SUMMARY In this paper, we discuss a fundamental theory of incremental verification for workflows. Incremental verification is a method to help multiple designers share and collaborate on huge workflows while maintaining their consistency. To this end, we introduce passbacks in workflows and their consistency property in the control flow perspective. Passbacks indicate redoing of works. Workflows with passbacks are useful to naturally represent human works. To define the consistency property above, we define normality of workflows with passbacks and total correctness of normal workflows based on transition system-based semantics of normal workflows. We further extend workflows to sorted workflows and define their vertical division and composition. We also extend total correctness to normal sorted workflows, for the sake of incremental verification of a large-scale workflow with passbacks via vertical division and composition.

**key words:** workflow, verification, correctness, passback, incremental verification

1. Introduction

We have not yet established a proper method that enables multiple designers to share the development of huge workflows while keeping consistency of them, although such a method is necessary for development of large scale information systems. The reason is that such a method needs to verify consistency of parts of a huge workflow before integrating them into the whole workflow, but, almost of all existing verification methodologies [1], [4], [6], [9], [10] cannot deal with such verification. We call such verification “incremental verification”. In this paper, we introduce a fundamental theory of incremental verification of correctness of workflows, where correctness of a workflow denotes a consistency property in the control flow perspective. Correctness of a workflow is an important property that ensures that, as far as a work progresses according to the workflow, the work will not be dead-locked or it will not have any unexpected duplication of the work.

To realize incremental verification, one needs proper workflows to be dealt with and a suitable way to divide them. Moreover, we need to consider proper correctness property of the workflows above, which should be preserved via the division above. To this end, we first introduce workflows with passbacks, where passbacks represent redoing of works in workflows. Workflows with passbacks are useful to represent human workflows naturally. In fact, we have been gathering over 500 workflow diagrams used in requirement analysis of large scale information systems, which can be described as workflows with passbacks [11], [13], [14]. Since a passback (flow) makes a loop in a workflow, a workflow that describes passbacks is a kind of a cyclic workflow. However, since existing semantics including petri net does not give simple and natural meaning to passbacks, we cannot use existing correctness of cyclic workflows for workflows with passbacks. Thus, in this paper, we define correctness property of workflows with passbacks by defining normality of them and total correctness of normal workflows based on transition system-based semantics of normal workflows.

In this paper, we also define sorted workflows that play the roles of subgraphs of workflows with passbacks, and we define vertical division and composition of sorted workflows. We also extend normality and total correctness to sorted workflows, and explain the fundamental properties including the preservation property of total correctness via vertical division and composition, by which one can find problematic points in small workflows before he/she integrate them to a large-scale workflow. Moreover, to demonstrate usefulness of the fundamental properties above, we provide a desktop evaluation of them. To this end, we use a large scale workflow that was developed in real information system development.

The remainder of this paper is organized as follows. Section 2 explains conventional workflows. Section 3 introduces passbacks, normality of workflows with passbacks, semantics based on transition systems and total correctness of normal workflows. Section 4 introduces sorted workflows and vertical composition and division of them. It also extends normality and total correctness to sorted workflows and explains the fundamental properties of the extended total correctness. Section 5 explains a desktop evaluation of the fundamental properties above. In the last two sections, related works and conclusions are explained.

2. Preliminaries

In this section, we first define workflows, which are essentially the same as those in previous studies such as [6], [9], [10]. Here we discuss workflows only on the control flow perspective. Therefore, we omit notions that are not relevant to control flow of workflows.

**Definition 2.1** A workflow W denotes a directed graph (Nodes, Arcs) that satisfies the following properties.
1. *Nodes* is a non-empty finite set whose element is called a *node* in *W*, while *Arcs* is a non-empty finite set whose element is called an *arc* in *W*.

2. Each arc *f* is assigned to a node called a *source of* *f* and another node called a *target of* *f*.

3. Each node is categorized as one of the following: *trigger*, *terminal*, *activity*, *XOR-split*, *XOR-join*, *AND-split* and *AND-join*. We employ the symbols in Fig. 1 to describe nodes in a workflow.

4. Whenever an arc *f* has a node *x* as the target (or the source) of *f*, *x* has *f* as an *incoming-arc* (resp. an *outgoing-arc*) of *x*.

5. The numbers of incoming-arcs and outgoing-arcs of a node are determined by the type of the node. We itemize them in Table 1.

6. *W* has the unique trigger and at least one terminal.

7. For any node *x* in *W*, there exists a path on *W* from the trigger *t* to *x*, where a *path* from *t* to *x* denotes the subgraph of *W* generated from a sequence (*f₀, f₁, ..., fₙ*) of arcs in *W* such that the source of *fᵢ* is *t* and the target of *fᵢ* is *x* for each *i* < *n*. Moreover, there exists another path on *W* from *x* to a terminal.

**Definition 2.2** (1) For a subgraph *G* of a workflow, *Nodes*(*G*) and *Arcs*(*G*) denote the set of all nodes in *G* and the set of all arcs in *G*, respectively.

(2) For an arc *f*, *src*(*)f*) and *tg*(*)f*) denote the source and the target of *f*, respectively.

(3) Let *G* be a subgraph of a workflow and *n* a node in *G*. If *G* contains no incoming-arc of *n*, then *n* is called a *starting node* of *G*. If *G* contains no outgoing-arc of *n*, then *n* is called an *ending node* of *G*.

Note that the trigger of a workflow *W* is the unique starting node of *W* and that a terminal of *W* is an ending node of *W*.

Here we define an important subgraph of a workflow, as follows.

**Definition 2.3** A subgraph *G* of a workflow *W* is called a *pre-instance* of *W* if *G* contains the trigger *t* of *W* and for each *n* ∈ *Nodes*(*G*) *G* includes a path from *t* to *n*. *PI*(*W*) denotes the set of all pre-instances of *W*.

The smallest pre-instance of a workflow *W* is the subgraph consisting of the trigger of *W* only. Hereinafter, *π*ₚ denotes the smallest pre-instance above.

3. Normal Workflow and Total Correctness over Workflows with Passbacks

In this section, we introduce workflows with passbacks and their consistency property in the control flow perspective, by defining normality and transition systems.

3.1 Normal Workflow

**Definition 3.1** A path on a workflow *W* with no twists from the trigger to a terminal is called a *main path* of *W*. Here, the path (*f₁, f₂, ..., fₙ*) is said to have “no twists” if, for arbitrary *i, j* ∈ *n* with *i ≠ j*, *tg*(*)fᵢ*) ≠ *tg*(*)fⱼ*).

**Definition 3.2** Let *f* be an arc in a workflow *W*. If *f* is contained in a main path of *W*, *f* is called a *main arc* in *W*. Otherwise, *f* is called a *passback arc* in *W*.

**Definition 3.3** The *main subgraph* of a workflow *W* is the subgraph composed of all main arcs in *W*. *MW* denotes the main subgraph of *W*. Meanwhile, the *passback subgraph* of *W* is the subgraph composed of all passback arcs in *W*, which is denoted by *PW*.

Note that the main subgraph of any workflow is connected as a graph.

**Example 3.4** The workflow in Fig. 2 describes a process of prescription in a hospital.

In the first step, a doctor carries out a medical examination and writes a prescription for a patient. Then, a pharmacist receives a prescription and checks it. If he/she finds a mistake in the prescription, then he/she sends back the prescription to the doctor (the first passback denoted by *f₁*). If the pharmacist finds no mistake in the prescription, he/she fills the prescription. Then, another pharmacist receives a prescription and medicines and checks them. If he/she finds a mistake in them, he/she sends back them to the first pharmacist (the second passback denoted by *f₂*). If the second...
The workflow in Fig. 2 has the main subgraph consisting of all arcs by a black color, and the passback subgraph consisting of $f_1$ and $f_2$ that are double-lined arcs by a red color.

**Definition 3.5** Let $W$ be a workflow and $G$ an acyclic subgraph of $W$. Then, an instance $G$ denotes a subgraph $V$ of $G$ that satisfies the following properties.

1. $V$ contains just one node $s$ of the starting nodes of $G$.
2. For a node $x$ in $V$, there is a path on $V$ from $s$ to $x$.
3. If $V$ contains an XOR-split, then $V$ contains just one outgoing-arc of $x$ on $G$.
4. If $V$ contains a node $x$ other than XOR-split, then $V$ contains all outgoing-arcs of $x$ on $G$.

**Definition 3.6** When $P_W$ is acyclic, an instance of $P_W$ is called a passback instance. When $M_W$ is acyclic, an instance of $M_W$ is called a main instance.

**Example 3.7** The workflow in Fig. 2 has two passback instances. One consists of just $f_1$, and another consists of just $f_2$. On the other hand, the workflow has the unique main instance that is the same as the main subgraph of the workflow.

**Definition 3.8** For each passback instance $I$ of a workflow with passback subgraph acyclic, the outgoing-arc of the starting point on $I$ is called the starting-passback arc (on $I$), and the incoming arc of an ending point on $I$ is called an ending-passback arc (on $I$).

Note that it is possible that some arc is a starting-passback arc as well as an ending-passback arc at the same time.

**Definition 3.9** The binary relation $\rightarrow_W$ on the set of nodes of a workflow $W$ is defined as follows: For arbitrary nodes $n, m, n \rightarrow_W m$ if there exists a path (of length $\geq 0$) from $n$ to $m$ that does not contain any ending-passback arc.

This definition directly implies the following lemma.

**Lemma 3.10** $\rightarrow_W$ is a preorder on Nodes($W$), that is, for each $x, y, z \in$ Nodes($W$),

i. $x \rightarrow_W x$, and

ii. $(x \rightarrow_W y$ and $y \rightarrow_W z)$ implies $x \rightarrow_W z$.

**Definition 3.11** A workflow $W$ is called a normal workflow if $W$ satisfies the following conditions.

1. Both $M_W$ and $P_W$ are acyclic.
2. Each starting point of $P_W$ (as an acyclic graph) is the XOR-split node on $M_W$.
3. Each ending point of $P_W$ (as an acyclic graph) is the XOR-join node on $M_W$.
4. $P_W$ does not have any AND-join node.
5. For a passback instance $I$ and ending-passback arcs $p_1$ and $p_2$ of $I$, we do not have $tg(p_1) \rightarrow_W tg(p_2)$.
6. For a main instance $I_M$ and a passback instance $I_p$, if $I_M$ contains the starting node of $I_p$, then $I_M$ contains all ending nodes of $I_p$.

**Example 3.12** The workflow in Fig. 2 is normal.

Here, we will review the conditions (1), . . . , (6) in Definition 3.11 based on our observation of actual workflows that we accumulated.

(1) comes from the observations, as follows:

i. Most of the actual workflows can be classified into some parts representing usual procedures and others representing passback procedures.

ii. A usual procedure has no loop or very simple loops that can be represented by passbacks, and

iii. A passback procedure has no loop.

(2) and (3) come from the observation that usual work and passback work never synchronously fork or rendezvous together. In fact, if we consider that passback work means to trace back over previous work, it is reasonable to assume that any synchronous forking or rendezvous of usual work and passback work is unlikely.

(4) means that we do not consider synchronization of multiple flows of passbacks. Recognizing this synchronism is the same as recognizing the synchronism of generally divided elements in the workflow; a further mechanism is necessary in order to achieve this.

(5) ensures that after restoration from passback work, the flow of a certain work does not run into the flow of other work. For example, look at the passback arc of the workflow on the left side in Fig. 3. We do not regard the workflow as a normal one. We see that $tg(p_1) \rightarrow_W tg(p_2)$ showing that, after the passback work, the arc of work from $tg(p_1)$ might run into the arc of work from $tg(p_2)$. This form of workflow, whereby the arcs of work meet after the passback work, is not considered appropriate. In fact, if $tg(p_1) \rightarrow_W tg(p_2)$ is true, then it is proper to think that $p_2$ should be omitted.

(6) ensures that passback work traces back over previous work. In other words, this condition’s purpose is to ensure that no matter the path by which the original work is performed, the nodes that are returned by passback are already performed by a certain point before entering into passback. For example, look at the workflow on the right of Fig. 3. We do not regard the workflow as a normal one. If the person who created the application is not senior then it results in passback, in going back to previously carried out...
work. However, if the person who created it is senior, then it results in passback, in moving to different work, namely “make a proposal with your senior student” and not back to previous work. So, this kind of workflow is not considered appropriate as a normal one.

The following lemmas hold immediately.

**Lemma 3.13** Every acyclic workflow is normal.

**Lemma 3.14** For a normal workflow $W$, $\rightarrow_W$ is a partial order on Nodes($W$). That is, $\rightarrow_W$ is a preorder and it satisfies that, for each $x, y, z \in \text{Nodes}(W)$,

$$(x \rightarrow_W y \text{ and } y \rightarrow_W x) \implies x = y.$$ 

### 3.2 Semantics of Normal Workflows

Here, we define transition systems over normal workflows.

**Definition 3.15** For a normal workflow $W$, we define a binary relation $T_W$ on $\text{PI}(W)$, as follows. For each pre-instances $H$ and $H'$ of $W$, $(H, H') \in T_W$ if there exists an arc $f$ that satisfies the following property (A) or (B).

A) $f$ satisfies the following properties (1)-(5).

1. $f \notin \text{Arcs}(H)$ but $\text{src}(f) \in \text{Nodes}(H)$.
2. If $\text{src}(f)$ is an XOR-split, then $\text{Arcs}(H)$ contains no outgoing-arc of $\text{src}(f)$.
3. If $\text{src}(f)$ is an AND-join, then $\text{Arcs}(H)$ contains all incoming-arcs of $\text{src}(f)$.
4. For any passback instance $I$ that contains $f$, there exists an ending-passback arc $g$ of $I$ with $g \notin \text{Arcs}(H) \cup \{f\}$.
5. $\text{Arcs}(H') = \text{Arcs}(H) \cup \{f\}$

B) $f$ satisfies the properties (1)-(3) above. Moreover, there exists a passback instance whose all ending-passback arcs $f_1, \ldots, f_n$ satisfying the following properties (4')-(6').

4'. $f_1 = f$ and $tg(f_1) \in \text{Nodes}(H)$.
5'. $\{f_2, \ldots, f_n\} \subseteq \text{Arcs}(H)$.
6'. $\text{Arcs}(H') = \{f \in \text{Arcs}(W)|\forall i \leq n\ mim(f_i) \rightarrow_W \text{src}(f)\}$.

We can view $(\text{PI}(W), T_W)$ as a transition system. Hereinafter, we represented $(H, H') \in T_W$ by $H \rightarrow_{T_W} H'$

For pre-instances $H$ and $H'$, if there are $H_1, \ldots, H_n \in \text{PI}(W)$ with $H = H_1 \rightarrow_{T_W} H_2 \rightarrow_{T_W} \ldots \rightarrow_{T_W} H_n = H'$, or if $H = H'$, then $H'$ is said to be accessible from $H$ (and $H$ is said to be accessible to $H'$) and it is represented by $H \rightarrow_{T_W} H'$. Moreover, $\text{AC}(W)$ denotes the set of pre-instances of $W$ that are accessible from $\pi_W$.

**Example 3.16** Let $H_1$ be a pre-instance in Fig. 4 that consists of double-lined arcs by a red color. Each pre-instance describes which activities have been performed. For example, $H_1$ describes that the activity of auditing medicines has been performed. If the result of the audit is “no”, then, via $f_3$, $H_1$ is translated to $H_2$ in Fig. 5 that consists of double-lined arcs by a red color.

### 3.3 Total Correctness of Normal Workflows

**Definition 3.17**

1. A pre-instance $H$ of a normal workflow $W$ is said to be deadlock free if, for an AND-join $n$ on $H$ that is not any ending node on $H$, $\text{Arcs}(H)$ contains all incoming-arcs of $n$.

2. A pre-instance $H$ is said to be lack of synchronization free if, there exists no XOR-join $n$ on $H$ satisfying that $\text{Arcs}(H)$ contains more than one incoming-arc of $n$ except ending-passback arcs or more than one incoming-arc of $n$ that is an ending-passback arc.

Here we define total correctness of normal workflows.

To simplify the definition of total correctness below, we consider an additional condition to the definition of normal workflows, as follows.

**XOR-join condition:** No normal workflow contains an XOR-join that has multiple ending-passback arcs.

The condition above gives no essential limitation to the definition of normal workflows. In fact, if there is an XOR-join that has multiple ending-passback arcs, one has only to
add a new XOR-join to bundle them into a single ending-passback arc.

**Definition 3.18** A normal workflow W is said to be **totally correct** if W satisfies the following properties.

1. For any $H \in AC(W)$, there exists $H_e$ satisfying the following properties (i)-(iv).
   i. $H \rightarrow^{TW} H_e$.
   ii. $H_e$ contains at least one terminal of W.
   iii. $H_e$ contains no ending node except the terminal of W.
   iv. $H_e$ contains no passback-arc.

2. Every $H \in AC(W)$ is deadlock free and lack of synchronization free.

Here, we compare the existing correctness property of acyclic workflows [2], [6], [9], [10] with the total correctness above. We here introduce the existing correctness property, as follows.

**Definition 3.19** An acyclic workflow W is said to be **correct** if every main instance V of W is deadlock free and lack of synchronization free.

Existing correctness property for acyclic workflows is important to verify consistency of workflows in the control flow perspective. Theorem 3.19 claims that total correctness is an extension of existing correctness, that is, total correctness is obtained from existing correctness by extending the range of application of it to workflows with passbacks by keeping the same property for workflows in the original range.

**Theorem 3.20** For any acyclic workflow W, W is totally correct if and only if W is correct.

Proof. See Appendix A.

Aalst et al. [6] prove the equivalency between soundness based on the Petri-net interpretation of workflows for an acyclic workflow and correctness in Definition 3.5. Therefore, Theorem 3.20 shows that the total correctness of an acyclic workflow is clearly equivalent to soundness.

We next explain that total correctness of a normal workflow amounts to the correctness of acyclic subgraphs of the workflow.

**Theorem 3.21** (Locally Representable Property of Total Correctness). For any normal workflow W, W is totally correct if and only if W satisfies the following properties.

1. $M_W$ is correct ($\Leftrightarrow$ totally correct) as an acyclic workflow.
2. For any passback instance I of W, I is deadlock free and lack of synchronization free.

Proof. See Appendix B.

By the theorem above, verifying total correctness of a normal workflow does not require calculation of the transition system for the workflow. On the other hand, normality of a workflow is essentially determined by the question of how the main instance and passback instances of the workflow are linked (see Definitions 3.11 (5) and (6)). Consequently, verifying normality and total correctness of an arbitrary workflow can be essentially accomplished by calculating the amount of extraction of the main instance and passback instances of the workflow. The fact provides a benefit in the incremental creation of large-scale workflows. In fact, we use Theorem 3.21 to prove theorems in the next section, though the proofs are omitted in this paper.

4. Sorted Workflows and Extended Total Correctness

In this section, we define extended versions of workflows with passbacks and extend total correctness to them, and then we introduce fundamental properties of the extended total correctness and explain their usefulness for incremental verification for workflows with passbacks.

4.1 Sorted Workflows

**Definition 4.1** A sorted workflow W denotes a directed graph (Nodes, Arcs) that satisfies the conditions obtained from Definition 2.1 by replacing the conditions 6 and 7 of Definition 2.1 with the following conditions.

6’. W has at least one trigger and at least one terminal.
7’. For a node $x$ in W, there exists a trigger $s$ having a path on W from $s$ to $x$, and a terminal $e$ having another path on W from $x$ to $e$.

Moreover, a sorted workflow has additional informations and conditions, as follows.

8’. Each arc in W has a single label “main” or “passback”.
9’. Let $M^L_W$ be the subgraph of W that consists of all arcs with label “main” and $P^L_W$ the subgraph of W that consists of all arcs with label “passback”. Then, $M^L_W$ and $P^L_W$ satisfy the following properties.

i. $M^L_W$ has at least one trigger of W and at least one terminal of W. Conversely, each starting node of $M^L_W$ is a trigger of W and each ending node of $M^L_W$ is a terminal of W.
ii. Each arc $f$ in $M^L_W$ is contained in a main path on $M^L_W$, that is, a path with no twist from the trigger in $M^L_W$ to the terminal in $M^L_W$ (cf. Definition 3.1 (1)). Conversely, each path with no twists from the trigger of $M^L_W$ to the terminal of $M^L_W$ is contained in $M^L_W$.

An arc with label “main” is called a main-sorted arc, while an arc with label “passback” is called a passback-sorted arc.

**Example 4.2** The directed graph in Fig. 6 is a sorted workflow, where each arc indicated in Fig. 6 by a black color is a...
main-sorted arc, while each arc indicated in Fig. 6 by a red color and a double line is a passback-sorted arc.

**Lemma 4.3** Each workflow can be regarded as a sorted workflow by labeling each main arc by “main” and each passback arc by “passback”. Moreover, each sorted workflow \( W \) with the unique trigger can be regarded as a workflow with \( M_w = M^L_w \) and \( P_w = P^L_w \).

For an acyclic subgraph \( G \) of a sorted workflow, let \( \text{INS}(G) \) be the set of all instances of \( G \). Moreover, for a starting node \( s \) of \( G \), let \( \text{INS}(G,s) \) be the set of instances of \( G \) with starting node \( s \).

**Definition 4.4** Let \( W \) be a sorted workflow and \( G \) an acyclic subgraph of \( W \).

1. For \( U_1, U_2 \in \text{INS}(G) \), \( U_1 \) and \( U_2 \) are said to conflict on an XOR-split \( c \) if \( U_1 \) and \( U_2 \) share \( c \) but the outgoing-arc of \( c \) in \( U_1 \) differs from that in \( U_2 \).
2. Let \( U \) be a set of some instances of \( G \) and \( c \) an XOR-split. Then, \( U \) is said to conflict on \( c \) if there exists a pair \((U, U') \) on \( U \) that conflicts on \( c \).
3. Let \( S \) be a set \( \{s_1, \ldots, s_n\} \) of starting nodes of \( G \) and let \( U_i \in \text{INS}(G,s_i) \) for \( i = 1, \ldots, n \). Moreover, assume that \( \{U_1, \ldots, U_n\} \) is not conflict on any XOR-split in \( G \). Then, \( U_1 \cup \cdots \cup U_n \) is called an extended instance of \( G \) from \( S \).

**Definition 4.5** Let \( W \) be a sorted workflow whose \( M^L_w \) and \( P^L_w \) are acyclic.

1. An (extended) instance of \( P^L_w \) is called an extended passback instance of \( W \). An (extended) instance of \( M^L_w \) is called an extended main instance of \( W \).
2. For each (extended) passback instance \( I \) of \( W \), the outgoing-arc of a starting node of \( I \) is called a starting-passback-sorted arc on \( I \), and the incoming arc of an ending point on \( I \) is called an ending-passback-sorted arc on \( I \).
3. An (extended) instance \( I \) of \( W \) is said to be consistent if \( I \) is deadlock free and not synchronization free, that is, for each AND-join \( r \) in \( I \), \( I \) contains all incoming-arcs of \( r \), and for each XOR-join \( m \) in \( I \), \( I \) contains a single incoming-arc of \( m \).
4. An extended main (or passback) instance \( I \) of \( W \) is said to be maximal if no extended main (or passback, respectively) instance \( J \) of \( W \) satisfies that \( I \subseteq J \).
5. The binary relation \( \rightarrow_w \) on \( \text{Nodes}(W) \) is defined as follows: \( n \rightarrow_w m \) if there exists a path from \( n \) to \( m \) that does not contain any ending-passback-sorted arc.

### 4.2 Normality of Sorted Workflows

We here extend normality of workflows (cf. Definition 3.11) into sorted workflows.

**Definition 4.6** Let \( W \) be a sorted workflow, \( \{u_1, \ldots, u_n\} \) the set of all triggers in \( P^L_w \) that are not contained in \( M^L_w \), and \( \{v_1, \ldots, v_m\} \) the set of all terminals in \( P^L_w \) that are not contained in \( M^L_w \). Then a graph obtained from \( W \) by the following process is called a closure of \( W \) and denoted by \( \text{Cl}(W) \).

1. For each \( u_i \) \( (i = 1, \ldots, n) \), add an XOR-split \( u_i \) to an incoming allow of a terminal of \( M^L_w \).
2. For each \( v_j \) \( (j = 1, \ldots, m) \), consider a new XOR-join \( v_j \) and add it to an outgoing allow of a trigger of \( M^L_w \).

**Notation 4.11** Let \( W \) be a sorted workflow. Then, \( \text{S}_M(W) \) denotes the set of all triggers in \( M^L_w \).
2. $S_p(W)$ denotes the set of all triggers in $P^L_W$.
3. $E_M(W)$ denotes the set of all terminals in $M^L_W$ and
4. $E_p(W)$ denotes the set of all terminals in $P^L_W$.

**Definition 4.12** Let $W_1$ and $W_2$ be sorted workflows, and $E^M_1 \subset E_M(W_1)$, $S^M_2 \subset S_M(W_2)$, $E^P_2 \subset E_p(W_2)$ and $S^P_1 \subset S_p(W_1)$. Moreover, assume that there exists bijections $f$ from $E^M_1$ to $S^M_2$ and $g$ from $E^P_2$ to $S^P_1$. Then, $W_1 \times f, g W_2$ denotes the sorted workflow obtained from $W_1$ and $W_2$ by executing the following procedures.

1. Connecting $M^L_{W_1}$ and $M^L_{W_2}$.
   1.1 Remove all terminals in $E^M_1$ and their incoming-arcs.
   1.2 Remove all triggers in $S^M_2$ and their outgoing-arcs.
   1.3 For the source $x$ of the incoming-arc of each terminal $e_1 \in E^M_1$ and the target $y$ of the outgoing-arc of each trigger $f(e_1) \in S^M_2$, add the arc from $x$ to $y$.

2. Connecting $P^L_{W_1}$ and $P^L_{W_2}$.
   2.1 Remove all terminals in $E^P_2$ and their incoming-arcs.
   2.2 Remove all triggers in $S^P_1$ and their outgoing-arcs.
   2.3 For the source $z$ of the incoming-arc of each terminal $e_2 \in E^P_2$ and the target $w$ of the outgoing-arcs of each trigger $g(e_2) \in S^P_1$, add the arc from $z$ to $w$.

$W_1 \times f, g W_2$ is called the vertical composition of $W_1$ and $W_2$ by $f$ and $g$.

For simplicity, we omit "$f$" and "$g$" in $W_1 \times f, g W_2$ and identify each $e_1 \in E^M_1$ with $f(e_1) \in S^M_2$ and each $e_2 \in E^P_2$ with $g(e_2) \in S^P_1$.

**Example 4.13** Figure 8 indicates a vertical composition of sorted workflows $W_1$ and $W_2$, where blue dotted arrows denote a function from $E^M_1$ to $S^M_2$ and a red dotted arrow indicates another function from $E^P_2$ to $S^P_1$.

**Definition 4.14** For a sorted workflow $W$, if there exist sorted workflows $W_1$ and $W_2$ with $W = W_1 \times W_2$, then $W$ is said to be vertically divided into $W_1$ and $W_2$.

The following lemmas can be shown immediately.

**Lemma 4.15** Vertical composition maps two sorted workflows to a single sorted workflow, that is, for any sorted workflows $W_1$ and $W_2$, $W_1 \times W_2$ is also a sorted workflow.

**Remark 4.16** The converse of the lemma above immediately holds by the definition of vertical division.

**Lemma 4.17** Vertical division preserve normality, that is, for any sorted workflows $W_1$ and $W_2$, if $W_1 \times W_2$ is normal, then $W_1$ and $W_2$ are normal.

**Remark 4.18** It does not always hold the converse of Lemma 4.17. In fact, we can easily make a counterexample from the right side of Fig. 5. To make vertical composition preserve normality, one needs additional conditions in the following lemma.

**Lemma 4.19** If $W_1$ and $W_2$ are normal sorted workflows and if $W_1 \times W_2$ satisfies the condition 5 in Definition 4.8, then, $W_1 \times W_2$ is normal.

4.4 Extended Total Correctness and Its Fundamental Properties

Hereinafter, $NLW$ denotes the set of all normal sorted workflows.

**Definition 4.20** Let $W \in NLW$, $S_M \subset S_M(W)$ with $S_M \neq \emptyset$, and $S_P \subset S_P(W)$ with $S_P \neq \emptyset$. Then, $S_M$ (or $S_P$) is called a main-in-port (or a passback-in-port, respectively) of $W$ if for each extended main (or passback) instance $I$ from $S_M$ (or from $S_P$) is consistent, that is, deadlock free and lack of synchronization free.

**Definition 4.21** Let $W \in NLW$, $I_M$ a subset of the power set of $S_M(W)$ and $I_P$ a subset of the power set of $S_P(W)$.

Then, $W$ is said to satisfy extended total correctness for $I_M$ and $I_P$ if the following properties hold.

1. $I_M$ is a set of some main-in-ports of $W$ and $I_P$ is a set of some passback-in-ports of $W$.
2. $S_M(W)$ is covered with $I_M$, that is, every $s \in S_M(W)$ is contained in some element of $I_M$. Moreover, $S_P(W)$ is covered with $I_P$.

We call $I_M$ and $I_P$ a covering main-in-port family and a covering passback-in-port family of $W$, respectively.

**Definition 4.22** A normal sorted workflow $W$ is said to satisfy extended total correctness if $W$ satisfies extended total correctness for some covering main- and passback-in-port families.

For a sorted workflow $W$, an $I \subset S_M(W)$ and a $J \subset S_P(W)$, let $EINS_M(W, I)$ be the set of all extended main instances of $W$ from $I$ and let $EINS_P(W, J)$ be the set of all extended passback instances of $W$ from $J$.

**Definition 4.23** For a sorted workflow $W$, a main-in-port family $I$ of $W$ and a passback-in-port family $J$ of $W$, the sets $\{E(V) | V \in EINS_M(W, I) \land I \in I\}$ and $\{E(V) | W \in EINS_P(W, J) \land J \in J\}$ are called the main-out-port family for $I$ of $W$ and the passback-out-port family for $J$ of $W$ and denoted by $\bigcup_M(W, I)$ and $\bigcup_P(W, J)$, respectively. Here, $E(V)$ denotes the set of all ending node of $V$.

Here we introduce three properties, which are useful to realize incremental verification of total correctness of normal workflows. Because of space limitations, we omit the
proofs of the properties. One can show them from Theorem 3.21 in the similar ways to those of the theorems in Sect. 4 in [12].

**Lemma 4.24** For a $W \in \text{NLW}$ with the unique start and the unique terminal, $W$ is totally correct if and only if $W$ satisfies extended total correctness.

We next show that extended total correctness is preserved by vertical composition and division of normal sorted workflows. Because of space limitations, we only show the result below based on Theorem 4.4 in [12].

**Theorem 4.25** Let $W_1$ and $W_2$ be sorted workflows with $E_M(W_1) = S_M(W_2)$ and $W_1 \ast W_2 \in \text{NLW}$ and let $\overline{I}_M$ be a covering main-in-port family of $S_M(W_1)$. Moreover, assume that $E_P(W_2) = S_P(W_1)$ and let $\overline{I}_P$ be a covering passback-in-port family of $S_P(W_2)$. Then, the vertical composition $W_1 \ast W_2$ satisfies extended total correctness for $\overline{I}_M$ and $\overline{I}_P$ if and only if the following properties hold.

1. $W_1$ satisfies extended total correctness for $\overline{I}_M$ and $\overline{O}_P(W_2, \overline{I}_P)$.
2. $W_2$ satisfies extended total correctness for $\overline{O}_M(W_1, \overline{I}_M)$ and $\overline{I}_P$.

Finally, we define “extensible property” of normal sorted workflows, which is often useful to start making a sub-workflow or to clip a sub-workflow from a large scale workflow.

**Definition 4.26** For a $W \in \text{NLW}$, $W$ is said to be extensible if there exist $W_1, W_2 \in \text{NLW}$ such that $(W_1 \ast W) \ast W_2$ is a totally correct workflow.

**Theorem 4.27** For a $W \in \text{NLW}$, $W$ is extensible if and only if $W$ satisfies extended total correctness.

The theorem above assures that one can complete a totally correct workflow while keeping extensibility of $W$ and expanding $W$ vertically. It also assures that, if $W$ does not satisfy extended total correctness, one has no choice but to modify $W$ in order to complete a totally correct workflow from $W$ by extending $W$ vertically. It is useful to check extended total correctness of an incomplete workflow (= a sorted workflow) to develop a consistent large scale workflow, since one may have an opportunity to modify structure of the incomplete workflow before it grows too large to modify the structure easily.

5. **Case Study**

To demonstrate usefulness of extended total correctness and the fundamental properties in the previous subsection, we provide a desktop evaluation of them. To this end, we use a large scale workflow that was developed in real information system development.

5.1 **Large Scale Workflow as an Object of Case Study**

The large scale workflow that we here use as an object of the desktop evaluation was developed in development of an accounting and finance system and a personnel system for an independent administrative institution.

The large workflow describes a process of payroll accounting of the institution above, and it consists of 21 small workflows as pictured in Fig. 9. Each small workflow denotes a business process of a work that office workers collaborate in performing, and it has a size between 10 and 40 nodes. Four engineers collaborated in developing the workflow.

The outline of whole process of payroll accounting the large workflow describes is as follows. In the first step, office workers register bank accounts of employees to pay their wages or expenses accounts ($W_1$). Next, approvals of fringe benefits of employees such as sustenance allowance ($W_2, 1, \ldots, W_2, 7$) and confirmations of approvals of particular fringe benefits ($W_3, 1$) are performed with calculations of three kinds of deduction amounts ($W_4, 1$, $W_6, 2$ and $W_6, 3$). Next, calculations of payrolls ($W_7$) are performed. In a calculation of payroll, if an error of a performance result occurs, then a passback of a fringe benefits approval, an approval confirmation or an attendance record confirmation is performed, which is described by passback instance starting from $W_7$. If no error occurs, creations of request papers such as payment requests ($W_8, 1$, $W_8, 2$ and $W_8, 3$) are performed. Next, creations of firm banking records ($W_9$) are performed, and finally, postings of payment statements ($W_{10}$) are performed.

5.2 **Verification of Extended Total Correctness of a Workflow and Improvement of a Workflow**

To demonstrate usefulness of the fundamental properties of extended total correctness and incremental verification based on the properties, we verified extended total correct-
ness of small workflows that constitute the large scale workflow of payroll accounting in the previous subsection and improve them based on the verification.

During verifications of the correctness, we noticed that the designers did not clearly and accurately model the connection between the workflows that describe works of the first half of the whole payroll accounting and the workflows describing works in the last half. Thus, we re-created a workflow \( W_4 \) with reference to documents and remarks in original workflows.

Each rectangle denotes a small workflow, which is actually a small sorted workflow. The dotted lines in Fig. 9 connecting rectangles denote bijections of vertical compositions of the small workflows. The dotted lines denoted in each rectangle describe a rough sketch of the process of each work. We omit the detailed description of a process in each small workflow. We do not describe any passback flow that occurs only in a single sorted workflow. However, we note that we do not intentionally add or replace any sorted workflow except \( W_4 \) and that we faithfully describe the connection between small sorted workflows. After verification of extended total correctness of each sorted workflow, we did not detect any defect that can be judged to be a defect by checking only a single sorted workflow. In fact, the workflows have been developed by professional engineers and they had checked the contents of the workflows before the verification of this subsection.

On the other hand, we detected a defect in the control flow of the whole workflow. We here explain the defect. When one obtained \( W_7, W_8-1, W_8-2, W_8-3 = W_9 \), one can detect a defect in the control flow by verifying extended total correctness. The defect occurs because, despite the flow forks in an XOR-split in \( W_7 \), the flows are bundled by an AND-join. Actually one can check that a sorted workflow \( \mathcal{W} \) consisting of \( W_7, W_8-1, W_8-2, W_8-3, W_9 \) does not satisfy extended total correctness on the defect point.

Thus, we found out the details of the contents of \( W_7 \) and \( W_9 \), and we found that, after a calculation of a payroll described by \( W_7 \), the office worker performs one of the three works, as follows.

1. Creation of a payment request paper.
2. Creation of a payment request paper and an addition payment request paper.
3. Creation of a payment request paper and a payment return request paper.

Therefore, we improve \( W_7 \) and \( W_9 \) to make them accurately describe the process above, and we obtained an improved version \( \mathcal{W}' \) in Fig. 10. This version accurately describes the process of the last half of the whole payroll accounting. Actually, \( \mathcal{W}' \) satisfies extended total correctness.

5.3 Discussion

Through the desktop evaluation, we confirmed two merits of the fundamental properties of extended total correctness and incremental verification based on them.

The first advantage is that the proposed verification actually helps to check the control flow of a part (= a sorted workflow) of the whole workflow in an early stage. The whole workflow of payroll accounting was developed by four engineers, and they kept their own small workflows. Therefore, it is not possible to verify correctness of the whole workflow unless all engineers complete developing their small workflows and they share all workflows. However, since the small workflows that constitute \( \mathcal{W} \) were developed by two engineers, it was possible to verify extended total correctness of \( \mathcal{W} \) and to improve it if only the engineers share the small workflows constituting \( W \). Since \( W_7 \) contains passbacks, it was not possible to accomplish incremental verification based on existing theory including [12]. The fundamental properties in this paper enables one to accomplish incremental verification even if small workflows have passbacks infecting multiple workflows.

The second advantage is that, if a sorted workflow (a part of the whole workflow) contains a defect in extended total correctness, then the defect will never be solved unless the sorted workflow itself is modified by Theorem 4.27. Therefore, one does not have to worry about the case that the “defect” actually gives no bad influence for the whole workflow. The proposed fundamental properties insure that such a defect should be always modified.

6. Related Works

There are a lot of researches of correctness of workflows such as [1], [4], [6], [9], [10]. However, most of such researches are based on the premise that a workflow is verified at once after the whole workflow is completely developed. Open workflow nets [7] and local- and global- soundness for sub-workflows [8] provide useful methods for incremental verification. While the methods use external information such as “contract” or “scenario”, our method does not need such information by dealing with workflows with passbacks instead of usual cyclic workflows.

On the other hands, the questions of how to represent passbacks (redoing) of work in a workflow and of what semantics to apply to passbacks comprise an important issue in workflow research. In order to address this problem, Aalst [3] defined reset workflow net by extending the work-
flow net that he had already devised, and carried out research into the semantics and soundness of workflows with cancel-
lations based on reset workflow net. It is known that the
soundness of reset workflow net is undecidable. In this pa-
per, by limiting cyclic workflows to those that are normal,
we make total correctness decidable and useful for incre-
mental verification (cf. Theorem 3.21).

Our incremental verification of correctness of work-
flows has been introduced in our previous paper [12]. How-
ever, the theory in [12] can deal with only acyclic work,
flows, and hence, to enhance usefulness of the theory, it is
necessary to extend the theory to cyclic workflows. How-
ever, it is not easy to realize incremental verification for all
cyclic workflows. Thus, we consider workflows with pass-
backs instead of all cyclic workflows, and establish new cor-
rectness properties and normality of (sorted) workflows, and
realize incremental verification for workflows with pass-
backs.

This paper is established from our previous pa-
pers [13], [14]. That is, Sect. 3 is based on [13], while Sect. 4
is based on [14].

7. Conclusions

We introduce workflows with passbacks and correctness of
them by defining normality of them and total correctness of
normal workflows based on transition system-based seman-
tics of them. By using normality and total correctness of
workflow with pbfs instead of existing correctness of cyclic
workflow, we can reduce correctness of a workflow \( W \) to ex-
isting correctness of acyclic workflows that compose \( W \)
and a question of how the main instance and passback instances
of the workflow are linked (Theorem 3.21). This theorem ena-
bles us to extend the fundamental theory of incremental
verification [12] to cyclic workflows. More specifically, we
first define sorted workflows, which are extended versions of
workflows with pbfs, and vertical composition and division
of them. We also extend normality and total correctness to
sorted workflows, and we introduce fundamental properties
including Theorem 4.25 that claims that total correctness is
preserved via vertical composition and division. Moreover,
we provide a desktop evaluation of the fundamental proper-
ties above by using a large scale workflow that was de-
veloped in real information system development.

One of the future works is to extend the fundamental theory for more general composition and division of work-
flows. As far as observing real workflows used in informa-
tion system development, vertical composition and division
have sufficient usefulness. However, it is interesting to ex-
tend composition and division of workflows, since it may
enable one to realize more flexible incremental verification
for large scale workflows.

Acknowledgments

This work was supported by Grant-in-Aid for Scientific Re-
search (C) 20500045.

References

[1] W.M.P. van der Aalst, “Verification of workflow nets,” Application
and Theory of Petri Nets 1997, LNCS 1248, pp.407–426, 1997.
[2] W.M.P. van der Aalst, “The application of petri nets to workflow
management,” J. Circuits, Systems and Computers, vol.8, no.1,
pp.21–66, 1998.
[3] W.M.P. van der Aalst, “Discovery, verification and conformance of
workflows with cancellation,” Proc. 4th international conference on
Graph Transformations table of contents, ser. LNCS 5214, pp.18–
37, 2008.
[4] W.M.P. van der Aalst and A.H.M. ter Hofstede, “YAWL: Yet another
workflow language,” Inf. Syst., vol.30, no.4, pp.245–275, 2005.
[5] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski,
and A.P. Barros, “Workflow patterns,” Distributed and Parallel
Databases, vol.14, no.1, pp.5–51, 2003.
[6] W.M.P. van der Aalst, A. Hinschall, and H.M.W. Verbeeck, “An
alternative way to analyze workflow graphs,” Proc. 14th Interna-
tional Conference on Advanced Information Systems Engineering
(CAiSE), ser. LNCS 2348, pp.535–552, 2002.
[7] W.M.P. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, and K.
Wolf, “From public views to private views: Correctness-by-design for
services,” Informal Proceedings the 4th International Work-
shop on Web Services and Formal Methods (WS-FM), LNCS 4937,
pp.139–153, 2007.
[8] E. Kindler, A. Martens, and W. Reisig, “Inter-operability of work-
ow applications: Local criteria for global soundness,” Business Process
Management: Models, Techniques, and Empirical Studies (BPM),
LNCS 1806, pp.235–253, 2000.
[9] H. Lin, Z. Zhao, H. Li, and Z. Chen, “A novel graph reduction al-
gorithm to identify structural conlicts,” Proc. 35th Annual Hawaii
International Conference on System Science (HICSS), 2002.
[10] W. Sadiq and M.E. Orlowska, “Analyzing process models using
graph reduction techniques,” Inf. Syst., vol.25, no.2, pp.117–134,
2000.
[11] O. Takaki, T. Seino, I. Takeuti, N. Izumi, and K. Takahashi, “Veri-
fication of evidence life cycles in workflow diagrams with passback
flows,” Int. J. Advances in Software, vol.1, no.1, pp.14–25, 2008.
[12] O. Takaki, I. Takeuti, T. Seino, N. Izumi, and K. Takahashi, “In-
cremental verification of consistency properties of large scale work-
flows from the perspectives of control flow and evidence life cycles,”
Int. J. Advances in Software, vol.1, no.2, pp.147–161, 2009.
[13] O. Takaki, I. Takeuti, N. Izumi, and K. Hasida, “Syntax and se-
manics of workflows that include passbacks,” Proc. 5th Inter-
national Conference on Software Engineering Advances (ICSEA
2010), pp.169–177, 2010.
[14] O. Takaki, I. Takeuti, N. Izumi, and K. Hasida, “Incremental verifi-
cation of consistency property of large-scale workflows that contain
passback flows,” Proc. 9th Joint Conference on Knowledge-Based
Software Engineering (JKKBSE 2010), pp.163–177, 2010.

Appendix A: Proof of Theorem 3.20

In this section, we fix an acyclic workflow \( W \) and prove that
\( W \) is totally correct if and only if \( W \) is correct in the sense of
Definition 3.19.

Lemma A.1 Every main instance contains the terminal of
\( W \). Moreover, every main instance contains no ending node
except the terminal of \( W \).

Proof. Let \( I \) be a main instance and \( n \) a node of \( I \) that is not
the terminal of \( W \). Then, \( I \) contains at least one outgoing
arc of \( n \), and hence, \( n \) is not an ending node of \( I \). Moreover,
since \( W \) is acyclic, \( I \) eventually has the terminal of \( W \).
Definition A.2 For a node $n$ of $W$, the depth of $n$ denotes the maximal length of the directed paths from the trigger of $W$ to $n$.

Lemma A.3 Let $I$ be a main instance of $W$ and $H$ a pre-instance with $H \subset I$. Then, $H \rightarrow_{TW} I$ if $W$ is correct.

Proof. We show this lemma by induction on $\#(Arcs(I)-Arcs(H))$. It is trivial in the case where $H = I$. Assume that $H \subset I$. Let $f$ be an arc in $Arcs(I)-Arcs(H)$ satisfying that $src(f)$ has the least depth among $\{src(g) | g \in Arcs(I)-Arcs(H)\}$. Moreover, let $H'$ be the pre-instance that is obtained from $H$ by adding $f$. Then, $H \subseteq H' \subseteq I$ and $H' \rightarrow_{TW} I$ by the induction hypothesis. Thus, it suffices to show the following property:

$$H \rightarrow_{TW} H'. \quad (A.1)$$

(i) If $n := src(f)$ is an XOR-split, then $I$ contains no outgoing-arc of $n$ except $f$ by the property (3) of Definition 3.5, and hence, $H$ does not contain any outgoing-arc of $n$. Thus, $H \rightarrow_{TW} H'$.

(ii) If $n$ is an AND-join, then $I$ contains all incoming-arcs of $n$ in $W$, since $I$ is deadlock free.

On the other hand, $H$ contains all incoming-arcs of $n$ in $I$, since $src(f)$ has the least depth among $\{src(g) | g \in Arcs(I)-Arcs(H)\}$. Thus, $H \rightarrow_{TW} H'$.

(iii) Otherwise, it is clear that $H \rightarrow_{TW} H'$.

Definition A.4 For subgraphs $X$ and $Y$ of $W$, $X \rightsquigarrow Y$ if $X$ and $Y$ satisfy one of the following properties.

1. $Y$ is obtained from $X$ by adding an outgoing-arc of an ending node of $X$.
2. $Y$ is obtained from $X$ by adding an outgoing-arc $f$ of a node $n$ of $X$, where $n$ is an AND-split and $f$ is not in $X$.

Lemma A.5 For pre-instances $H$ and $H'$ of $W$, if $H \rightarrow_{TW} H'$, then $H \rightsquigarrow H'$.

Proof. Since $W$ is acyclic, there exists an arc $f$ in $W$ that satisfies the condition (A) of Definition 3.15. So, $f$ also satisfies one of the conditions (1) and (2) of Definition A.4.

Lemma A.6 Let $I$ be a main instance of $W$, $H$ a pre-instance with $H \subset I$, and $G$ a subgraph of $W$ that is a closure of $H$ with respect to $\rightsquigarrow$. Then, $G$ is a main instance of $W$.

Proof. It is clear that $G$ satisfies the properties (1), (3) and (4) of Definition 3.5. So, we show (2) of Definition 3.5. For subgraphs $X$ and $Y$ of $W$, if $X \rightsquigarrow Y$, then $X \subseteq Y$. Thus, there exist subgraphs $X_1, …, X_n$ of $W$ such that $H = X_1 \rightsquigarrow X_2 \rightsquigarrow \ldots \rightsquigarrow X_n = G$. Then, it is easily shown that an $X_i$ satisfies (2) of Definition 3.5 by the induction on the index $I$.

Lemma A.7 For any $H \in AC(W)$, there exists a main instance $I$ such that $H \subset I$.

Proof. We show this lemma by induction on $\#Arcs(H)$. If $H$ is the smallest pre-instance consisting of the trigger only, then every instance contains $H$. Assume that there exists $H_0$ with $H_0 \rightarrow_{TW} H$. Then, by Lemma A.6, $H_0 \rightsquigarrow H$. On the other hand, by the induction hypothesis, there exists a main instance $I_0$ with $H_0 \subset I_0$. So, by Lemma A.5, there exists a main instance $I$ that is the closure of $H_0$ with respect to $\rightsquigarrow$. Since $H_0 \rightsquigarrow H$, $H \subset I$. So, we have the result.

Lemma A.8 For any $H \in AC(W)$, $H$ is deadlock free.

Proof. One can easily show this lemma by induction on $\#Arcs(H)$.

Lemma A.9 For any $H$ and $H' \in AC(W)$ with $H \rightarrow_{TW} H'$, if $H$ is not lack of synchronization free, neither is $H'$.

Proof. It is trivial since $H \subset H'$.

Lemma A.10 If $W$ is totally correct, then every main instance $I$ of $W$ satisfies that $I \in AC(W)$.

Proof. Let $I$ be a main instance of $W$ and $H$ a pre-instance with $H \in AC(W)$ and $H \subset I$. Then, we show that $H \rightarrow_{TW} I$ in a similar way to the proof of Lemma A.3, that is, by induction on $\#(Arcs(I)-Arcs(H))$. It is trivial in the case where $H = I$. Thus, we assume $H \subset I$ and construct $H'$ satisfying that $H \rightarrow_{TW} H'$ and that $H' \subset I$.

(1) Assume that $H$ contains an ending node except the terminal of $W$. Then, by (1) of Definition 3.18, there exists a $G \in AC(W)$ with $H \rightarrow_{TW} G$. Let $f \in Arcs(G)-Arcs(H)$ and $m := src(f)$. Then, $H$ contains $m$. If $m$ is not any XOR-split, then $I$ contains $f$, since $I$ contains all outgoing-arcs of $m$. Thus, we have $H'$ to be $G$. Let $m$ be an XOR-split. Then, $H$ contains no outgoing-arc of $m$ and $I$ contains one outgoing-arc $G$ of $m$. Moreover, the pre-instance $G'$ obtained from $H$ by adding $G$ to $H$ satisfies that $H \rightarrow_{TW} G'$. So, we have $H'$ to be $G'$.

(2) Assume that $H$ contains no ending node except the terminal of $W$. Then, there exists an arc $f \in Arcs(I)-Arcs(H)$ satisfying that $src(f) = AND-split$ and that $H$ contains $src(f)$. Thus, we have $H'$ that is obtained from $H$ by adding $f$ to $H$, since $H \rightarrow_{TW} H'$.

Proof of Theorem 3.20 Let $W$ be an acyclic workflow.

(1) Assume that $W$ is correct. Then, by Lemmas A.3 and A.7, every $H \in AC(W)$ is accessible to a main instance of $W$. Thus, by Lemma A.1, $W$ satisfies (1) of Definition 3.18. On the other hand, every $H \in AC(W)$ is deadlock free by Lemma A.8. Moreover, by Lemma A.9, there exists no $H \in AC(W)$ that is not lack of synchronization free, since every $H \in AC(W)$ is accessible to a main instance of $W$. Thus, $W$ satisfies (2) of Definition 3.18, and hence, $W$ is totally correct.

(2) Assume that $W$ is totally correct. Then, by Lemma A.10, every instance $I$ satisfies that $I \in AC(W)$, and hence, $I$ is deadlock free and lack of synchronization free. Thus, $W$ is correct.

Appendix B: Sketch of a Proof of Theorem 3.21

Here we provide a sketch of a proof of Theorem 3.21. Let $W$ be a normal workflow.

(1) We first show the “only if” part of Theorem 3.21. That is, we assume that $W$ is totally correct and show that all main instances and all passback instances of $W$ are deadlock free and lack of synchronization free, as follows.

(1.1) Let $I_M$ be an arbitrary main instance of $W$. Then, by Lemma A.10, $I_M \in AC(M_W) \subset AC(W)$. Thus, $I_M$ is deadlock free and lack of synchronization free, since $W$ is totally

---

1For a set $X$, $\#X$ denotes the number of elements of $X$.  

---

TAKAKI et al.: WORKFLOWS WITH PASSBACKS AND INCREMENTAL VERIFICATION OF THEIR CORRECTNESS 999
correct.

(1.2) Let $I_P$ be an arbitrary passback instance of $W$. Thus, in a similar way to the proof of Lemma A.10, one can show that there exists a pre-instance $H \in \mathcal{AC}(W)$, that consists of some pre-instance of $M_W$ and $I_P$ minus an ending-passback arc $f$. Since $H$ is deadlock free and since $f$ does not share the target with any other arc in $I_P$ by the XOR-join-condition in Sect. 3.3, $I_P$ is lack of synchronization free. On the other hand, $I_P$ is deadlock free except $src(f)$, since $H$ is deadlock free. Moreover, $H$ is accessible to some pre-instance via $f$ by Definition 3.15, since $W$ satisfies the condition (1) of Definition 3.18. Thus, $I_P$ is deadlock free at $src(f)$. So, $I_P$ is deadlock free and lack of synchronization free.

(2) We next show the “if” part of Theorem 3.21. That is, we assume that all main instances and all passback instances of $W$ are deadlock free and lack of synchronization free and show that $W$ is totally correct, as follows.

(2.1) Let $H$ be a pre-instance of $W$. Then, $H$ can be divided into a pre-instance $H_0$ of $M_W$ and pre-instances $H_1, \ldots, H_n$ of $P_{W_1}, \ldots, P_{W_n}$, respectively. Here, each $P_{W_i}$ is a passback instance that satisfies that $P_{W_i} = P_{W_i}^{1} \cup \ldots \cup P_{W_i}^{n}$. By induction on $n$, one can show that $H$ can be accessible to some pre-instance $G$ of $M_W$. Moreover, one can show that $G$ can be accessible to a main instance of $M_W$, that satisfies the condition of $H_r$ in (1) of Definition 3.18. These proofs are accomplished in a similar way to the proofs of Lemmas A.3 and A.7.

(2.2) Let $H \in \mathcal{AC}(W)$. Then, as in (2.1), $H$ can be divided into $H_0, H_1, \ldots, H_n$, where $H_0$ is a pre-instance $H_0$ of $M_W$ and $H_1, \ldots, H_n$ are pre-instances of $P_{W_1}^{1}, \ldots, P_{W_n}^{n}$. Suppose that $H$ is not deadlock free or lack of synchronization free. Then, there exists an $H_r$ that is not deadlock free or lack of synchronization free. Then, in a similar way to the proofs of Lemmas A.7, A.8 and A.9, one can have a main instance or a passback instance, which is not deadlock free or lack of synchronization free. It contradicts the assumption that all main instances and all passback instances of $W$ are deadlock free and lack of synchronization free. Therefore, $H$ is deadlock free and lack of synchronization free, and hence, $W$ satisfies (2) of Definition 3.18. □

Appendix C: Proofs of Lemmas and Theorems in Sect. 4

In this appendix, we prove lemmas and theorems in Sect. 4. To this end, we also show preliminary lemmas.

Proof of Theorem 4.10 We first show “if” part of this lemma. Assume that $W$ is normal as a workflow. Then, $W$ clearly satisfies all conditions in Definition 4.8 other than the condition (5). Thus, we here show (5). Since $W$ is a workflow, all triggers and terminals of $P_{W_1}^{1}$ are contained in $M_W^{1}$. Thus, $Cl(W) = W$, and hence, (5.1) and (5.2) immediately hold. Let $p$ be a trigger of $P_{W_1}^{1}$ and $U$ be a maximal extended main instance containing $p$. Since $W$ is a workflow, $U$ is a main instance of $W$. Thus, for each passback instance $I_P$ starting from $p$, $U$ contains all ending nodes of $I_P$ by the condition (6) in Definition 3.11.

We next show “only if” part of this lemma. Assume that $W$ is normal as a sorted workflow and we only show the condition (6) in Definition 3.11. Let $I_M$ be a main instance and $I_P$ a passback instance starting from a node $p$ in $I_M$. Since $W$ is a workflow, every main instance is a maximal extended main instance. Thus, $I_M$ contains all ending nodes of $I_P$ by the condition (5.3) in Definition 4.8.

Lemma C.1 Let $W$ be a sorted workflow, $\{s_1, \ldots, s_n\}$ a set of triggers of $M_W^{1}$ and $\{t_1, \ldots, t_m\}$ a set of triggers of $P_{W_1}^{1}$. Then, there exist extended main and passback instances starting from $\{s_1, \ldots, s_n\}$ and $\{t_1, \ldots, t_m\}$, respectively.

Proof. Let $\{c_1, \ldots, c_k\}$ be the set of all XOR-split in $M_W^{1}$ and let $\{f_j_1, \ldots, f_j_k\}$ be a set of arcs that are outgoing-arcs of $c_1, \ldots, c_k$, respectively. Then, for each trigger $s_i$ of $M_W^{1}$, there exists a main instance $I_i$ stating from $s_i$ such that, if $I_i$ has an XOR-split $c_j$, then $I_i$ has $f_j$. Since $\{I_1, \ldots, I_n\}$ is not conflict, $I_1 \cup \ldots \cup I_n$ is an extended main instance that starts from $\{s_1, \ldots, s_n\}$. One can obtain an extended passback instance in a similar manner. □

Because of space limitations, we only provide a sketch of a proof of Lemma 4.17, as follows.

Sketch of a Proof of Lemma 4.17 Assume that $W_1 \ast W_2$ is normal. We only show that $W_1$ satisfies the condition 5 in Definition 4.8, since one can show that $W_1$ satisfies other conditions in a similar manner to the following proof or easier than the condition 5.

By normality of $W_1 \ast W_2$, there exists a closure $Cl(W_1 \ast W_2)$ satisfying the conditions 5.1, 5.2 and 5.3 in Definition 4.8. Then, we first construct a closure of $W_1$. Let $p$ be a trigger of $P_{W_1}^{1}$ with $p \notin M_W^{1}$. Then, there exists a trigger $q$ of $P_{Cl(W_1 \ast W_2)}^{1}$ with $q \rightarrow_{w} p$. Since $Cl(W_1 \ast W_2)$ satisfies the condition 5.1, there exists a terminal $t$ of $M_W^{1}$ with $t \rightarrow_{w} q$. We denote such a terminal $t$ by $p$. Thus, by connecting each trigger $p$ above with $p$, and by connecting each terminal with the same trigger of $Cl(W_1 \ast W_2)$, one can obtain a closure $Cl(W_1)$ of $W_1$.

We next show that the closure $Cl(W_1)$ satisfies the conditions in Definition 4.8.5. Here, we only show that $Cl(W_1)$ satisfies the condition 5.3. Let $U_1$ be a maximal extended main instance of $Cl(W_1)$ containing $t$. Note that $U_1$ is also a maximal extended main instance of $W_1$. Moreover, by Lemma C.1, there exists a maximal extended main instance $U_2$ of $W_2$ from all ending nodes of $U_1$ contained in $W_2$. Let $U_1 \ast U_2$ be the subgraph of $W_1 \ast W_2$ obtained from $U_1$ and $U_2$ in the same way as Definition 4.10. Then, $U_1 \ast U_2$ is clearly an extended main instance of $W_1 \ast W_2$. If $U_1 \ast U_2$ is not maximal, then there exists a main instance $V$ of $W_1 \ast W_2$ and $(U_1 \ast U_2) \subseteq (U_1 \ast U_2) \cup V$. Thus, $U_1 \subseteq (U_1 \ast U_2) \cup W_2$ and $(U_1 \ast U_2) \subseteq (U_1 \ast U_2) \cup V$ where $(U_1 \ast U_2) \cup W_2$ is the extended main instance of $W_1$ obtained from $(U_1 \ast U_2) \cup V$ by restricting to $W_1$. It contradicts that $U_1$ and $U_2$ are maximal extended main instance. Therefore, $U_1 \ast U_2$ is a maximal extended main instance of $W_1 \ast W_2$. On the other hand, let $I_P$ be a passback instance of $Cl(W_1)$ starting from $t$. Then, there exists a passback instance $I_Q$ of $Cl(W_1 \ast W_2)$ starting from $q$
and including $I_p$ other than $t$. Thus, by normality of $W_a^*W_2$, $U_1^*U_2$ contains all ending nodes of $I_Q$. Thus, $U_1$ contains all ending nodes of $I_p$. □

**Proof of Lemma 4.24** Let $W$ be a workflow. Then, by Lemma 4.10, $W$ is also normal as a workflow. Moreover, by Theorems 3.20 and 3.21, $W$ is totally correct iff all main instances and passback instances of $W$ are deadlock free and lack of synchronization free. On the other hand, since $W$ has the unique trigger and the unique terminal, $W$ has extended total correctness iff the singleton of the trigger is a main-in-port and all singletons of starting nodes of passback instances are passback-in-ports. Since, each extended instance of $W$ starting from just one trigger is an instance of $W$, $W$ has extended total correctness iff all main instances and passback instances of $W$ are deadlock free and lack of synchronization free. Thus, we have the result. □

**Proof of Lemma 4.25** We here provide a proof of this lemma in a similar manner to the proof of Theorem 4.2 in [12].

(1) We first show “if” part of this lemma. Assume that $W_1$ satisfies extended total correctness for $\overline{I}_M$ and $\overline{O}_M(W_2, I_p)$ and that $W_2$ satisfies extended total correctness for $\overline{O}_M(W_1, I_M)$ and $\overline{I}_p$. Let $I_M \in \overline{I}_M$ and $U$ an extended main instance of $W_1^*W_2$ starting from $I_M$. Then, $U$ can be divided into two main instances $U_1$ and $U_2$ of $W_1$ and $W_2$ such that $U_1 \in \text{EINS}_M(W_2, I_M)$ and $U_2 \in \text{EINS}_M(W_2, E(U_1))$. By extended total correctness of $W_1$, $U_1$ is consistent. Moreover, since $E(U_1) \in O_M(W_1, I_M)$, $U_2$ is consistent by extended total correctness of $W_2$. Thus, since $U = U_1^*U_2$, $U$ is consistent. In a similar manner, for each $I_p \in \overline{I}_p$, each extended passback instance of $W_1^*W_2$ starting from $I_p$ can be shown to be consistent. Thus, $W_1^*W_2$ satisfies extended total correctness for $\overline{I}_M$ and $\overline{I}_p$.

(2) We next show “only if” part. Assume that $W_1^*W_2$ satisfies extended total correctness for $\overline{I}_M$ and $\overline{I}_p$ and let $I_M \in \overline{I}_M$ and $U_1$ an extended main instance of $W_1$ starting from $I_M$. By Lemma C.5, there exists an extended main instance $U_2$ of $W_2$ starting from $E(U)$. Then, $U_1^*U_2$ is an extended main instance of $W_1^*W_2$, and hence, $U_1^*U_2$ is consistent by extended total correctness of $W_1^*W_2$, and hence, $U_1$ is also consistent. We here show that, for each $I_p \in \overline{I}_p$, each extended passback instance $V_1$ of $W_1$ starting from $I_p$ is consistent. Let $I_p \in \overline{I}_p$. Then, since there exists $J \in \overline{I}_p$ and $V_2 \in \text{EINS}_p(W_2, J)$ with $E(V) = I_p$. Thus, $V_1^*V_2$ is consistent by extended total correctness of $W_1^*W_2$, and hence, $V_1$ is also consistent. Thus, $W_1$ satisfies extended total correctness for $\overline{I}_M$ and $\overline{O}_p(W_2, I_p)$. In a similar manner, one can show that $W_2$ satisfies extended total correctness for $O_M(W_1, I_M)$ and $\overline{I}_p$. □

Because of space limitations, we only provide a sketch of a proof of Theorem 4.27, as follows.

**Sketch of a Proof of Theorem 4.27**

(1) We first show “only if” part of this theorem. Assume that $W$ is extensible. Then, there exist $W_1, W_2 \in \text{NLW}$ such that $(W_1^*W)^*W_2$ is a total correct workflow. Then, by Lemma 4.24 an Theorem 4.25, $W$ satisfies extended total correctness.

(2) We next show “if” part of this theorem. Assume that $W$ is satisfies extended total correctness. Then, we provide $W_1, W_2 \in \text{NLW}$ such that $(W_1^*W)^*W_2$ is a totally correct workflow in the following three steps.

(Step-1) Since $W \in \text{NLW}$, there exists a closure $C(W)$ of $W$ satisfying the condition 5 in Definition 4.8. Moreover, for some sorted workflow $W_a$ and $W_b$, $C(W)$ can be vertically divided with $(W_a^*W)^*W_b$. Clearly, $W_a$ and $W_b$ satisfy extended total correctness.

(Step-2) We construct an acyclic workflow $W_0$ with no passback arc in the same way to the proof of Lemma A.16 in [12]. Let $\{s_1, \ldots, s_n\}$ be the set of all trigger of $M^2W$ and $\{I_1, \ldots, I_m\}$ be the covering main-in-port family. Without loss of generality, one can consider $I_1, \ldots, I_k$ to be singletons and $I_{k+1}, \ldots, I_m$ be sets having multiple elements, respectively. Then, we consider $\{s_1, \ldots, s_n\}$ to be a set of triggers of $W_0$. Moreover, for $\{I_1, \ldots, I_k\}$ we consider a set of arcs $\{f_1, \ldots, f_k\}$, and for $\{I_{k+1}, \ldots, I_m\}$ we consider a set of AND-split nodes and their incoming arcs $\{f_{k+1}, \ldots, f_m\}$. Finally, we connect the nodes and arcs in Fig. A-1 below. Here, each outgoing arc of each AND-split corresponds to an element of each $I_i$ ($i = k + 1, \ldots, m$). Clearly, $W_0$ also satisfies extended total correctness with main-out-port family $\{I_1, \ldots, I_m\}$.

(Step-3) Let $W_1 = W_0^*W_a$ and $W_2 = W_b$. Then, $W_1$ and $W_2$ are normal sorted workflows and $(W_1^*W)^*W_2$ satisfies extended total correctness. □

Osamu Takaki received his doctoral degree in mathematics from Kyoto Sangyo University in 1998. He is currently working in Japan Advanced Institute of Science and Technology. His research interests include knowledge modeling and quantification of medical services.
Izumi Takeuti  received his doctoral degree of informatics from Kyoto University in 2000. He is affiliated with AIST (National Institute of Advanced Industrial Science and Technology) from 2005. His subject of research is logic.

Noriaki Izumi  is a researcher at the National Institute of Advanced Industrial Science and Technology. He received B.E. and M.E. degrees from Osaka Prefecture University in 1992 and 1994, respectively. His research interests include Intelligent Systems, Knowledge Modeling, Semantic Web, and Web Services. He is a member of JSSST, IPSJ, and JSAI.

Koiti Hasida  received his doctoral degree of science from the University of Tokyo in 1986. He was affiliated with ETL (Electrotechnical Laboratory) from 1986 to 2001, and also with the Institute of New Generation Computer Technology (ICOT) from 1988 to 1992. He is currently at AIST (National Institute of Advanced Industrial Science and Technology). His research encompasses linguistics, cognitive science, natural language processing, intelligent content, service science, and so forth.