# Conformance for DecSerFlow Constraints*

Yutian Sun and Jianwen Su

Department of Computer Science, U C Santa Barbara, USA
{sun,sun}@cs.ucsb.edu

**Abstract.** DecSerFlow is a declarative language to specify business processes. It consists of a set of temporal predicates that can be translated into LTL but limited to finite sequences. This paper focuses on the "conformance problem": Given a set of DecSerFlow constraints, is there an execution sequence that satisfies all given constraints? This paper provides syntactic characterizations of conformance for several subclasses of DecSerFlow constraints. These characterizations directly lead to efficient (polynomial time) conformance testing. Furthermore, algorithms are developed to generate conforming strings if the set of constraints is conformable. A conformance analyzer is developed based on the syntactic characterizations and the string generating algorithms. Experiments reveal several interesting factors concerning performance and scalability.

## 1 Introduction

Enterprises rely on business processes to accomplish business goals (handling a loan application, etc.) Business process models are either imperative or declarative [12]. Imperative models typically employ graphs (e.g., automata, Petri Nets) to depict how a process should progress. Declarative models are usually based on constraints [2], they are flexible and easy to change during design time or runtime [18]. A practical problem is whether a given set of constraints allows at least one execution. It is fundamental in business process modeling to test satisfiability of a given set of constraints.

A process execution is a (finite) sequence of activities through time. The declarative language DecSerFlow [2] uses a set of temporal predicates as a process specification, The DECLARE system [11] supports design and execution of DecSerFlow processes. In [14] an orchestrator for a declarative business process called REFlex was developed, where a subset of DecSerFlow can be expressed by REFlex. In this paper, we study the following conformance problem: does there exist an execution that satisfies a given DecSerFlow specification? Clearly efficient conformance testing provides an effective and efficient help to the user of DECLARE and the scheduler of [14]. Temporal predicates in DecSerFlow can be translated into linear temporal logic (LTL) [13] but limited to *finite sequences*. A naive approach to conformance checking is to construct automata representing individual constraints and determine if their cross product accepts a string. Complexity of this approach is exponential in the number of given constraints. This paper aims at efficient comformance checking.

Most DecSerFlow constraints can be categorized into two directions: "response" (*Res*), which specifies that an activity should happen in the future, and "precedence"

* Supported in part by a grant from Bosch.

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 139–153, 2014.

(*Pre*), which specifies that an activity should happen in the past. For each direction, there are three types of constraints: (1) An ordering constraint $Res(a, b)$ (or $Pre(a, b)$) for activities $a$ and $b$ specifies that if $a$ occurs, then $b$ should occur in the future (resp. past). As a practical example of a loan application, if activity "loan approval" happens, then in the past a "credit check" activity should have happened. (2) An alternating constraint $aRes(a, b)$ (or $aPre(a, b)$) specifies that each occurrence of $a$ implies a future (resp. past) occurrence of $b$ but before (resp. after) the occurrence of $b$, $a$ cannot occur again (i.e., between two occurrences of $a$, there should exist an occurrence of $b$). As an example, if a "house evaluation request" activity happens, a "house evaluation feedback" activity should happen in the future and before receiving the feedback, the applicant cannot submit another evaluation request, i.e., "request" and "feedback" should alternate. (3) An immediate constraint $iRes(a, b)$ (or $iPre(a, b)$) restricts that if $a$ occurs, then $b$ should immediately follow (resp. occur before). In addition to "response" and "precedence" constraints, there is a special type of "existence" constraints that only require occurrences in any order. An existence constraint $Exi(a, b)$ restricts that if $a$ occurs, then $b$ should occur either earlier or later. In practice, a common existence constraint can be that a guest can either choose to pay the hotel expense online then check in, or check in first and pay the expense later, i.e., *Exi*("check in", "payment").

In addition to temporal constraints, there may be cardinality requirements on each activity, i.e., an activity should occur at least once. For example, in an online order process, "payment" activity is always required to occur; while "shipping" is not (a customers may pick up the ordered items in a store).

The contributions of this paper are the following: We present a reduction from general DecSerFlow to DecSerFlow "Core" with no existence constraints nor cardinality requirements (Theorem 2.3). For DecSerFlow Core, we formulate syntactic characterizations (sufficient and necessary for conformance) for constraints involving (1) ordering and immediate constraints (Theorem 3.5), (2) ordering and alternating constraints (Theorem 3.9), (3) alternating and immediate constraints (Theorem 3.20), or (4) only precedence (or only response) constraints (Theorem 3.23). For the general case, it remains open whether syntactic characterizations exist. Algorithms are also developed to generate conforming strings when the schema is conformable. Finally, we designed and implemented a conformance analyzer and our experimental evaluation shows that (1) the syntactic condition approach is polynomially scalable (in time) comparing with the exponential-time naive approach using automata, (2) the time complexity of conforming string generation varies from polynomial to exponential complexity, and (3) the increasing number of constraints will increase the time needed of the automata approach exponentially more than the time needed by the syntactic condition approach.

The remainder of the paper is organized as follows. Section 2 defines DecSerFlow constraints studied in this paper. Section 3 focuses on different combinations of constraints together with their conformance checking and conforming string generation. A conformance checker is developed and evaluated in Section 4. Related work and conclusions are provided in Sections 5 and 6, resp. Detailed proofs, some examples, algorithms, and formal definitions are omitted due to space limitation.

## 2 DecSerFlow Constraints

In this section we introduce DecSerFlow constraints, define the conformance problem, state a straightforward result, and present a reduction to the case of "core" constraints.

Let $\mathcal{A}$ be an infinite set of *activities*, $\mathbb{N}$ the set of natural numbers, and $A \subseteq \mathcal{A}$ a finite subset of $\mathcal{A}$. A *string over* $A$ (or $\mathcal{A}$) is a finite sequence of 0 or more activities in $A$ (resp. $\mathcal{A}$). $A^*$ ($\mathcal{A}^*$) the set of all strings over $A$ (resp. $\mathcal{A}$).

A *subsequence* of $a_1a_2...a_n$ is a string $a_{k_1}a_{k_2}...a_{k_m}$, where (1) $m \in \mathbb{N}$ and $m \geqslant 1$, (2) $k_i \in [1..n]$ for each $i \in [1..m]$, and (3) $k_i < k_{i+1}$ for each $i \in [1..(m-1)]$; a *substring* is a subsequence $a_{k_1}a_{k_2}...a_{k_m}$ where for each $i \in [1..(m-1)]$, $k_{i+1} = k_i + 1$.

Let $A \subseteq \mathcal{A}$, $a, b \in A$. A *(sequence) constraint on* $a, b$ is a constraint shown in Fig. 1.

| | Response | Precedence |
|---|---|---|
| Ordering | *Res*$(a, b)$: each occurrence of $a$ is followed by an occurrence of $b$ | *Pre*$(a, b)$: each occurrence of $a$ is preceded by an occurrence of $b$ |
| Alternating | *aRes*$(a, b)$: in addition to *Res*$(a, b)$, $a$ and $b$ alternate | *aPre*$(a, b)$: in addition to *Pre*$(a, b)$, $a$ and $b$ alternate |
| Immediate | *iRes*$(a, b)$: each occurrence of $a$ is immediately followed by an occurrence of $b$ | *iPre*$(a, b)$: each occurrence of $a$ is immediately preceded by an occurrence of $b$ |
| Existence | *Exi*$(a, b)$: each occurrence of $a$ implies an occurrence of $b$ | |

**Fig. 1.** Summary of Constraints

For ordering precedence constraint *Pre*$(a, b)$, if "$a$" occurs in a string, then before "$a$", there must exist a "$b$", and between this "$b$" and "$a$", all activities are allowed to occur. Similarly, for alternating response constraint *aRes*$(a, b)$, after an occurrence of "$a$", no other $a$'s can occur until a "$b$" occurs. For immediate precedence constraint *iPre*$(a, b)$, a "$b$" should occur immediately before "$a$". The existence constraints have no restrictions on temporal orders. Given a constraint $c$ and a string $s$, denote $s \models c$ if $s$ *satisfies* $c$, for example, $s \models$ *Res*$(a, b)$, if $s = abcadb$.

**Definition 2.1.** A (*DecSerFlow*) *schema* is a triple $S = (A, C, \kappa)$ where $A \subseteq \mathcal{A}$ is a finite set of activities, $C$ a finite set of constraints on activities in $A$, and $\kappa$ is a total mapping from $A$ to $\{0, 1\}$, called *cardinality*, to denote that an activity $a \in A$ should occur at least once (if $\kappa(a) = 1$) or no occurrence requirement (if $\kappa(a) = 0$).

**Definition 2.2.** A finite string $s$ over $A$ *conforms to* schema $S = (A, C, \kappa)$ if $s$ satisfies every constraint in $C$ and for each activity $a \in A$, $s$ should contain $a$ for at least $\kappa(a)$ times. If a string $s$ conforms to $S$, $s$ is a *conforming string* of $S$ and $S$ is *conformable*.

***Conformance Problem***: Given a schema $S$, is $S$ conformable?

A naive approach to solve the conformance problem is to construct an automaton $A$ for each given constraint $c$ (and each cardinality requirement $r$, i.e., an actvitiy occurring at least 0 or 1 times), such that $A$ can accept all strings that satisfy $c$ (resp. accept all strings that satisfy $r$) and reject all other strings. Then the conformance problem is reduced to checking if the cross product of all constructed automata accepts a string.

However, the automata approach yields to exponential complexity in the size of the input schema. Our goal is to find syntactic conditions to determine conformity that lead to polynomial complexity.

For notation convenience, given a DecSerFlow schema $S = (A, C, \kappa)$, if for each $a \in A$, $\kappa(a) = 1$, we simply use $(A, C)$ to denote $S$.

**Theorem 2.3.** Given $S = (A, C, \kappa)$ as a schema, there exists a schema $S' = (A', C')$ such that $S$ is conformable iff $S'$ is conformable.

Theorem 2.3 shows that conformance of arbitrary schemas can be reduced to conformance of schemas where each activity occurs at least once. If each activity in a given schema occurs at least once, the existence constraints are redundant. In the remainder of this paper, we only focus on schemas with *core* constraints, i.e., from set $\{Res, Pre, aRes, aPre, iRes, iPre\}$ and that each activity occurs at least once.

## 3 Characterizations for Conformance

### 3.1 Ordering and Immediate Constraints

This subsection focuses on syntactic characterizations of conformable schemas that only contain ordering and/or immediate constraints.

For each schema $S = (A, C)$, we construct the *causality graph* $\mathcal{G}_S$ of $S$ as a labeled graph $(A, E^{\text{or}}_{\blacktriangleright}, E^{\text{or}}_{\blacktriangleleft}, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleleft})$ with the vertex set $A$ and six edge sets where $E^{\text{x}}_{\blacktriangleright}$ ($E^{\text{x}}_{\blacktriangleleft}$) corresponds to response (resp. precedence) constraints of ordering (x = 'or'), alternating (x = 'al'), or immediate (x = 'im') flavor. Specifically, for all $a, b \in A$, $(a, b) \in E^{\text{or}}_{\blacktriangleright}$ iff $Res(a, b)$ is in $C$, $(a, b) \in E^{\text{al}}_{\blacktriangleleft}$ iff $aPre(a, b) \in C$, $(a, b) \in E^{\text{im}}_{\blacktriangleright}$ iff $iRes(a, b) \in C$, and the other three cases are similar.

Given a causality graph $(A, E^{\text{or}}_{\blacktriangleright}, E^{\text{or}}_{\blacktriangleleft}, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleleft})$, if an edge set is empty, we will conveniently omit it; for example, if $E^{\text{im}}_{\blacktriangleright} = E^{\text{im}}_{\blacktriangleleft} = \varnothing$, we write the causality graph simply as $(A, E^{\text{or}}_{\blacktriangleright}, E^{\text{or}}_{\blacktriangleleft}, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft})$.

For technical development, we review some well-known graph notions. Given a (directed) graph $(V, E)$ with vertex set $V$ and edge set $E \subseteq V \times V$, a *path* is a sequence $v_1 v_2 ... v_n$ where $n > 1$, for each $i \in [1..n]$, $v_i \in V$, and for each $i \in [1..(n-1)]$, $(v_i, v_{i+1}) \in E$; $n$ is the *length* of the path $v_1 ... v_n$. A path $v_1 ... v_n$ is *simple* if $v_i$'s are pairwise distinct except that $v_1, v_n$ may be the same node. A (*simple*) *cycle* is a (resp. simple) path $v_1 ... v_n$ where $v_1 = v_n$. A graph is *cyclic* if it contains a cycle, *acyclic* otherwise. Given an acyclic graph $(V, E)$, a *topological order* of $(V, E)$ is an enumeration of $V$ such that for each $(u, v) \in E$, $u$ precedes $v$ in the enumeration. A *subgraph* $(V', E')$ of $(V, E)$ is a graph, such that $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$. A graph is *strongly connected* if there is a path from each node in the graph to each other node. Given a graph $G = (V, E)$ and a set $V' \subseteq V$, the *projection* of $G$ on $V'$, $\pi_{V'} G$, is a subgraph $(V', E')$ of $G$ where $E' = E \cap (V' \times V')$. A *strongly connected component* $(V', E')$ of a graph $G = (V, E)$ is a strongly connected subgraph $G' = (V', E')$ of $G$, such that (1) $G' = \pi_{V'} G$, and (2) for each $v \in V - V'$, $\pi_{V' \cup \{v\}} G$ is not strongly connected.

To obtain the syntactic conditions for deciding the conformance of ordering and immediate constraints, we first present a pre-processing upon a given schema, such that the given schema is conformable if and only if the pre-processed the schema is conformable, and then show the syntactic conditions upon the pre-processed schemas.

**Lemma 3.1.** Given a schema $S = (A, C)$ and its causality graph $(A, E^{\text{or}}_{\blacktriangleright}, E^{\text{or}}_{\blacktriangleleft}, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleleft})$, for each $(u, v) \in E^{\text{im}}_{\blacktriangleright} \cup E^{\text{im}}_{\blacktriangleleft}$, if there exists $w \in A - \{u\}$, such that $(v, w) \in E^{\text{or}}_{\blacktriangleleft}$ (or $E^{\text{or}}_{\blacktriangleright}$), then for each conforming string $s$ of $S$, $s$ satisfies $Pre(u, w)$ (resp. $Res(u, w)$).

Lemma 3.1 is straightforward. Based on Lemma 3.1, we define the following pre-processing given a schema.

**Definition 3.2.** Given a schema $S = (A, C)$, the *immediate-plus* (or *im*$^+$) *schema* of $S$ is a schema $(A, C')$ constructed as follows: 1. Initially $C' = C$. 2. Repeat the following steps while $C'$ is changed: for each distinct $u, v, w \in A$, if (1) $iPre(u, v)$ or $iRes(u, v)$ is in $C'$ and (2) $Pre(v, w) \in C'$ (or $Res(v, w) \in C'$), then add $Pre(u, w)$ (resp. $Res(u, w)$) to $C'$.

**Example 3.3.** A schema $S$ has 3 activities, $a, b, c$, and 4 constraints $iRes(a, c)$, $iRes(b, c)$, $Pre(c, a)$, and $Pre(c, b)$. Let $S'$ be the im$^+$schema of $S$. According to the definition of im$^+$schema, in addition to the constraints in $S$, $S'$ also contains constraints: $Pre(a, b)$ (which is obtained from $iRes(a, c)$ and $Pre(c, b)$) and $Pre(b, a)$. ∎

It is easy to see that for each given schema, its corresponding im$^+$schema is unique. The following is a consequence of Lemma 3.1.

**Corollary 3.4.** A schema is conformable iff its im$^+$schema is conformable.

For reading convenience, we introduce the following notations: let x, y, z be one of 'or', 'al', 'im'; we denote $E^{\mathrm{x}}_{\blacktriangleright} \cup E^{\mathrm{y}}_{\blacktriangleright}$ as $E^{\mathrm{x}\cup\mathrm{y}}_{\blacktriangleright}$ and use similar notations $E^{\mathrm{x}\cup\mathrm{y}}_{\blacktriangleleft}$ or $E^{\mathrm{x}\cup\mathrm{y}\cup\mathrm{z}}_{\blacktriangleleft}$.

**Theorem 3.5.** Given a schema $S = (A, C)$ where $C$ contains only ordering and immediate constraints, the im$^+$schema $S'$ of $S$, and the causality graph $(A, E^{\mathrm{or}}_{\blacktriangleright}, E^{\mathrm{or}}_{\blacktriangleleft}, E^{\mathrm{im}}_{\blacktriangleright}, E^{\mathrm{im}}_{\blacktriangleleft})$ of $S'$, $S$ is conformable iff the following conditions all hold.

(1). $(A, E^{\mathrm{or}\cup\mathrm{im}}_{\blacktriangleright})$ and $(A, E^{\mathrm{or}\cup\mathrm{im}}_{\blacktriangleleft})$ are both acyclic,

(2). for each $(u, v) \in E^{\mathrm{im}}_{\blacktriangleright}$ (or $E^{\mathrm{im}}_{\blacktriangleleft}$), there does not exist $w \in A$ such that $w \neq u$ and $(v, w) \in E^{\mathrm{im}}_{\blacktriangleleft}$ (resp. $E^{\mathrm{im}}_{\blacktriangleright}$), and

(3). for each $(u, v) \in E^{\mathrm{im}}_{\blacktriangleright}$ (or $E^{\mathrm{im}}_{\blacktriangleleft}$), there does not exist $w \in A$ such that $w \neq v$ and $(u, w) \in E^{\mathrm{im}}_{\blacktriangleright}$ (resp. $E^{\mathrm{im}}_{\blacktriangleleft}$).

In Theorem 3.5, Condition (1) restricts that the response or precedence direction does not form a loop (a loop of the same direction can lead to infinite execution). Conditions (2) and (3) similarly restrict that the immediate constraints are consistent. For example, it is impossible to satisfy constraints $iRes(a, b)$ and $iRes(a, c)$, where $a, b, c$ are activities.

Example 3.6 shows the importance of "pre-processing" to obtain im$^+$schemas.

**Example 3.6.** Let $S$ and $S'$ be as stated in Example 3.3. Note that $S$ satisfies all conditions in Theorem 3.5. However, $S'$ does not, since $Pre(a, b)$ and $Pre(b, a)$ form a cycle in $E^{\mathrm{or}\cup\mathrm{im}}_{\blacktriangleleft}$, which leads to non-conformability of $S$. Therefore, a pre-processing to obtain an im$^+$is necessary when determining conformability. ∎

Given a conformable schema that contains only ordering and immediate constraints, one question to ask is how to generate a conforming string. To solve this problem, we first introduce a (data) structure, which is also used in the later sections.

For a schema $S = (A, C)$, let $\pi_{\mathrm{im}}(S) = (A, C')$ be a schema where $C'$ is the set of all immediate constraints in $C$. The notation $\pi_{\mathrm{im}}(S)$ holds the *projection* of $S$ on immediate constraints. Similarly, let $\pi_{\mathrm{al}}(S)$ be the *projection* of $S$ on alternating constraints.

Given a schema $S = (A, C)$, if $\pi_{\mathrm{im}}(S)$ satisfies the conditions stated in Theorem 3.5, then for each activity $a \in A$, denote $\bar{\mathrm{S}}_{\mathrm{im}}(a)$ as a string constructed iteratively as follows:

(i) $\bar{s}_{im}(a) = a$ initially, (ii) for the leftmost (or rightmost) activity $u$ of $\bar{s}_{im}(a)$, if there exists $v \in A$ such that $iPre(u, v) \in C$ (resp. $iRes(u, v) \in C$), then update $\bar{s}_{im}(a)$ to be $v\bar{s}_{im}(a)$ (resp. $\bar{s}_{im}(a)v$), i.e., prepend (resp. append) $\bar{s}_{im}(a)$ with $v$, and (iii) repeat step (ii) until no more changes can be made. For each $a \in A$, it is easy to see that $\bar{s}_{im}(a)$ is unique and is finite. Let $s_{im}(a)$ be the set of activities that occur in $\bar{s}_{im}(a)$.

Alg. 1 shows the procedure to create a conforming string given a schema with only ordering and immediate constraints. The main idea of Alg. 1 relies on a topological order of both the "precedence" and "response" directions (to satisfy the ordering constraints); then replace each activity $a$ by $\bar{s}_{im}(a)$ (to satisfy the immediate constraints).

**Algorithm 1.**

Input: A causality graph $(A, E^{or}_{\blacktriangleright}, E^{or}_{\blacktriangleleft}, E^{im}_{\blacktriangleright}, E^{im}_{\blacktriangleleft})$ of an im$^+$schema of a schema $S$ that satisfies all conditions in Theorem 3.5

Output: A finite string that conforms to $S$

A. Let "$a_1a_2...a_n$" and "$b_1b_2...b_n$" be topological sequences of $(A, E^{or \cup im}_{\blacktriangleright})$ and $(A, E^{or \cup im}_{\blacktriangleleft})$, resp.

B. Return the string "$\bar{s}_{im}(b_n)...\bar{s}_{im}(b_1)\bar{s}_{im}(a_1)...\bar{s}_{im}(a_n)$".

### 3.2 Ordering and Alternating Constraints

This subsection focuses on syntactic conditions for conformance of schemas that contain ordering and alternating constraints.

We begin with defining "pre-processing" for schemas such that the original schema is conformable if and only if the schema after the pre-processing also is. The pre-processing will also be used in the next subsection.

**Definition 3.7.** Given a schema $S = (A, C)$ and its causality graph $(A, E^{or}_{\blacktriangleright}, E^{or}_{\blacktriangleleft}, E^{al}_{\blacktriangleright}, E^{al}_{\blacktriangleleft}, E^{im}_{\blacktriangleright}, E^{im}_{\blacktriangleleft})$, the *alternating-plus* (or *al*$^+$) *schema* of $S$ is a schema $(A, C')$ where

$$C' = C \cup \{aPre(v, u) \mid (u, v) \in E^{al}_{\blacktriangleright}, u \text{ and } v \text{ are on a common cycle in } (A, E^{al}_{\blacktriangleright} \cup E^{al}_{\blacktriangleleft})\} \cup \{aRes(v, u) \mid (u, v) \in E^{al}_{\blacktriangleleft}, u \text{ and } v \text{ are on a common cycle in } (A, E^{al}_{\blacktriangleright} \cup E^{al}_{\blacktriangleleft})\}$$

It is easy to see that for each given schema, its corresponding al$^+$schema is unique.

**Lemma 3.8.** A schema is conformable iff its al$^+$schema is conformable.

**Theorem 3.9.** Given a schema $S$ that only contains ordering and alternating constraints, let $S' = (A, C)$ be the al$^+$schema of $S$ and $(A, E^{or}_{\blacktriangleright}, E^{or}_{\blacktriangleleft}, E^{al}_{\blacktriangleright}, E^{al}_{\blacktriangleleft})$ the causality graph of $S'$. $S$ is conformable iff both $(A, E^{or \cup al}_{\blacktriangleright})$ and $(A, E^{or \cup al}_{\blacktriangleleft})$ are acyclic.

**Example 3.10.** Consider a schema with 5 activities, $a, b, c, d, e$, and constraints in the form of a graph $(A, E^{or \cup al}_{\blacktriangleright} \cup E^{or \cup al}_{\blacktriangleleft})$ as shown in Fig. 2, where the edge labels denote constraint types. Note that its al$^+$schema is itself. The conditions in Theorem 3.9 are satisfied, thus the schema is conformable. A conforming string is *dcebadce*. If we add the constraint $aPre(d, b)$ into the schema, it is no longer conformable since *bcd* forms a cycle in $(A, E^{or \cup al}_{\blacktriangleleft})$, forcing the subsequence *bcd* to occur infinitely many times. ∎

Alg. 2 presents the procedure to construct a conforming string given a conformable schema that contains only ordering and alternating constraints. A key step of the Alg. 2 is to first create a topological order of precedence constraints and that of response constraints, then for each violated alternating constraint, insert a string to fix the violation.

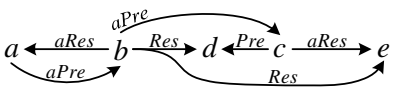


**Fig. 2.** An al$^+$schema example

**Algorithm 2.**

Input: The causality graph $(A, E^{\text{or}}_{\blacktriangleright}, E^{\text{or}}_{\blacktriangleleft}, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft})$ of an al$^+$schema $S$ satisfying conditions of Theorem 3.9

Output: A string that conforms to $S$

A. Let $s_{\blacktriangleright} = a_1 a_2 ... a_n$ be a topological order of $(A, E^{\text{or}\cup\text{al}}_{\blacktriangleright})$ and $s_{\blacktriangleleft} = b_n b_{n-1} ... b_1$ a reversed topological order of $(A, E^{\text{or}\cup\text{al}}_{\blacktriangleleft})$.

B. For each $a \in A$, define $\mathrm{R}(a)$ as the set of nodes in $A$ reachable from $a$ through edges in $E^{\text{al}}_{\blacktriangleright} \cup E^{\text{al}}_{\blacktriangleleft}$ (i.e., each $b \in \mathrm{R}(a)$ is either $a$ itself or reachable from $a$ in $(A, E^{\text{al}}_{\blacktriangleright} \cup E^{\text{al}}_{\blacktriangleleft})$), and denote $\bar{\mathrm{R}}_{\blacktriangleright}(a)$ and $\bar{\mathrm{R}}_{\blacktriangleleft}(a)$ the two enumerations of $\mathrm{R}(a)$ such that $\bar{\mathrm{R}}_{\blacktriangleright}(a)$ and $\bar{\mathrm{R}}_{\blacktriangleleft}(a)$ are subsequences of $s_{\blacktriangleright}$ and $s_{\blacktriangleleft}$, resp.

C. Let $V_{\text{ns}} \subseteq C$ be the set of alternating constraints that are not satisfied by $s_{\blacktriangleleft} s_{\blacktriangleright}$, and $E_{\text{ns}} \subseteq V_{\text{ns}} \times V_{\text{ns}}$ such that an edge $(X(a,b), Y(c,d))$ is in $E_{\text{ns}}$ iff $c \in \mathrm{R}(b)$, where $X, Y \in \{aRes, aPre\}$ and $a, b, c, d \in A$. Denote $\bar{v}_{\text{ns}}$ to be a topological order of $(V_{\text{ns}}, E_{\text{ns}})$. (It can be shown that $(V_{\text{ns}}, E_{\text{ns}})$ is acyclic)

D. For each edge $aRes(u, v)$ (or $aPre(u, v)$) in $V_{\text{ns}}$ in the order of $\bar{v}_{\text{ns}}$, let $s_{\blacktriangleleft} = s_{\blacktriangleleft} \bar{\mathrm{R}}_{\blacktriangleleft}(v)$ (resp. $s_{\blacktriangleright} = \bar{\mathrm{R}}_{\blacktriangleright}(v) s_{\blacktriangleright}$).

E. Return $s_{\blacktriangleleft} s_{\blacktriangleright}$.

### 3.3 Immediate and Alternating Constraints

Before discussing conformity for schemas with alternating and immediate constraints, we define a "pre-processing" for the given al$^+$schema such that the original al$^+$schema is conformable if and only if after the pre-processing, the schema is conformable.

**Lemma 3.11.** Given an al$^+$schema $S = (A, C)$ that only contains alternating and immediate constraints, the causality graph $(A, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleleft})$ of $S$, and two activities $u, v \in A$ such that there is a path from $v$ to $u$ in the graph $(A, E^{\text{al}\cup\text{im}}_{\blacktriangleright} \cup E^{\text{al}\cup\text{im}}_{\blacktriangleleft})$, then (1) $iRes(u, v) \in C$ implies if a string $s$ satisfies $iRes(u, v)$, then $s \models iPre(v, u)$, and (2) $iPre(u, v) \in C$ implies if a string $s$ satisfies $iPre(u, v)$, then $s \models iRes(v, u)$

Let $u$ and $v$ be as stated in Lemma 3.11. Note that if $u$ and $v$ satisfy the condition in the lemma, $u$ and $v$ will always "occur together" in a conforming string as if they were one activity. With such an observation, we can then pre-process a given schema by "collapsing" such nodes according to in Lemma 3.11. However, two nodes satisfying Lemma 3.11 does not necessarily mean they are "safe" to be collapsed. For example, if nodes $u$ and $v$ in some schema are eligible to be combined based on Lemma 3.11 and there is a node $w$ in the same schema that has constraint $iRes(w, u)$. The collapsing of $u$ and $v$ implies that $iRes(w, v)$ is also a constraint that should be satisfied. According to Theorem 3.5, the schema is not satisfiable. Thus, in the following definition, we define when two nodes are "safe" to collapse (i.e., "collapsable").

**Definition 3.12.** Given an al$^+$schema $S = (A, C)$ that contains only alternating and immediate constraints, and its causality graph $(A, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleleft})$, $S$ is *collapsable* if $S$ satisfies all of the following.

(1). $(A, E^{\text{al}\cup\text{im}}_{\blacktriangleright})$ and $(A, E^{\text{al}\cup\text{im}}_{\blacktriangleleft})$ are acyclic,
(2). for each $(u, v) \in E^{\text{im}}_{\blacktriangleright}$ (or $E^{\text{im}}_{\blacktriangleleft}$), there does not exist $w \in A$ such that $w \neq u$ and $(v, w) \in E^{\text{im}}_{\blacktriangleleft}$ (resp. $E^{\text{im}}_{\blacktriangleright}$),
(3). for each $(u, v) \in E^{\text{im}}_{\blacktriangleright}$ (or $E^{\text{im}}_{\blacktriangleleft}$), there does not exist $w \in A$ such that $w \neq v$ and $(u, w) \in E^{\text{im}}_{\blacktriangleright}$ (resp. $E^{\text{im}}_{\blacktriangleleft}$), and
(4). for each distinct $u, v, w \in A$, if $(u, w), (v, w) \in E^{\text{im}}_{\blacktriangleright}$ or $(u, w), (v, w) \in E^{\text{im}}_{\blacktriangleleft}$, then there is no path from $w$ to either $u$ or $v$ in graph $(A, E^{\text{al}\cup\text{im}}_{\blacktriangleright} \cup E^{\text{al}\cup\text{im}}_{\blacktriangleleft})$.

Note that Conditions (1)–(3) in the above definition are similar to the characterization stated in Theorem 3.5.

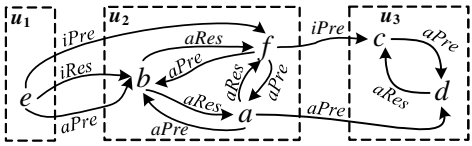


**Fig. 3.** A collapsed schema example

**Example 3.13.** Consider an al$^+$schema with activities $a, b, c, d, e, f$, and constraints shown in Fig. 3 as $(A, E^{\text{al}\cup\text{im}}_{\blacktriangleright} \cup E^{\text{al}\cup\text{im}}_{\blacktriangleleft})$ where the edge labels denote types of constraints. (Ignore the dashed boxes labeled $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3$ for now.) The schema is collapsable. However, if constraint $iPre(a, c)$ is added to the schema, Condition (4) (in the collapsability definition) is violated and thus the new schema is not collapsable, since $(f, c), (a, c) \in E^{\text{im}}_{\blacktriangleleft}$ and there is a path $cda$ from $c$ to $a$ in $(A, E^{\text{al}\cup\text{im}}_{\blacktriangleright} \cup E^{\text{al}\cup\text{im}}_{\blacktriangleleft})$. ∎

**Definition 3.14.** Given a collapsable schema $S = (A, C)$ with only alternating and immediate constraints, the *collapsed schema* of $S$ is a schema $(A', C')$ constructed as follows:

1. Initially $A' = A$ and $C' = C$.
2. Repeat the following steps while $(A', C')$ is changed:
   i. Let $(A', E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleleft})$ be the corresponding causality graph of $(A', C')$.
   ii. for each $u, v \in A$ on a common cycle in $(A, E^{\text{al}\cup\text{im}}_{\blacktriangleright} \cup E^{\text{al}\cup\text{im}}_{\blacktriangleleft})$, If $(u, v) \in E^{\text{im}}_{\blacktriangleright}$ or $E^{\text{im}}_{\blacktriangleleft}$, then (1) remove each $X(u, v)$ or $X(v, u)$ from $C'$, where $X$ ranges over *aRes*, *aPre*, *iRes*, and *iPre*. (2) Create node $w_{uv}$; let $A' := A' - \{u, v\} \cup \{w_{uv}\}$, and (3) replace each $u$ and $v$ in $C'$ by $w_{uv}$.

It is easy to show that given a collapsable al$^+$schema, the corresponding collapsed schema is unique. The following lemma (Lemma 3.15) is easy to verify.

**Lemma 3.15.** Given a collapsable al$^+$schema $S$ with only alternating and immediate constraints, $S$ is conformable iff its collapsed schema is conformable.

By Corollaries 3.8 and 3.15, conformance checking of a schema that only contains alternating and immediate constraints can be reduced to the checking of its collapsed version. Thus, in the remainder of this subsection, we focus on collapsed schemas.

In order to have a clean statement of the necessary and sufficient condition, we introduce a concept of "gap-free". Essentially, "gap-free" is to deal with a special case of a schema illustrated in the following Example 3.16.

**Example 3.16.** Continue with Example 3.13; note that the schema in Fig. 3 is a collapsed schema. Consider a schema $S^{\boldsymbol{u}_2}$ that only contains activities $a$, $b$, and $f$, together with the constraints among them shown in Fig. 3 (i.e., a "subschema" bounded by the dashed box labeled as "$\boldsymbol{u}_2$"). Based on Theorem 3.9, $S^{\boldsymbol{u}_2}$ is conformable and a conforming string is $baf$. Now consider a schema $S^{\boldsymbol{u}_{1,2}}$ that only contains activities $e$, $a$, $b$, and $f$, together with the constraints among them shown in Fig. 3 (i.e., a "subschema" bounded by the dashed boxes labeled as "$\boldsymbol{u}_1$" and "$\boldsymbol{u}_2$" together with the constraints crossing $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$). Due to constraints $iRes(e, b)$ and $iPre(e, f)$, if $S^{\boldsymbol{u}_{1,2}}$ is conformable, then each conforming string of $S^{\boldsymbol{u}_{1,2}}$ must contain substring "$feb$". This requirement leads to some restriction upon schema $S^{\boldsymbol{u}_2}$, i.e., if we take out activity "$e$" from $S^{\boldsymbol{u}_{1,2}}$ and focus on schema $S^{\boldsymbol{u}_2}$ again, one restriction would be: is there a conforming string of $S^{\boldsymbol{u}_2}$ that contains a substring $fb$? If the answer is negative, then apparently, $S^{\boldsymbol{u}_{1,2}}$ is not conformable, since no substring $feb$ can be formed. ∎

With the concern shown in Example 3.16, we need a checking mechanism to decide if two activities can occur as a substring (i.e., "gap-free") in some conforming string. More specifically, given $(A, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleleft})$ as a causality graph of a collapsed schema $S$, we are more interested in checking if two activities that in the same strongly connected component in $(A, E^{\text{al}}_{\blacktriangleright} \cup E^{\text{al}}_{\blacktriangleleft})$ can form a substring in a conforming string of $S$. Note that in Example 3.16, activities $a$, $b$, and $f$ are in the same strongly connected component labeled with $\boldsymbol{u}_2$ in $(A, E^{\text{al}}_{\blacktriangleright} \cup E^{\text{al}}_{\blacktriangleleft})$.

**Definition 3.17.** Let $S = (A, C)$ be a schema that only contains alternating constraints and $(A, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft})$ the causality graph of $S$, such that $(A, E^{\text{al}}_{\blacktriangleright} \cup E^{\text{al}}_{\blacktriangleleft})$ is strongly connected. Given two distinct activities $u, v \in A$, $u$, $v$ are *gap-free* (wrt $S$) if for each $w, x, y \in A$, the following conditions should all hold wrt graph $(A, E^{\text{al}}_{\blacktriangleright})$:

(a). if there is a path $p$ with length greater than 2 from $u$ to $v$, the following all hold:
  (i). if $w$ is on $p$, then $(u, v) \notin E^{\text{al}}_{\blacktriangleright}$,
  (ii). if there is a path from $x$ to $u$, then $(x, v) \notin E^{\text{al}}_{\blacktriangleright}$,
  (iii). if there is a path from $v$ to $y$, then $(u, y) \notin E^{\text{al}}_{\blacktriangleright}$,
  (iv). if there are paths from $x$ to $u$ and $v$ to $y$, and then $(x, y) \notin E^{\text{al}}_{\blacktriangleright}$, and

(b). if there is a path from $v$ to $u$, then the following all hold:
  (i). if there is a path from $x$ to $v$, then $(x, u) \notin E^{\text{al}}_{\blacktriangleright}$,
  (ii). if there is a path from $u$ to $y$, then $(v, y) \notin E^{\text{al}}_{\blacktriangleright}$, and
  (iii). if there are paths from $x$ to $v$ and $u$ to $y$, then $(x, y) \notin E^{\text{al}}_{\blacktriangleright}$.

**Lemma 3.18.** Given a conformable al$^+$schema $S = (A, C)$ that only contains alternating constraints, the causality graph $(A, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft})$ of $S$, such that $(A, E^{\text{al}}_{\blacktriangleright} \cup E^{\text{al}}_{\blacktriangleleft})$ is strongly connected, and two activities $u, v \in A$, "$uv$" can appear as a substring in some a conforming string of $S$ iff $u$, $v$ are gap-free wrt $S$.

Given a graph $G = (V, E)$, for each $v \in V$, denote $\text{SV}(v)$ to be the set of all the nodes in the strongly connected component of $G$ that contains $v$.

Let $(A, C)$ be a collapsed schema and $(A, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleleft})$ its causality graph. Consider graph $(A, E^{\text{al}}_{\blacktriangleright} \cup E^{\text{al}}_{\blacktriangleleft} \cup E^{\text{im}}_{\blacktriangleright} \cup E^{\text{im}}_{\blacktriangleleft})$; given an activity $a \in A$, denote $S(a)$ to be a schema defined as $(\text{SV}(a), \{aRes(u, v) \mid (u, v) \in E^{\text{al}}_{\blacktriangleright} \wedge \text{SV}(a) = \text{SV}(u) = \text{SV}(v)\} \cup \{aPre(u, v) \mid (u, v) \in E^{\text{al}}_{\blacktriangleleft} \wedge \text{SV}(a) = \text{SV}(u) = \text{SV}(v)\})$.

**Example 3.19.** Continue with Example 3.13; consider the schema in Fig. 3. Note that the schema is a collapsed schema. $\text{SV}(a) = \text{SV}(b) = \text{SV}(f)$ is the strongly connected component of the graph in Fig. 3 with nodes $a$, $b$, and $f$. Moreover, $S(a) = S(b) = S(f)$ is a schema that only contains activities $a$, $b$, and $f$, together with the constraints among them in Fig. 3. ∎

The following Theorem 3.20 provides a necessary and sufficient condition for conformability of schema with only alternating and immediate constraints.

**Theorem 3.20.** Given a schema $S$ that only contains alternating and immediate constraints, $S$ is conformable iff the following conditions all hold.

(1). $S$ is collapsable,
(2). $\pi_{\text{al}}(\tilde{S})$ is conformable (recall that $\pi_{\text{al}}$ denotes the "projection" only upon alternating constraints), where $\tilde{S}$ is the collapsed schema of $S$, and
(3). Let $(A, E^{\text{al}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleleft})$ be the causality graph of the collapsed schema $\tilde{S}$, for each $u, v, w \in A$, if there is a path from $u$ to $w$ in $(A, E^{\text{im}}_{\blacktriangleright})$, there is a path from $u$ to $v$ in $(A, E^{\text{im}}_{\blacktriangleleft})$, and $\text{SV}(w) = \text{SV}(v)$ wrt $(A, E^{\text{al}\cup\text{im}}_{\blacktriangleright} \cup E^{\text{al}\cup\text{im}}_{\blacktriangleleft})$, then either (1) $v$, $w$ are gap-free wrt $S(v)$ if $v \neq w$, or (2) $v$ has no outgoing edge in graph $(A, E^{\text{al}\cup\text{im}}_{\blacktriangleright} \cup E^{\text{al}\cup\text{im}}_{\blacktriangleleft})$ if $v = w$.

**Example 3.21.** Continue with Example 3.19; consider the schema in Fig. 3. The schema satisfies the conditions in Theorem 3.20 and is conformable. A conforming string can be $bdacfebdacf$. ∎

Similar to the previous combinations of constraints, given a schema with only ordering and alternating constraints, an algorithm to construct a conforming string is desired. In this case, the algorithm is rather complicated and thus omitted The main idea is that (1) for each activity $a$, construct a string that satisfies each constraint "related" to $a$ as well as each alternating constraint with in a strongly connected component, and (2) hierarchically link these constructed strings together. In this paper, we only provide an example of the algorithm.

**Example 3.22.** Consider the schema shown in Fig. 3. We first construct a string for activity $e$ starting with base $\bar{\text{S}}_{\text{im}}(e) = cfeb$, where $f$ and $b$ are both in strongly connected component $\boldsymbol{u}_2$; while $c$ is in $\boldsymbol{u}_3$. According to the property of gap-free for $f$ and $b$, there must exist a string that satisfies every constraint in $\boldsymbol{u}_2$ and has $fb$ as a substring; a possible string could be: $s_1 = \underline{bafbaf}$. Similarly, string $s_2 = \underline{dc}$ satisfies every constraint in $\boldsymbol{u}_3$ and has $c$ as a substring; then we "glue" the underline parts of $s_1$ and $s_2$ to each end of $\bar{\text{S}}_{\text{im}}(e)$ and have $\underline{badcfebaf}$. Note that this string satisfies every immediate constraint containing $e$ and every alternating constraint within $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, and $\boldsymbol{u}_3$. Further, as there is an alternating precedence constraint from $e$ to $b$, to satisfy that, we "glue" the topological order of $\boldsymbol{u}_2$ before $\underline{badcfebaf}$, and have $\hat{s}(e) = \underline{bafbadcfebaf}$. Note that $\hat{s}(e)$ satisfies every constraint containing $e$ and every alternating constraint within $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, and $\boldsymbol{u}_3$. In general, for each activity, a string $\hat{s}(*)$ is constructed. For example $\hat{s}(b) = b$ and $\hat{s}(a) = \underline{dca}$.

The second step is to link these $\hat{s}(*)$ strings together. The way to link them is first constructing a topological order of all the strongly connected components. For example

in Fig. 3, a topological order is $\boldsymbol{u}_1\boldsymbol{u}_2\boldsymbol{u}_3$. And within each strongly connected component, the order of activities can be arbitrary, for example $ebafcd$. Then, based on the order $ebafcd$, we first replace each $b$ that occurs in the under line parts in $\hat{s}(e)$ by $\hat{s}(b)$, and we have $bafbadcfebaf$. Further, according to the topological order, we replace $a$ that occurs in the under line parts in $bafbadcfebaf$ by $\hat{s}(a)$ and have $bdcafbdcadcfebdcaf$. We repeat these steps and it can be shown that the final string satisfies each constraint. ∎

### 3.4 Response or Precedence Constraints

In this subsection, we study comformity of either response or precedence constraints but not combined. The following Theorem 3.23 states the syntactic condition for conformity of schemas containing only response constraints or only precedence constraints.

**Theorem 3.23.** Given a schema $S = (A, C)$ where $C$ contains only response (or only precedence) constraints, and its causality graph $(A, E^{\text{or}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleright})$ (resp. $(A, E^{\text{or}}_{\blacktriangleleft}, E^{\text{al}}_{\blacktriangleleft}, E^{\text{im}}_{\blacktriangleleft})$), $S$ is conformable iff the following conditions both hold:

(1). $(A, E^{\text{or}\cup\text{al}\cup\text{im}}_{\blacktriangleright})$ (resp. $E^{\text{or}\cup\text{al}\cup\text{im}}_{\blacktriangleleft}$) is acyclic, and

(2). for each $(u, v) \in E^{\text{im}}_{\blacktriangleright}$ (resp. $E^{\text{im}}_{\blacktriangleleft}$), there does not exist any $w \in A$ such that $w \neq v$ and $(u, w) \in E^{\text{im}}_{\blacktriangleright}$ (resp. $E^{\text{im}}_{\blacktriangleleft}$).

**Example 3.24.** Consider a schema $S$ with 5 activities, $a, b, c, d, e$, and 6 constraints $iRes(a, b)$, $iRes(c, e)$, $iRes(e, d)$, $aRes(b, c)$ $Res(a, e)$, and $Pre(b, d)$. Based on Theorem 3.23, $S$ satisfies all conditions, thus conformable. A conforming string can be $abced$. However, if constraint $aRes(d, c)$ is added, $S$ will not be conformable as the edge set forms a cycle (violating Condition (1)). ∎

Alg. 3 is used to construct a conforming string from an input schema that satisfies both conditions in Theorem 3.23. The main idea is again to build a topological order based on the causality graph and then fix each violated immediate constraint in the string. Note that the execution of Alg. 3 replies on Theorem 3.23, where Conditions (1) is to ensure the topological order in Step A is achievable, and Condition (2) is to guarantee Step B1 is unique.

**Algorithm 3.**

Input: A causality graph $(A, E^{\text{or}}_{\blacktriangleright}, E^{\text{al}}_{\blacktriangleright}, E^{\text{im}}_{\blacktriangleright})$ of a schema $S$ that satisfies both conditions in Theorem 3.23

Output: A finite string that conforms to $S$

A. Let string $s$ be a topological order of $(A, E^{\text{or}\cup\text{al}\cup\text{im}}_{\blacktriangleright})$. For each $a \in A$, let $\hat{s}(a)$ be the substring $s^{[k]}s^{[k+1]}...s^{[len(s)]}$ of $s$ such that $s^{[k]} = a$ (clearly $k \in [1..len(s)]$). Let $i = 1$.

B. While $i \leqslant len(s)$, repeat the following step:

B1. If $(s^{[i]}, v) \in E^{\text{im}}_{\blacktriangleright}$ for some $v \in A$ and either $i = len(s)$ or $s^{[i+1]} \neq v$, then replace $s^{[i]}$ in $s$ by $s^{[i]}\hat{s}(v)$.

B2. Increment $i = i + 1$.

C. Return $s$.

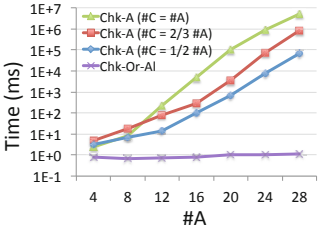


**Fig. 4.** Automata vs Syn. Cond.

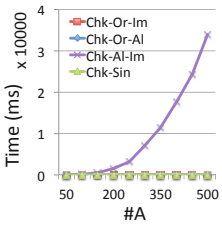


**Fig. 5.** Scalability

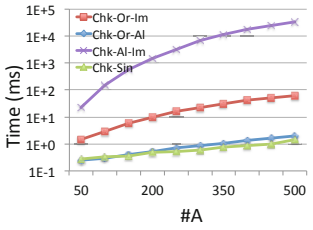


**Fig. 6.** Scalability (log)

## 4 Experimental Evaluations

In this section, several experiments are conducted to evaluate the performance of the syntactic-condition-based conformance checking approaches. Three main types of algorithms are implemented, including: (1) The naive algorithm to check DecSerFlow conformance using automata (denoted as Chk-A), (2) the syntactic-condition-based conformance checking algorithms for all four combinations of predicates (denoted as Chk-Or-Im for ordering and immediate constraints, Chk-Or-Al, Chk-Al-Im, and Chk-Sin for single direction constraints, i.e., either response or precedence), and (3) all four conforming string generation algorithms (denoted as Gen-Or-Im, Gen-Or-Al, Gen-Al-Im, and Gen-Sin). All algorithms are implemented in Java and executed on a computer with 8G RAM and dual 1.7 GHz Intel processors. The data sets (i.e., DecSerFlow schemas) used in experiments are randomly generated. Schema generation uses two parameters: number of activities (#*A*) and number of constraints (#*C*), where each constraint is constructed by selecting a DecSerFlow predicate and two activities in a uniform distribution. Each experiment records the time needed for an algorithm to complete on an input schema. In order to collect more accurate results, each experiment is done for 1000 times to obtain an average time result with the same #*A* and same #*C* for schemas having $\#A < 200$, 100 times for schemas having $\#A \in [200, 400)$, and 10 times for $\#A \in [400, \infty)$. The reason to have less times of experiments for larger #*A* is that it takes minutes to hours for a single algorithm execution with large #*A*, which makes it impractical to run 1000 times. We now report the findings.

***The automata approach is exponentially more expensive than syntactic conditions***

We compared the time needed for the automata and syntactic condition approaches on checking the same set of schemas that contain only ordering and alternating constraints. (For other three types of combinations of constraints, the results are similar). The input schemas have $n$ activities and either $n$, $\frac{n}{2}$, or $\frac{2n}{3}$ constraints, where $n$ ranges from 4 to 28. Fig. 4 shows the results (x-axis denotes the number of activities and y-axis denotes the time needed in the log scale). It can be observed that for the automata approach, the time needed is growing exponentially wrt the number of activities/constraints. For a schema with 28 activities and 28 constraints, it takes more than 3 hours to finish the checking. However, the syntactic condition approaches (whose complexity is polynomial) can finish the conformance checking almost instantly. As the times needed for either $n$, $\frac{n}{2}$, or $\frac{2n}{3}$ constraints are all too close around 1ms, we only use one curve (instead of three) in Fig. 4 to represent the result for the syntactic conditions approach.

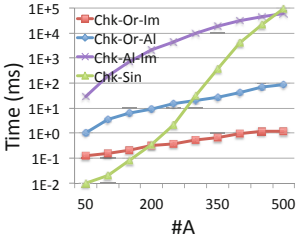


**Fig. 7.** String Generation

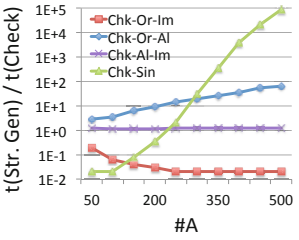


**Fig. 8.** Str. Gen. / Checking

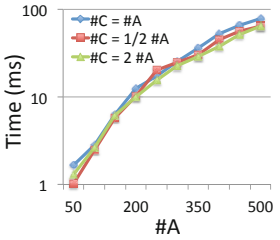


**Fig. 9.** Changing #Constraints

***The syntactic conditions approaches have at most a cubic growth rate in the size of the input schemas***

We compute the times needed for the syntactic condition approaches for input schemas with $n$ activities and $n$ constraints, $n$ between 50 and 500. Fig. 5 and 6 show the same result with normal and logarithm scales (resp.) of all four combinations of the constraints. From the result, the complexity of the syntactic condition approach for alternating and immediate constraints appears cubic due to the checking of Condition (4) of Definition 3.12 (collapsable); the complexity for ordering and immediate constraints is quadratic due to the pre-processing to form an $\text{im}^+$schema; the complexity for ordering and alternating constraints is linear as the pre-processing (to form an $\text{al}^+$schema by detecting strongly connected components) as well as the acyclicity check of the causality graphs are linear; finally, the complexity for the constraints of a single direction is also linear.

***Conforming string generation requires polynomial to exponential times***

With the same experiment setting as above, Fig. 7 shows the time to generate a conforming string for a conformable schema. From the results, all string generating approaches are polynomial except for the single direction case (i.e, either response or precedence). According to Alg. 3, the length of a generated string can be as long as $2^n$, where $n$ is the number of activities in the given schema. Fig. 8 presents the ratios of the time to generate a conforming string over the time to check conformance of the same schema for conformable schemas. The results indicate that the complexity to generate a string can be polynomially lower (ordering and immediate case), the same (alternating and immediate case), polynomially higher (ordering and alternating case), and exponentially higher (single direction case) than the corresponding complexity to check conformance of the same schema. Note that the curves in Fig. 8 is lower or "smaller" than dividing "Fig. 7" by "Fig. 5" due to the reason that the data shown in Fig. 7 is only for the conformable schemas; while the one in Fig. 5 is for general schemas, where non-conformable schemas can be determined 5 - 15% faster than conformable ones due to the reason that a non-comformable schema fails the checking if it does not satisfy one of the conditions (e.g., in Theorem 3.5, there are three conditions to check); while a comformable schema can pass the check only after all conditions are checked.

***Increasing the number of constraints increases more time for the automata approach than syntactic condition approaches***

We compute the time needed for the syntactic condition approaches with input schemas containing only ordering and immediate constraints with $n$ activities and either $n$, $2n$,

or $\frac{n}{2}$ constraints, where $n$ ranges from 50 to 500. (For other three types of combinations of constraints, the results are similar). Fig. 9 shows the three curves for $n$, $2n$, and $\frac{n}{2}$ constraints respectively. Comparing the similar settings shown in Fig. 4, there does not exist an obvious growth in time when the number of constraints grow and the curves are almost the same. The reason is that the algorithms we used to check conformance and generate strings are graph-based approaches. As $\#C \in [\frac{\#A}{2}, 2\#A]$, we have $O(\#C) = O(\#A)$ that can provide the same complexity. Moreover, if $\#C < \frac{\#A}{2}$, there will be activities involving in no constraint, which leads to a non-practical setting; if $\#C > 2\#A$, almost all the randomly generated schemas will be non-confomable based on uniform distribution.

## 5 Related Work

The work reported here is a part of the study on collaborative systems and choreography languages [16]. The constraint language studied is a part of DecSerFlow [2], whose constraints can be translated to LTL [13].

Original LTL [13] is defined for infinite sequences. [15] proved that LTL satisfiability checking is PSPACE-Complete. A well-know result in [17] shows that LTL is equivalent to Büchi automata; and the LTL satisfiability checking can be translated to language emptiness checking. Several complexity results on satisfiability developed for subsets of LTL. [5] shows that restriction to Horn formulas will not decrease the complexity of satisfiability checking. [6] investigates the complexity of cases restricted by the use of temporal operators, their nesting, and number of variables. [4] and [3] provide upper and lower bounds for different combinations of both temporal and propositional operators. [7] presents the tractability of LTL only with combination of "XOR" clauses.

For the finite semantics, [8] studies the semantics of LTL upon truncated paths. [10] provides an exponential-time algorithm to check if a given LTL formula can be satisfied by a given finite-state model, but the execution is still infinite.

Business process modeling has been studied variously in the last decade ([9,1]). Previous studies of declarative models focus mostly on formal verification of general properties involving data, generally, such verification problems have exponential or higher time complexity (see [9]).

## 6 Conclusions

This paper studied syntactic characterization of conformance for "core" DecSerFlow constraints that are reduced from general DecSerFlow constraints. We provided characterizations for (1) ordering and immediate constraints, (2) ordering and alternating constraints, (3) alternating and immediate constraints, and (4) ordering, alternating, and immediate constraints with precedence (or response) direction only. The general case for ordering, immediate, and alternating constraints with both precedence and response directions remains as an open problem; furthermore, it is unclear if the conformance problem for DecSerFlow constraints is in PTIME.

## References


1. van der Aalst, W.M.P.: Business process management demystified: A tutorial on models, systems and standards for workflow management. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Lectures on Concurrency and Petri Nets. LNCS, vol. 3098, pp. 1–65. Springer, Heidelberg (2004)
2. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
3. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: The complexity of clausal fragments of LTL. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR-19 2013. LNCS, vol. 8312, pp. 35–52. Springer, Heidelberg (2013)
4. Bauland, M., Schneider, T., Schnoor, H., Schnoor, I., Vollmer, H.: The complexity of generalized satisfiability for linear temporal logic. In: Seidl, H. (ed.) FOSSACS 2007. LNCS, vol. 4423, pp. 48–62. Springer, Heidelberg (2007)
5. Chen, C.C., Lin, I.P.: The computational complexity of satisfiability of temporal horn formulas in propositional linear-time temporal logic. Inf. Proc. Lett. 45(3), 131–136 (1993)
6. Demri, S., Schnoebelen, P., Demri, S.E.: The complexity of propositional linear temporal logics in simple cases. Information and Computation 174, 61–72 (1998)
7. Dixon, C., Fisher, M., Konev, B.: Tractable temporal reasoning. In: Proc. International Joint Conference on Artificial Intelligence (IJCAI). AAAI Press (2007)
8. Eisner, C., Fisman, D., Havlicek, J., Lustig, Y., McIsaac, A., Van Campenhout, D.: Reasoning with temporal logic on truncated paths. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 27–39. Springer, Heidelberg (2003)
9. Hull, R., Su, J., Vaculín, R.: Data management perspectives on business process management: tutorial overview. In: SIGMOD Conference, pp. 943–948 (2013)
10. Lichtenstein, O., Pnueli, A.: Checking that finite state concurrent programs satisfy their linear specification. In: POPL, pp. 97–107 (1985)
11. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: Declare: Full support for loosely-structured processes. In: EDOC, pp. 287–300 (2007)
12. Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.A.: Imperative versus declarative process modeling languages: An empirical investigation. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 383–394. Springer, Heidelberg (2012)
13. Pnueli, A.: The temporal logic of programs. In: FOCS, pp. 46–57 (1977)
14. Silva, N.C., de Carvalho, R.M., Oliveira, C.A.L., Lima, R.M.F.: REFlex: An efficient web service orchestrator for declarative business processes. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 222–236. Springer, Heidelberg (2013)
15. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. J. ACM 32(3), 733–749 (1985)
16. Sun, Y., Xu, W., Su, J.: Declarative choreographies for artifacts. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) Service Oriented Computing. LNCS, vol. 7636, pp. 420–434. Springer, Heidelberg (2012)
17. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification (preliminary report). In: LICS, pp. 332–344 (1986)
18. Xu, W., Su, J., Yan, Z., Yang, J., Zhang, L.: An Artifact-Centric Approach to Dynamic Modification of Workflow Execution. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 256–273. Springer, Heidelberg (2011)