On Verification and Application of Behavioral Inheritance for Parallel Synchronized Interworkflows

Shingo Yamaguchi†, Tetsushi Narui†, Qi-Wei Ge†, and Minoru Tanaka†
†Graduate School of Science and Engineering, Yamaguchi University,
16-1 Tokiwadai 2-chome, Ube-shi, 755-8611, Japan
‡Faculty of Education, Yamaguchi University,
1677-1 Yoshida, Yamaguchi-shi, 753-8513, Japan
E-mail: {shingo, k028vk, gqw, tanakam}@yamaguchi-u.ac.jp

Abstract: An interworkflow \( N \) is constructed by connecting a workflow \( N_X \) with another workflow \( N_Y \), so interworkflow \( N \) should inherit the behavior from workflow \( N_X \) (and workflow \( N_Y \)). Behavioral inheritance guarantees that interworkflow \( N \) can be substituted for workflow \( N_X \). Nevertheless it may happen that the behavior is not inherited. Behavioral inheritance can be verified by comparing the reachability graphs of the WF-nets representing interworkflow \( N \) and workflow \( N_X \). However this verification method is limited to small interworkflows due to the complexity of the state-space explosion. Focusing on a pattern of interworkflows, called Parallel synchronized pattern, we propose a condition to verify behavioral inheritance of Parallel synchronized interworkflows. This condition enables us to verify the behavioral inheritance in polynomial time.

1. Introduction

Interoperability technology enables two or more organizations to participate in a workflow. A workflow crossing organizational boundaries is called an interorganizational workflow (interworkflow for short). The Workflow Management Coalition (WFMC for short), an international standardization organization on workflows, has given a protocol for interworkflows. In the protocol, there are three patterns of interoperability: Chained, Nested, and Parallel synchronized; and an interworkflow \( N \) is constructed by connecting a workflow \( N_X \) with another workflow \( N_Y \) by using those interoperability patterns. Since interworkflow \( N \) is an extension of \( N_X \), the behavior of interworkflow \( N \) should preserve the behavior of workflow \( N_X \).

In [1], we have given a workflow net (WF-net for short) based modeling method of interworkflows based on the WFMC protocol. A WF-net represents the structure of an interworkflow, and the reachability tree of the WF-net represents the behavior of the interworkflow. Van der Aalst et al. [2] have proposed a notion of inheritance of behavior between workflows. A WF-net \( N_A \) is said to be a subclass under projection inheritance of a WF-net \( N_B \) if the behaviors of \( N_A \) and \( N_B \) cannot be distinguished by hiding some of the transition newly added into \( N_A \). Behavioral inheritance can be verified by comparing the reachability graphs of the WF-nets. However this verification method is limited to small workflows due to the complexity of the state-space explosion. Focusing on a pattern of interworkflows, called Parallel synchronized pattern, we propose a condition to verify behavioral inheritance of Parallel synchronized interworkflows.

2. Preliminary

2.1 Labeled WF-Nets

Labeled WF-nets are labeled Petri nets representing workflows. The labeled WF-net representing a workflow possesses the transitions corresponding to a single action (identified by the label) in the workflow. The label of a transition may be used to abstract the transition from the action. Transitions may be labeled as a designated label \( \tau \). Label \( \tau \) is used to distinguish between external and internal behavior of the WF-net. The firing of transitions labeled as \( \tau \) cannot be observed. Only the firing of transitions with a label different from \( \tau \) are externally observable. Thus transitions labeled as \( \tau \) are called internal, and the others are called external. We use a symbol \( \Sigma \) to denote the set of all possible labels hereafter. \( \Sigma \)\( - \{ \tau \} \) denotes the set of all possible external labels.

Definition 1 (Labeled WF-nets [3]) A labeled Petri net \( N=\langle P, T, A, \ell \rangle \) is a labeled WF-net iff (i) \( N \) has a single source place \( p_1 \), i.e. \( p_1^*=\phi \) and \( \forall p\in P-\{p_1\}:p^*\neq\phi \), and a single sink place \( p_0 \), i.e. \( p_0^*=\phi \) and \( \forall p\in P-\{p_0\}:p^*\neq\phi \), and (ii) every place or transition is on a path from \( p_1 \) to \( p_0 \).

The modifier labeled of a WF-net or a transition may be omitted if clear from the context. \( [p_1] \) and \( [p_0] \) denote the initial state and the final state of \( N \), respectively. A set can also be used as a multi-set. The Petri net obtained by connecting \( p_0 \) with \( p_1 \) via an additional transition \( t^* \) is called the short-circuited net of \( N \), denoted by \( \mathcal{N} := \langle \langle P, T, \{t^*\}, A, \ell \rangle \rangle \). Let \( x \) be a place or a transition. A WF-net is said to be free choice (FC for short) if \( \forall p_1,p_2\in P: p_1^*\cap p_2^*\neq\phi \Rightarrow p_1^*=p_2^* \) holds.

Soundness is a correctness condition defined for WF-nets. Intuitively, a WF-net \( N \) is said to be sound iff, for any case, the initial state is transformed to the final state, and \( N \) has no dead transitions.

Definition 2 (Soundness [2]) A WF-net \( N \) is sound iff (i) \( \forall M\,:\, [p_1]^*M \Rightarrow M[p_0] \), (ii) \( \forall M\,:\, ([p_1]^*M \models M[p_0]) \Rightarrow M=[p_0] \), and (iii) \( \forall t\in T, \exists M, M'\,:\, [p_1]^*M[t] \models M' \).

2.2 Behavior and Equivalence of WF-Nets

Branching bisimilarity is widely used as an equivalence relation on WF-nets. Intuitively, branching bisimilarity is a behavioral equivalence that equates WF-nets with the same...
externally observable behavior but possibly different internal behavior. For the details, refer to [4].

\((N_A,[p^A_1])\) and \((N_B,[p^B_1])\) are called branching bisimilar, denoted by \((N_A,[p^A_1]) \sim (N_B,[p^B_1])\) iff there exists a branching bisimulation \(\mathcal{B}\) between the reachability graph of \((N_A,[p^A_1])\) and that of \((N_B,[p^B_1])\).

### 2.3 Behavioral Inheritance of WF-Nets

Intuitively, an interworkflow is a subclass of a workflow if all the activities of the interworkflow can be executed in the same order as the workflow by hiding all the activities newly added into the interworkflow. To give the formal definition of behavioral inheritance of WF-nets, we use abstraction operator. Abstraction is an operator to hide external transitions into internal ones.

**Definition 3** (Abstraction operator[4]) Let \(N=(P,T,A,\ell)\) be a WF-net. For any \(I (\subseteq \Sigma - \{\tau\})\), the abstraction operator \(\tau_I\) is a function that renames all transition labels in \(I\) to \(\tau\). Formally, \(\tau_I: N=(P,T,A,\ell)\) such that for any \(t \in T\), \(\ell(t) \in I\), \(\ell'(t)=\tau\). Otherwise \(\ell'(t)=\ell(t)\).

Let a WF-net \(N_A\) be a subclass of a WF-net \(N_B\). A WF-net \(N_A\) is said to be a subclass under projection inheritance of a WF-net \(N_B\) if the behaviors of \(N_A\) and \(N_B\) cannot be distinguished by hiding some of the transition newly added into \(N_A\).

**Definition 4** (Projection inheritance [3]) Let \(N_A=(P_A, T_A, A_A, \ell_A)\) and \(N_B=(P_B, T_B, A_B, \ell_B)\) be WF-nets, respectively. Let \(p^A_1\) and \(p^B_1\) be the source places of \(N_A\) and \(N_B\), respectively. \(N_A\) is a subclass under projection inheritance of \(N_B\) iff there is an \(I (\subseteq \Sigma - \{\tau\})\) such that \((\tau_I(N_A),|p^A_1\rangle) \sim (N_B,|p^B_1\rangle)\).

**Property 1** ([3]) Projection inheritance relation is partial-order.

#### 2.4 Labeled WF-Net Based Modeling Method of Interworkflows

We give a method to model interworkflows based on the WfMC protocol in terms of labeled WF-nets. In the protocol, there are three patterns of interoperability: Chained, Nested, and Parallel synchronized; and an interworkflow is constructed from existing workflows by using those interoperability patterns. The Parallel synchronized pattern is as follows [5]:

**Definition 5** (Parallel synchronized pattern) A workflow runs parallel with another workflow, but they synchronize at specified pairs of activities, i.e. having reached a specified activity, a workflow waits for the other to reach the corresponding activity.

It is known that most actual workflows can be modeled as sound free choice WF-nets (FCWF-nets for short). Soundness is a condition of logical correctness defined for WF-nets. Therefore we assume that the WF-net representing an interworkflow is constructed from sound FCWF-nets. Figure 1 shows two examples of WF-nets. If we connect \(N_1\) with \(N_2\) via a synchronization net, the resultant WF-net \(N_3\) is shown in Fig. 2.

---

![Figure 1. Examples of WF-nets.](image1.png)

![Figure 2. The WF-net representing interworkflow \(i_{old} (N_3)\), which is obtained by connecting the bank’s workflow \((N_1)\) and the builder’s workflow \((N_2)\).](image2.png)
and sufficient condition of soundness [6]. The condition is given by the following property.

**Property 2**: An FCWF-net \( N \) is sound iff

(i) \( \text{Rank}(A_N) = |\mathcal{C}_N| - 1 \) holds, where \( A_N \) is the incidence matrix of \( N \), \( \text{Rank}(\cdot) \) is the rank of a given matrix, and \( \mathcal{C}_N \) is the set of clusters \( 2 \) in \( N \), and

(ii) Every proper siphon is marked at \([pr]\). □

Furthermore it is known that the condition can be checked in polynomial time by using the method given in Ref. [7]. Thus if a given interworkflow is constructed by using the Parallel synchronized pattern and can be modeled as an FCWF-net, we can verify the behavioral inheritance of the interworkflow in polynomial time.

Let us consider computation complexity to verify the Condition (ii) of Theorem 1, i.e., \( \forall t_i \in T_A \cap P_S, \forall t_j \in T_A \cap P_S^{\cdot} : \) There exists a path between \( t_i \) and \( t_j \). We can decide whether there exists a path between \( t_i \) and \( t_j \) by using DFS (Depth-First Search) twice. The first DFS starts from \( t_i \). The second DFS starts from \( t_j \). If one of those DFSes finds the pair of the start transition, there exists the path between \( t_i \) and \( t_j \). The computation complexity of those DFSes is \( O(|P_A| + |T_A| + |A_A|) \). Thus, the computation complexity of the Condition (ii) of Theorem 1 is totally \( O(|T_A|(|P_A| + |T_A| + |A_A|)) \).

For example, let us consider the behavior inheritance between workflow \( N_1 \) shown in Fig. 1 and interworkflow \( N_3 \) shown in Fig. 2. \( N_3 \) satisfies Condition (i) of Property 2 because \( \text{Rank}(N_3) = 12 \) and \( \text{cluster}(N_3) = 13 \). In addition, \( N_3 \) satisfies Condition (ii) of Property 2. And there exists a path between \( t_1^1 \) and \( t_2^2 \). Therefore \( N_3 \) is a subclass under projection inheritance of WF-net \( N_1 \).

### 4. Application

As an application of the proposed condition, we give an example of substituting a part of a workflow for an interworkflow. The interworkflow is for constructing houses, which a builder and a bank participate in. The builder designs a house and estimates the cost for the client. The bank inspects the loan document for the cost, and makes the round robin for the loan document. Simultaneously the bank inspects the client’s credit and makes the report of the credit inspection. Then the bank discusses the loan document and the client’s credit, and settles the financing of the loan. The builder starts to construct the house. After the house is completed and delivered to the client, the interworkflow is finished. Let us consider a new cooperation between the builder and an inspection institution. This cooperation should not affect the cooperation between the bank and the builder. Without any side-effect, the inspection institution’s workflow should be added to the existing interworkflow.

This example can be modeled as follows: The old interworkflow, denoted by \( I_{\text{old}} \), is shown in Fig. 2, and is composed of two workflows: the bank’s workflow (see Fig. 1(a)) and the builder’s workflow (see Fig. 1(b)). The new interworkflow, denoted by \( I_{\text{new}} \), is shown in Fig. 5, and is composed of the two workflows and the inspection institution’s workflow (see Fig. 3). The inspection institution’s workflow is firstly connected with the builder’s workflow as an interworkflow, denoted by \( I \) (see Fig. 4). Interworkflow \( I \) is connected with the bank’s workflow. This change does not affect the synchronization between the bank and the builder.

The reason is as follows: Interworkflow \( I \) is a subclass under projection inheritance of the builder’s workflow \( N_2 \). Actually, interworkflow \( I \) is soundness because incidence matrix \( \text{Rank}(I) = 9 \) and \( \text{cluster}(I) = 10 \). And, there clearly exists path between \( t_1^2 \) and \( t_1^3 \). Interworkflow \( I_{\text{new}} \) is under projection inheritance of interworkflow \( I \). Therefore, This change does not affect the synchronization between the bank’s workflow and the builder’s workflow.

Therefore we can say from Theorem 1 that interworkflow \( I \) is a subclass under projection inheritance of the builder’s workflow, i.e. all the activities of interworkflow \( I \) can be executed in the same order as the builder’s workflow. Actually, we compare interworkflow \( I_{\text{new}} \) with interworkflow \( I_{\text{old}} \), the bank performs financing acceptance after the design/estimate of the builder with both interworkflow. And a builder starts construction after the discussion/settlement of the bank. Thus the synchronization between the bank and the builder needs not to be changed.

In this example, using the proposed condition, we can ver-
ify that the interworkflow inherits the behavior of the builder’s workflow in polynomial time. Therefore we can effectively guarantee that the interworkflow can be substituted for the builder’s workflow.

5. Conclusion

In this paper, we have proposed a necessary and sufficient condition to verify Parallel synchronized interworkflows. This condition enables us to verify Parallel synchronized interworkflows in polynomial time. Furthermore, as an application of the proposed condition, we have given an example of substituting a workflow for another workflow. In this example, an interworkflow is constructed by connecting the workflow of a builder with that of a inspection institution. Using the proposed condition, we can verify that the interworkflow inherits the behavior of the builder’s workflow in polynomial time. Therefore we can effectively guarantee that the interworkflow can be substituted for the builder’s workflow.

As a future work, we apply our results on behavioral inheritance to analysis of dynamic change of a workflow to an interworkflow.

Acknowledgement

This research is partly supported by Grant-in-Aid for Scientific Research (C) 17560342 of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References