminism of invisible activities (bottom).

The effect of problems derived from unfitting logs can be alleviated by making some assumptions over the observed behavior. For example, in Section 5.6.1 the unfitting part of the trace is considered noisy or infrequent and it is discarded for the precision metric. This drastic approach is useful when the percentage of log discarded is small, and we are not interested on providing all possible points of imprecision. However, the effect in a worst case scenario is not negligible. For example, in a log where only the first event in all traces is wrongly recorded, the whole log is discarded. Similar, heuristic assumptions could be considered to solve indeterministic situations with no formal guarantees, but still have practical applicability.

Undeterminism is produced because the escaping arcs are detected at a log level. At a model level, each task is unique, e.g., although t_2 and t_3 have the same label b, they are distinguishable in the model M_2. Therefore, a precision checking at a task level will solve indeterministic situations, transferring the responsibility to mapping correctly log traces and tasks sequences of the model. Thus, each trace in the log is translated to a complete tasks sequence in the model. For example, log traces $\langle a,b,c,e\rangle$ and $\langle a,b,d,e\rangle$ may be associated with $\langle t_1,t_2,t_4,t_6\rangle$ and $\langle t_1,t_3,t_5,t_6\rangle$ of model M_2, respectively.

The mapping between log traces and model traces is far from trivial. An approach may consider the use of heuristics, such as *look-ahead* [90, 28]. Informally, when the approach reaches a decision point, the algorithm looks ahead to choose the most suitable option. For example, in the trace $\langle a,b,c,e\rangle$ for the M_2, t_2 is selected because there is a c next. Similar happens when the indeterminism is introduced by invisibles, e.g., trace $\langle a,b,d\rangle$ is associated with complete tasks sequence $\langle t_1,t_2,t_4,t_6\rangle$ in M_3. Look-ahead heuristics can be also used to associate unfitting observed states to reachable states of the model. For example, given the observed trace $\langle a,b,c,d,f\rangle$ and the model M_1, the algorithm could consider c an event wrongly recorded in the log.

Look-ahead and other heuristics are heuristics after all, and therefore, they lack formal guarantees. The decision taken heuristically may not be the optimal. Even, when the number of events considered in the look-ahead is increased, the decision may still not be optimal, e.g., the optimal path may require the reconsideration of previous decisions [17].

In this chapter we introduce a precision checking approach based on aligning observed and modeled behavior. The alignment is done at a model level, for each one of the traces in the log. The alignment techniques provide global optimal results and therefore, there are guarantees on the escaping arcs detected. Notice that, the computation cost of aligning observed and modeled behavior in a global optimal may be considerable in some cases. Therefore, there are situations where other alternatives need to be considered, for example, decomposed aligning for conformance diagnosis (cf. Chapter 12), or a heuristic replay-based approach in real-time scenarios where the time is a crucial element (cf. Chapter 18).

9.2 Cost-Optimal Alignment

The use of alignment techniques in conformance checking was first proposed by Adriansyah, van Dongen, and van der Aalst [17]. An alignment between an event log and a process model relates occurrences of activities in the log to tasks of the model. As the execution of a case is often independent from the execution of another case, the alignment is performed per traces. This is a common assumption taken in process mining techniques, and reduces the complexity of the analysis.

For each trace in an event log that fits a process model, each *move* in the trace (i.e., an activity observed in the log) can be mimicked by a *move* in the model (i.e., a task executed in the model). However, this is not the case if the trace does not fit the model perfectly. We use the symbol $\gg$ to denote "no move" in either log or model.

Definition 9.1 (Moves [17]) *Let $L \in \mathscr{B}(A^*)$ be an event log over the activities A, and let M be a model where T is the set of tasks of the model, $A_v(M)$ is the set of observable activities of M, and l is the labeling function between tasks and observable activities in M. For the sake of clarity, we abuse the notation writing $l(t) = \tau$ if $t \notin dom(l)$, i.e., if t is an invisible task.*

- $(a_L, (a_M, t))$ *is a* move, *where* $a_L \in A \cup \gg$, $(a_M, t) \in (A_v(M) \cup \tau \times T) \cup \gg$, *and* $l(t) = a_M$.
- $(a, (a, t))$ *is a* synchronous move *(also called* move in both*), where* $a \in A$, $t \in T$, *and* $l(t) = a$.
- $(a, \gg)$ *is a* move on log, *where* $a \in A$.
- $(\gg, (a, t))$ *is a* move on model, *where* $(a, t) \in (A_v(M) \cup \tau \times T)$, *and* $l(t) = a$.
- *A* legal move *is a move on log, a move on model, or a synchronous move. Otherwise, it is an* illegal move. *A_{LM} denotes the set of possible legal moves between the model M and log L.*

Given a sequence of moves γ, $row_L^{\gg}(\gamma)$ denotes the sequence of log activities in γ, i.e., the first element. Similar, $row_M^{\gg}(\gamma)$ and $row_T^{\gg}(\gamma)$ denote the sequence of model activities and tasks, respectively. row_L, row_M and row_T denote the projection of sequences of activities in the log, model and tasks, filtering $\gg$.

Definition 9.2 (Alignment [17]) *Let $\sigma_L \in L$ be a log trace and $\sigma_M \in \phi_t(M)$ a complete task sequence of model M. An* alignment *of σ_L and σ_M is a sequence of moves $\gamma \in A_{LM}{}^*$ such that the sequence of log activities (ignoring $\gg$) yields σ_L, and the sequence of model tasks (ignoring $\gg$) yields σ_M, i.e., $row_L(\gamma) = \sigma_L$ and $row_T(\gamma) = \sigma_M$.*

Let us consider a medical process for an oncological treatment in a hospital – this process will be used as running example during this chapter. Model M in Figure 9.2 represents a possible model for this medical process. Assuming that the end state of the model is reached when place *end* in the model contains exactly one token, the model represents an infinite set of complete activity sequences, e.g., $\langle a,b,c,d\rangle$, $\langle a,c,b,d\rangle$, $\langle a,b,c,e\rangle$, $\langle a,c,b,e\rangle$, $\langle a,f,g,h\rangle$, $\langle a,b,i,c,b,e\rangle$. Given an unfitting trace $\sigma_L = \langle a,b,d,e\rangle$, Figure 9.3 shows some possible alignments between σ_L and M.

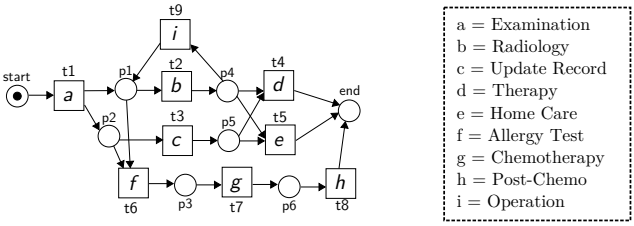


Fig. 9.2: Model for a medical process, used as running example on this chapter.

$$\gamma_1 = \begin{vmatrix} a & \gg & b & d & e \\ a & c & b & \gg & e \\ t_1 & t_3 & t_2 & & t_5 \end{vmatrix} \quad \gamma_2 = \begin{vmatrix} a & b & \gg & d & e \\ a & b & c & \gg & e \\ t_1 & t_2 & t_3 & & t_5 \end{vmatrix} \quad \gamma_3 = \begin{vmatrix} a & \gg & b & d & e \\ a & c & b & d & \gg \\ t_1 & t_3 & t_2 & t_4 & \end{vmatrix}$$

$$\gamma_4 = \begin{vmatrix} a & b & \gg & d & e \\ a & b & c & d & \gg \\ t_1 & t_2 & t_3 & t_4 & \end{vmatrix} \quad \gamma_5 = \begin{vmatrix} a & b & d & \gg & e \\ a & b & \gg & c & e \\ t_1 & t_2 & & t_3 & t_5 \end{vmatrix} \quad \gamma_6 = \begin{vmatrix} a & \gg & \gg & \gg & b & d & e \\ a & f & g & h & \gg & \gg & \gg \\ t_1 & t_6 & t_7 & t_8 & & & \end{vmatrix}$$

Fig. 9.3: Some alignments between trace $\sigma_L = \langle a, b, d, e \rangle$ and the model M in Figure 9.2.

The moves are represented vertically, e.g., the second move of γ_1 is $(\gg, (c, t_3))$, indicating that the model performs t_3 while the log does not make any move. Note that after removing $\gg$, the projections of all moves in the model are by definition complete task sequences allowed by the model. This property is not always ensured by other conformance checking approaches. For example, given a trace and a process model, when using the approach in [77], the so-called *missing tokens* are added to allow activities that occur in the trace but are not supposed to occur according to the model. The addition of such missing tokens introduces extra behavior that is not allowed in the original process model, thus overestimating its behavior.

In order to compare alignments and select the most appropriate one, *costs* are associated to undesirable moves and the alignment with the lowest total costs is selected. To quantify the costs of an alignment, a *cost function* δ is defined.

Definition 9.3 (Cost of alignment [17]) *The* cost function $\delta : A_{LM} \to \mathbb{N}$ *assigns costs to legal moves. The* cost *of an alignment* $\gamma \in {A_{LM}}^*$ *is the sum of all costs, i.e.,* $\delta(\gamma) = \sum_{(x,y)\in\gamma} \delta(x,y)$.

Different scenarios may require different cost functions. The costs may depend on the nature of the activity, e.g., skipping a payment may be more severe than sending an email. Moreover, the severity assumed for a move on log and a move on model may be different, e.g., a system with constant recording problems should be more tolerant with activities skipped on the log. Abstracting from particular cases, we can define a *standard cost function* that assigns unit costs to moves in log or moves on model only.

Definition 9.4 (Standard Cost Function [17]) *A* standard cost function δ_S *is defined such that:*

- *Synchronous move has cost* 0*, i.e.,* $\delta_S(x,(x,t)) = 0$ *for all* $x \in A$.
- *Move on log has cost* 1*, i.e.,* $\delta_S(x,\gg) = 1$.
- *Move on model from a visible task has cost* 1*, i.e.,* $\delta_S(\gg,(x,t)) = 1$.
- *Move on model from an invisible task has cost* 0*, i.e.,* $\delta_S(\gg,(\tau,t)) = 0$.

Using the standard cost function, the cost of alignment γ_1 is $\delta_S(\gamma_1) = \delta_S(a,(a,t_1)) + \delta_S(\gg,(c,t_3)) + \delta_S(b,(b,t_2)) + \delta_S(d,\gg) + \delta_S(e,(e,t_5)) = 0+1+0+1+0 = 2$. Note that the function returns the number of mismatches in the alignment. On the other hand, $\delta_S(\gamma_6) = 6$. Hence, we conclude that γ_1 is close to the log trace $\sigma_L = \langle a,b,d,e\rangle$ than γ_6.

Given a trace from an event log and a process model, we are interested in an activity sequence from the model that is most similar to the trace, i.e., the *optimal* alignment.

Definition 9.5 (Optimal Alignments [17]) *We define the set of* alignments $\Gamma_{\sigma_L,M} = \{\gamma \in A_{LM}{}^* \mid \gamma$ *is an alignment between* σ_L *and* $M\}$ *to be all possible alignments between* σ_L *and* M*. Accordingly, we define the set of* optimal alignments *as the set of all alignments with minimum cost, i.e.,* $\Gamma^o_{\sigma_L,M} = \{\gamma \in \Gamma_{\sigma_L,M} \mid \forall_{\gamma' \in \Gamma_{\sigma_L,M}}\ \delta(\gamma) \leq \delta(\gamma')\}$.

It is easy to see that there can be more than one optimal alignment between a trace and a model. For example, $\{\gamma_1,\gamma_2,\gamma_3,\gamma_4,\gamma_5\}$ is the set of optimal alignments between the trace $\sigma_L = \langle a,b,d,e\rangle$ and the model M.

By definition, the task component of all alignments yields a complete task sequence of the model. Thus, given an optimal alignment γ between σ_L and M, $row_T(\gamma)$ provides a complete tasks sequence that both perfectly fits M and is closest to σ_L. In the running example, $row_T(\gamma_1) = \langle t_1,t_3,t_2,t_5\rangle$ is one of the complete task sequences of M that is most similar to trace $\langle a,b,d,e\rangle$.

Given a log and a model, constructing all optimal alignments between all traces in the log and the model is computationally expensive [18, 19]. Thus, there are cases where computing all optimal alignments between traces and process models may not always be feasible in practice. Hence, instead of computing all optimal alignments between traces in the log and the model to obtain insights into deviations, one may also compute just some representative optimal alignments for each trace. In this chapter, we consider three approaches: *one* optimal alignment per trace, *all* optimal approaches, and a set of *representative* optimal alignments. We define three functions that provide optimal alignments between traces in the log and the model:

- $\Lambda^*_M : A^*_L \to \mathscr{P}(A_{LM}{}^*)$ returns *all optimal alignments* between traces of L and M, such that for all $\sigma_L \in L, \Lambda^*_M(\sigma_L) = \Gamma^o_{\sigma_L,M}$,
- $\Lambda^1_M : A^*_L \to A_{LM}{}^*$ returns *one optimal alignment* between traces of L and M, such that for all $\sigma_L \in L, \Lambda^1_M(\sigma_L) \in \Gamma^o_{\sigma_L,M}$, and
- $\Lambda^R_M : A^*_L \to \mathscr{P}(A_{LM}{}^*)$ returns *representatives of optimal alignments* between traces of L and M, such that for all $\sigma_L \in L, \Lambda^R_M(\sigma_L) \subseteq \Gamma^o_{\sigma_L,M}$.

$$\gamma_7 = \begin{array}{|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & f & g & h \\ \hline t_1 & t_6 & t_7 & t_8 \\ \hline \end{array} \quad \gamma_8 = \begin{array}{|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & b & c & d \\ \hline t_1 & t_2 & t_3 & t_4 \\ \hline \end{array} \quad \gamma_9 = \begin{array}{|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & c & b & d \\ \hline t_1 & t_3 & t_2 & t_4 \\ \hline \end{array}$$

$$\gamma_{10} = \begin{array}{|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & c & b & e \\ \hline t_1 & t_3 & t_2 & t_5 \\ \hline \end{array} \quad \gamma_{11} = \begin{array}{|c|c|c|c|}\hline a & \gg & \gg & \gg \\ \hline a & b & c & e \\ \hline t_1 & t_2 & t_3 & t_5 \\ \hline \end{array}$$

Fig. 9.4: All optimal alignments between trace $\sigma_L = \langle a \rangle$ and the model M in Figure 9.2.

In [22, 18, 19] various approaches to obtain an optimal alignment between a trace and a model with respect to different cost functions are investigated. Given a trace σ_L of L and a model M, if there are multiple optimal alignments, Λ_M^1 chooses one of them according to other external criteria. With our previous example, suppose that Λ_M^1 selects an alignment that has the longest consecutive occurrence of synchronous moves in the beginning, $\Lambda_M^1(\sigma_L) = \gamma_4$.

In [18, 19], an $A^\star$-based algorithm is proposed to compute one optimal alignment between a trace and a model. The same algorithm can be extended to provide more than one optimal alignment between them. Given a trace σ_L of L and a model M, the algorithm constructs one optimal alignment by computing a shortest path from the initial to the final state of the state space of the synchronous product between σ_L and M. It is shown in [19] that all shortest paths from the initial to the final state of the state space yield an optimal alignment. For each state in the state space, the algorithm records a shortest path from the initial state to reach this state and thus, becomes the *representative* of all other shortest paths from the initial state to the state. An optimal alignment is constructed from a shortest path from the initial state to the final state that is also representing all other shortest paths that connect the same pair of states. By recording all represented shortest paths during state space exploration for each state, we can obtain all shortest paths from the initial to the final state of the state space (i.e., obtain *all optimal alignments*). Different representatives may represent different number of optimal alignments. Given a representative $\gamma \in \Lambda_M^R(\sigma_L), rep_M(\gamma)$ denotes the number of optimal alignments represented by γ. Furthermore, due to possible pruning of state space, the total number of represented optimal alignments by the representatives is a lower bound of the total number of all optimal alignments, i.e., $\sum_{\gamma \in \Lambda_M^R(\sigma_L)} rep_M(\gamma) \leq |\Gamma^o_{\sigma_L,M}|$. The interested reader is referred to [18, 19, 17] for details on the constructed state space with the $A^\star$-based algorithm approach.

Take for example a trace $\sigma_L = \langle a \rangle$. All optimal alignments between the trace and the medical model M are shown in Figure 9.4. Given a possible function Λ^R, $\Lambda^R(\sigma_L) = \{\gamma_7, \gamma_9, \gamma_{10}\}$ where $rep_M(\gamma_7) = 1$ (γ_7 represents $\{\gamma_7\}$), $rep(\gamma_9) = 2$ (γ_9 represents $\{\gamma_8, \gamma_9\}$), and $rep(\gamma_{10}) = 2$ (γ_{10} represents $\{\gamma_{10}, \gamma_{11}\}$).

For simplicity, in the remainder we omit the model notation M in functions Λ_M^*, Λ_M^1, Λ_M^R, and rep_M whenever the context is clear. Note that in cases where a process

model has duplicate tasks (more than one task to represent an activity) or invisible tasks (tasks whose execution are not logged), approaches to construct alignments (e.g., [22, 18]) keep the mapping from all model moves to the tasks they correspond to. Hence, given an alignment of a trace and such models, we know exactly which task is executed for each model move. We refer to [22, 18] for further details on how such mapping is constructed.

9.3 Precision based on Alignments

The technique described in the previous section provides optimal alignments for each trace in the log. This section presents a technique to compute *precision* based on the use of these optimal alignments. Like the approach on Chapter 5, the behavior observed in the log is used to traverse the modeled behavior, detecting escaping arcs for possible points of imprecision. However, whereas in Chapter 5 is based on model replay directly from the log, the approach presented here uses the alignments as a more faithful representation of the observed behavior. The advantages are manifold. First of all, traces in the log do not need to be completely fitting. In Chapter 5, the non-fitting parts are simply ignored. For most real-life situations, this implies that only a fraction of the event log can be used for computing precision. Second, the existence of indeterminism in the model poses no problems when using the alignments. Finally, the use of alignments instead of log-based model replay improves the robustness of conformance checking. The remainder of this section is devoted to explain how precision can be calculated from the alignments. In particular, we consider the precision checked from *one* alignment, *all* alignments, and *representative* alignments. To illustrate the three approaches, in the remainder of the section we use the following running example: the model M shown in Figure 9.2 and the log $L = [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5]$, containing the 5 traces that appear in in Table 9.1. The table also provides the optimal alignments for the traces in L.

Freq	Trace	Optimal Alignment
1	$\sigma_1 = \langle a \rangle$	$\gamma_{1a} = \begin{vmatrix} a & \gg & \gg & \gg \\ a & f & g & h \\ t_1 & t_6 & t_7 & t_8 \end{vmatrix}$ $\gamma_{1b} = \begin{vmatrix} a & \gg & \gg & \gg \\ a & b & c & d \\ t_1 & t_2 & t_3 & t_4 \end{vmatrix}$ $\gamma_{1c} = \begin{vmatrix} a & \gg & \gg & \gg \\ a & c & b & d \\ t_1 & t_3 & t_2 & t_4 \end{vmatrix}$ $\gamma_{1d} = \begin{vmatrix} a & \gg & \gg & \gg \\ a & c & b & e \\ t_1 & t_3 & t_2 & t_5 \end{vmatrix}$ $\gamma_{1e} = \begin{vmatrix} a & \gg & \gg & \gg \\ a & b & c & e \\ t_1 & t_2 & t_3 & t_5 \end{vmatrix}$
1	$\sigma_2 = \langle a,b,c,d \rangle$	$\gamma_2 = \begin{vmatrix} a & b & c & d \\ a & b & c & d \\ t_1 & t_2 & t_3 & t_4 \end{vmatrix}$
1	$\sigma_3 = \langle a,c,b,e \rangle$	$\gamma_3 = \begin{vmatrix} a & c & b & e \\ a & c & b & e \\ t_1 & t_3 & t_2 & t_5 \end{vmatrix}$
1	$\sigma_4 = \langle a,f,g,h \rangle$	$\gamma_4 = \begin{vmatrix} a & f & g & h \\ a & f & g & h \\ t_1 & t_6 & t_7 & t_8 \end{vmatrix}$
1	$\sigma_5 = \langle a,b,i,b,c,d \rangle$	$\gamma_5 = \begin{vmatrix} a & b & i & b & c & d \\ a & b & i & b & c & d \\ t_1 & t_2 & t_9 & t_2 & t_3 & t_4 \end{vmatrix}$

Table 9.1: Optimal alignments of log $[\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5]$ and the medical model M of Figure 9.2

9.4 Precision from 1-Alignment

Like Chapter 5, precision is estimated by confronting model and log behavior: escaping arcs between the model and the log (i.e., situations where the model allows more behavior than the one reflected in the log) are detected by juxtaposing behavior observed in the log and the one allowed by the model. This juxtaposition is done in terms of an automaton: first, an automaton is built from the alignments. Then, the automaton is enhanced with behavioral information of the model. Finally, the enhanced automaton is used to compute the precision. In order to build the automaton, observed behavior must be determined in terms of model perspective, i.e., we consider the optimal alignments of each trace in the log for this purpose. For ex-

ample, given the running example L and M, the trace σ_1 has 5 optimal alignments, $\{\gamma_{1a}, \gamma_{1b}, \gamma_{1c}, \gamma_{1d}, \gamma_{1e}\}$, shown in Table 9.1. However, in 1-alignment only one alignment is considered. For this example, we assume that the alignment assigned to σ_1 by Λ^1 based on an external criterion corresponds to γ_{1a}, i.e., $\Lambda^1(\sigma_1) = \gamma_{1a}$. On the other hand, traces $\sigma_2 \dots \sigma_5$ are perfectly fitting and have only one optimal alignment containing only synchronous moves. Given an alignment γ, in order to build the automaton, we only consider the projection of tasks moves, i.e., $row_T(\gamma)$. In this example, the sequences used as observed behavior are $\langle t_1, t_6, t_7, t_8\rangle$, $\langle t_1, t_2, t_3, t_4\rangle$, $\langle t_1, t_3, t_2, t_5\rangle$, $\langle t_1, t_6, t_7, t_8\rangle$ and $\langle t_1, t_2, t_9, t_2, t_3, t_4\rangle$. We use $row_T(\Lambda^1)_L$ to denote the application of function row_T on all the alignments provided by the functions Λ^1 for the traces in log L. We can omit the subindex L whenever the context is clear. Note that, by definition, any alignment projection $row_T(\gamma)$ is a valid complete firing sequence of the model.

Similar to Chapter 5, the automaton is built considering all the prefixes for the sequences in $row_T(\Lambda^1)$ as the states. For instance, given a sequence $\langle t_1, t_2, t_3, t_4\rangle$ resulting of $row_T(\Lambda^1)(\sigma_2)$, the states considered are $\langle\rangle$, $\langle t_1\rangle$, $\langle t_1, t_2\rangle$, $\langle t_1, t_2, t_3\rangle$ and $\langle t_1, t_2, t_3, t_4\rangle$. We denote as $\bullet(row_T(\gamma))$ the set of *prefixes of the tasks sequence* of the alignment γ and as $\bullet(row_T(\Lambda^1))$ the multiset of *prefixes of the the tasks sequences* of all alignments in Λ^1.

Definition 9.6 (Prefix Automaton of the 1-Alignment) *Let $L \in \mathscr{B}(A^*)$ be an event log, let M be a model with tasks T , and let $row_T(\Lambda^1)$ be the alignments between them projected on the model tasks. We define the* prefix automaton of the 1-Alignment $\mathscr{A}_{\Lambda^1 M} = (S, T, \nearrow, \omega, s_0)$ *such that:*

- *the set of* states *corresponds to the set of prefixes of the alignments projected on the model tasks, i.e., $S = \{\sigma | \sigma \in \bullet(row_T(\Lambda^1))\}$.*
- *the set of* labels *corresponds to the set of tasks T.*
- *the* arcs *$\nearrow \subseteq (S \times T \times S)$ define the concatenation between states and tasks, i.e., $\nearrow = \{(\sigma, t, \sigma \cdot \langle t\rangle) | \sigma \in S \wedge \sigma \cdot \langle t\rangle \in S\}$.*
- *the function that determines the* weight *of a state is determined by the number of occurrences of the state in the multiset of prefixes of the tasks sequences, i.e., $\omega(\sigma) = \bullet(row_T(\Lambda^1))(\sigma)$ for all $\sigma \in S$.*
- *the initial state s_0 corresponds with the empty prefix $\langle\rangle$.*

Figure 9.5 shows the resulting automata for the running example L using the function Λ^1 (only the white states). For example, the weight of the state $\langle t_1\rangle$ is greater than the weight of $\langle t_1, t_3\rangle$ because there are more tasks sequences with the prefix $\langle t_1\rangle$ (all 5 sequences), than the ones with prefix $\langle t_1, t_3\rangle$ (only the sequence $\langle t_1, t_3, t_2, t_5\rangle$ contains that prefix).

Once the observed behavior has been determined in terms of an automaton, the confrontation with the actual modeled behavior is required in order to determine the precision. For each state of the automaton, we compute its set of *modeled tasks*, i.e., possible direct successor tasks according to the model (*mod*), and then compare it with the set of *observed tasks*, i.e., tasks really executed in the log (*obs*)(cf. Definition 5.4). Let us consider, for example, state $\langle t_1, t_2, t_3\rangle$ of automaton in Figure 9.5.

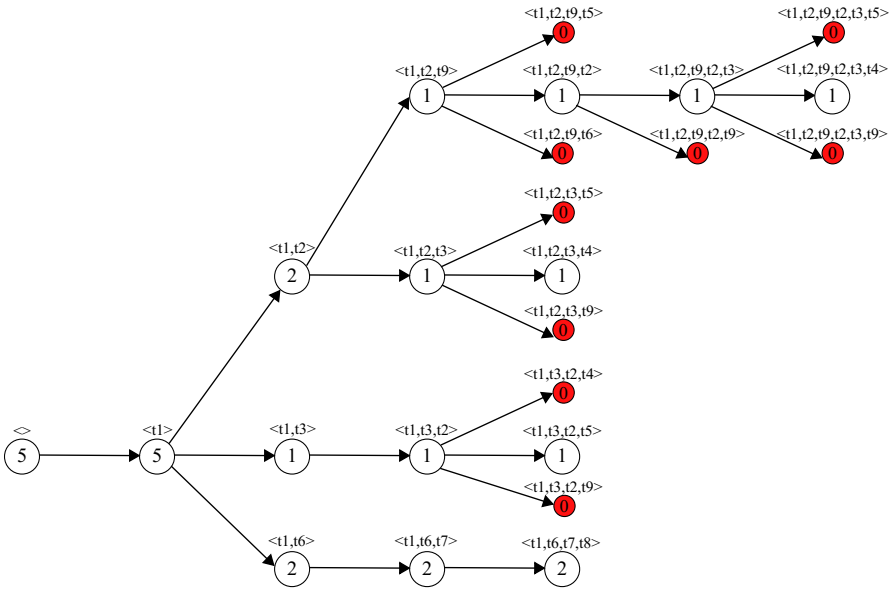


Fig. 9.5: Automaton from 1-alignments between model M and log L.

The set of observed tasks of the state is $obs(\langle t_1,t_2,t_3\rangle) = \{t_4\}$, i.e., for all traces with prefix $\langle t_1,t_2,t_3\rangle$, their direct successor is only t_4. The set of modeled tasks for the state is $mod(\langle t_1,t_2,t_3\rangle) = \{t_4,t_5,t_9\}$ because after performing the sequence of tasks $\langle t_1,t_2,t_3\rangle$, the model allows to do t_4, t_5 or t_9. Note that, by definition of alignment, $obs(s) \subseteq mod(s)$, i.e., the set of executed tasks of a given state is always a subset of all available tasks according to the model.

The arcs that are modeled according to the model, but do not occur in the event log according to the alignments, are used to collect the *escaping arcs* of the system, i.e., arcs that escapes from the observed behavior. The tasks on the escaping arcs and the states reached are called escaping tasks and escaping states respectively. In Figure 9.5 the escaping states are in color. For example, the escaping tasks of the state $\langle t_1,t_2,t_3\rangle$ are $\{t_4,t_5,t_9\} \setminus \{t_4\} = \{t_5,t_9\}$. The computation and analysis of these escaping arcs are the cornerstone of the precision checking technique presented in this book. All identified escaping arcs can be analyzed and further used to correct the model and make it more precise. Furthermore, in order to globally estimate precision, these escaping arcs in turn are weighted and normalized defining a metric to measure precision called *1-align precision* metric.

Definition 9.7 (1-Align Precision metric) *Let $\mathscr{A}_{\Lambda^1 M} = (S,T,\nearrow,\omega,s_0)$ be the prefix automaton of the alignments in Λ^1 enhanced with the behavior of the model M. The metric* 1-Align Precision *estimates the precision of the system comparing, for each state in S, the number of escaping arcs with the number of allowed arcs. The numbers are weighted according to the importance of the state. Formally:*

$$a_p^1(\mathscr{A}_{\Lambda^1 M}) = 1 - \frac{\sum_{s \in S} \omega(s) \cdot |esc(s)|}{\sum_{s \in S} \omega(s) \cdot |mod(s)|}$$

For example, the precision for the automaton derived from Λ^1 shown in Figure 9.5 is 0.79.

9.5 Summary

This chapter presented an overview on the alignment techniques. These techniques are used as a pre-processing step to align modeled and observed behavior, providing meaningful precision results in unfitting scenarios. In the next chapter several variants of the main technique are presented, to explore different scenarios.