Verification of Itineraries for Mobile Agent Enabled Interorganizational Workflow

Sea Ling† and Seng Wai Loke‡
†School of Computer Science and Software Engineering
‡CRC for Enterprise Distributed Systems Technology
Monash University, Caulfield VIC 3145, Australia
{sling,swloke}@dstc.monash.edu.au

Abstract

We apply mobile agents to synchronize workflows of different organizations resulting in an interorganizational workflow. This paper models agent enabled interorganizational workflows using Interorganizational Workflow Nets. This model provides a means to verify the correctness (and, so the viability) of the itineraries of agents used in enacting interorganizational workflows. We also outline an algorithm for our verification method. We believe that our formal apparatus provides a sound basis for building mobile agent enabled interorganizational workflows that work.

1. Introduction

Mobile agents are software components that can move from host to host to perform computations. Mobile agents run within agent server programs which we call places (each with an address (e.g., www.dstc.edu.au:888)) that receive agents and execute them. An agent can utilise the resources at places (e.g., if the place is interfaced with a database or with a person). Mobile agents may exhibit other agent attributes such as autonomy, social ability, and adaptivity. Recent work is exploring mobile agents for efficient database querying and workflows [6].

In earlier work [8], we have proposed that agent itineraries can be mapped onto a particular type of Petri nets [9] called Workflow nets [2], and each participating organization's internal workflow can also be represented as a Workflow net. This collection of Workflow nets and the agent's itinerary expressed as a Workflow net can be combined into one interorganizational Workflow net and analyzed together. In this paper, we further our work on modeling agent enabled interorganizational workflows based on Interorganizational Workflow Nets [4]. This model provides a means to verify the correctness (and, so the viability) of the itineraries of agents used in enacting interorganizational workflows: given an agent's itinerary and the relevant workflows of the participating organizations, we can verify if an agent can complete the itinerary and is not blocked or unnecessarily waiting. Blocking can occur due to the agent's interaction with several organizations' own internal workflows. For example, an agent is blocked if the task of an agent (as specified in the agent's itinerary) is dependent upon the completion of some task in an organization's internal workflow and that task is not completed.

In Section 2, we provide definitions of Workflow nets and the itinerary algebra we use for specifying agent itineraries. Then, in Section 3, we formalize the correctness of agent itineraries in a Petri net model, illustrate our method with examples, and describe how agent itineraries are verified. We conclude in Section 4.

2. Preliminaries

An Itinerary Algebra. We provide a brief description of the itinerary algebra (the reader is referred to [7] for a full account). We assume an object-oriented model of agents (e.g., with Java in mind), where an agent is an instance of a class given roughly by: mobile agent = state + action + mobility. Also, agents have the capability of cloning, that is, creating copies of themselves with the same state and code. Agents can communicate to synchronize their movements, and the agent's code is runnable in each place it visits.

Let $A$, $O$ and $P$ be finite sets of agent, action and place symbols, respectively. Itineraries (denoted by $I$) are now formed as follows representing the null activity, atomic activity, parallel, sequential, nondeterministic, conditional nondeterministic behaviour, and have the following syntax:

$$I ::= 0 \mid A^* \mid (I || u \cdot I) \mid (I \cdot \cdot I) \mid (I \cdot I) \mid (I ; I)$$

where $A \in A$, $a \in O$, $p \in P$, $\oplus$ is an operator which, after...
a parallel operation causing cloning, recombinates an agent
with its clone to form one agent, and II is an operator which
returns a boolean value to model conditional behaviour. We
specify how \( \oplus \) and II are used but we assume that their
definitions are application-specific.

We assume that all agents in an itinerary have a starting
place (which we call the agent’s home) denoted by \( h \in P \).
Below, we describe the meaning of the operators informally.
Given an itinerary \( I \), we shall use \( \text{agents}(I) \) to refer to the
agents mentioned in \( I \).

**Agent Movement** \((A_{\oplus}^p)\. \) \( A_{\oplus}^p \) means “move agent \( A \) to place
\( p \) and perform action \( a \)”. This expression involves one
agent, one move and one action at the destination.

**Parallel Composition** (“||”). Two expressions composed
by “||” are executed in parallel. For instance, \((A_{\oplus}^p || B_{\oplus}^q)\)
means that agents \( A \) and \( B \) are executed concurrently. Parallelism
may imply cloning of agents. For instance, to execute the expression \((A_{\oplus}^p || A_{\oplus}^q)\),
where \( p \neq q \), cloning is needed since agent \( A \) has to perform actions at both \( p \) and \( q \)
in parallel. In the case where \( p = q \), the agents are cloned
as if \( p \neq q \). In general, given an itinerary \((I \mid J)\) the agents
in \( \text{agents}(I) \cap \text{agents}(J) \) are cloned and although having
the same name are different agents. When cloning has
occurred, decloning is needed, i.e. clones are combined using
an associated application-specific operator (denoted by
\( \otimes \) as mentioned earlier). For example, given the expression
\((A_{\oplus}^p || A_{\oplus}^q) \cdot A_{\oplus}^r \) and suppose that after the parallel operation,
the configuration has clones. Then, decloning is carried out
before continuing with \( A_{\oplus}^r \). The resulting agent from decloning resides in the original place (in this case \( s \)).

**Sequential Composition** (“.”). Two expressions composed
by the operator “.” are executed sequentially. For example,
\((A_{\oplus}^p, A_{\oplus}^q) \) means move agent \( A \) to place \( p \) to perform action
\( a \) and then to place \( q \) to perform action \( b \).

**Independent Nondeterminism** (“||”). An itinerary of the
form \((I \mid J)\) is used to express nondeterministic choice: “I
don’t care which but perform one of \( I \) or \( J \)”. If \( \text{agents}(I) \cap \text{agents}(J) \neq \emptyset \), no clones are assumed, i.e. \( I \) and \( J \)
are treated independently.

**Conditional Nondeterminism** (“\(|\).”) Independent non-
determinism does not specify any dependencies between
its alternatives. We introduce conditional nondeterminism
which is similar to short-circuit evaluation of boolean
expressions in programming languages such as C.

**Workflow Nets.** A workflow (or business process) is a set
of coordinated tasks to fulfill a specific business purpose
[12], and Petri nets [9] have been widely used for process
specification and verification. In a special class of Petri
nets, called Workflow nets (WF-nets) [2], workflow concepts
are modelled by Petri net elements: workflow activi-

---

3An S-element is more commonly called a “place” in Petri net terminology. However, since there is a conflict in its meaning with mobile agent locality, we use S-elements for our Petri net description instead.
analysis methods and the soundness criteria for a single WF-net, mentioned earlier. For completeness, we reproduce the definition of IOWF soundness [3] as follows:

**Definition 2.1 (IOWF Soundness)** An IOWF-net is sound if and only if it is locally sound and globally sound. The IOWF-net is locally sound if and only if each of its local WF-nets $N_k (1 \leq k \leq n, n \in \mathbb{N})$ is sound. The IOWF-net is globally sound if and only if the unfolded IOWF-net is sound.

3. Agent Itinerary Verification

Intuitively, the semantics of IOWF-net resembles the behaviour of mobile agent(s) interacting with several organizations at different locations. An agent with specific itinerary moves among workflow processes to activate and synchronize the organizations' local tasks. Effectively, the agent itinerary can be represented by a single WF-net, whose transitions are the tasks at different locations specified in the itinerary. We call such a WF-net an Agent Itinerary (AI) net.

This AI-net together with the WF-nets of the participating organizations' workflows form an IOWF-net representing the whole interorganizational workflow. In other words, the AI-net is "embedded" in this IOWF-net.

The three requirements for soundness mentioned in Section 2 relate to the dynamics of a WF-net. Intuitively, they collectively infer (from the agent's perspective) that the agent must complete its itinerary by reaching the final state without any unnecessary blocking and waiting. Using the Petri net notation $M_1 \rightarrow M_2$ where $\sigma$ denotes a task execution sequence that leads from state $M_1$ to state $M_2$, we define a notion of agent liveness:

**Definition 3.1 (Agent Liveness)** Let $A$ be an AI-net interacting with $n$ number of WF-nets ($N_1 \cdots N_n$) in an IOWF-net with an initial state $M_0 = i_{N_1} + \cdots + i_{N_n} + i_A$. An agent $\alpha$, whose behaviour is specified by $A$, is a-live if and only if the following conditions hold: 1. $A$ is sound; and 2. For every transition sequence $\sigma$ generated in $A$, such that $i_A \xrightarrow{\alpha} o_A$, there is a corresponding sequence $\sigma'$ generated in IOWF such that $M_0 \xrightarrow{\sigma'} M_f$, where $M_f$ is the final state and $o_A \in M_f$; and $\sigma$ is a subsequence of $\sigma'$.

Condition 1 is a local condition. It states that the stand-alone AI-net must be sound locally and the agent must perform the tasks assigned in the itinerary and reach its final state, $o_A$. Condition 2 insists that in an interorganizational, cooperative and global environment, the agent must also be able to reach its final state. Agent liveness does not imply global soundness. However, global soundness does imply agent liveness. To prove this, we first prove the following lemma.

**Lemma 3.2**

If an IOWF-net is sound, then the terminating transition $t_o$ of the unfolded IOWF-net will eventually fire if the global source $i$ is initially marked.

**Proof:** From the definition of soundness, the global sink $o$ will be eventually marked. This means that $t_o$ will eventually fire to arrive at the final state $o$.

**Theorem 3.3 (Soundness implies Agent Liveness)**

Given an IOWF-net containing an AI-net $A$ of an agent $\alpha$, if the IOWF-net is sound, then $\alpha$ is a-live.

**Proof:** From Definition 2.1, if the net is sound, it is also locally sound, i.e., $A$ is sound and the unfolded net is also sound. Then, based on the topology of the unfolded IOWF and on Lemma 3.2, the following transition occurrence that leads from $i$ to $o$ in the unfolded net exists: $i \xrightarrow{\delta} (i_A + i_{N_1} + \cdots + i_{N_n}) \rightarrow \cdots \rightarrow M_f \xrightarrow{\delta} o$ for some marking $M_f$ and $o_A \in M_f$. Let $\sigma'$ be some occurrence sequence $t_i \cdots t_o$. By examining the markings generated in the transition occurrence, for each $\sigma$ generated in $i_A \xrightarrow{\alpha} o_A$ of an AI-net, $\sigma$ is a subsequence of $\sigma'$.

The following examples illustrate how to verify agent liveness and IOWF soundness using these concepts.

**Example 1: Voting.** An agent $V$, starting from home, carries a list of candidates from host to host visiting each voting party. Once each party has voted, the agent goes home to tabulate results (assuming that home provides the resources and details about how to tabulate), and then announces the results to all voters in parallel (and cloning itself as it does so). Assuming two voters (at place $p$ and place $q$), vote is an action accepting a vote (e.g., by displaying a graphical user interface), tabulate is the action of tabulating results, and announce is the action of displaying results, the mobility behaviour is:

$$V_p \xrightarrow{\text{vote}} V_q \xrightarrow{\text{vote}} V_{\text{tabulate}} \cdot (V_{\text{announce}} || V_q)$$

Figure 1 shows the corresponding IOWF-net which consists of two local workflows at places $p$ and $q$ respectively and an AI-net for agent $V$. There are four synchronous communication elements: $sc1$, $sc2$, $sc3$ and $sc4$. In the sense of [3], a synchronous element forces local workflow processes to execute specific tasks at the same time. In other words, the execution of a local task is dependent not only on local conditions, but also on the external environment. In
Petri net terminology, a synchronous communication corresponds to the fusion or the melting of a number of transitions. For the purpose of Petri net analysis, the IOWF-net in Figure 1 is unfolded, synchronous communications are fused and a transition \( t^* \) is included to short-circuit the net. The result is shown in Figure 2, which turns out to be a live and bounded Petri net. Therefore, the IOWF-net is globally sound. Individually, the WF-nets and the AI-net are locally sound, and hence, agent \( V \) is a-live.

**Example 2: Sales Order Processing.** The scenario for processing sales orders in a virtual enterprise was first proposed in [10] and then described in our previous work [7, 8]. Each sales order is carried out by a mobile agent which moves through several entities to process the order. Let \( us_{-}sc \) be a place where the agent can interact with the US stock control, \( asia_{-}sc \) be a place where the agent can interact with the Asian stock control, \( mat \) be a place where the agent can purchase raw materials for manufacturing the products requested in a sales order, \( man \) be the place where the agent can place an order for products to be manufactured (i.e., \( man \) represents the manufacturer), and \( ext \) be a place where the agent can interact with an external buyer. Also, let \( query \) be an action where the agent queries a stock control, \( report \) be an action where the agent reports the results of a completed sales order, \( buy_{-}raw \) be the action of purchasing raw materials, \( buy_{-}prod \) be the action of buying products for a sales order, and \( order \) be an action of placing an order to have some products manufactured.

As in [8], we model a particular business logic, expressing independent non-deterministic choice for the agent. Its essence is captured in its itinerary as follows:

\[
(A_{query} \cdot A_{report}^a) \mid (A_{query} \cdot A_{report}^b) \mid (A_{buy_{-}raw} \cdot A_{order} \cdot A_{report}) = (A_{query} \cdot A_{report}^a) \mid (A_{buy_{-}raw} \cdot A_{order} \cdot A_{report}) \cdot A_{buy_{-}prod}^a
\]

(by distribution of \( \cdot \) over \( \mid \) )

Figure 3 shows the interaction of the agent with six possible workflow processes or WF-nets which exist in different places or organizations. Task related transitions are fused for synchronous communication. The unfolded IOWF-net is shown in Figure 4. Using Petri net analysis methods, it is not live and therefore not sound. However, the AI-net is locally sound and agent \( A \) is a-live. In other words,
agent liveness does not imply IOWF soundness. No matter which non-deterministic task option agent A chooses, agent A will always arrive at the final state $o_A$. Only organizations whose tasks are being selected by the agent will reach its final local state $o$. Those that are not selected will remain in their initial state $i$ and will never reach $o$.

As mentioned earlier, our example assumes that only one single, trivial task exists in each organization and the organization is too dependent on the agent - it will only complete its process if the agent comes to visit! In reality, an organization is expected to have more than one tasks executing in parallel, in sequence and/or as non-deterministic choices. To make the IOWF-net globally sound, each local WF-net must be expanded to include more local tasks, so that its final state $o$ is also reachable via other alternative paths of execution.

Verification Algorithm. Agent liveness is closely related to IOWF soundness. It is insufficient for agent verification to only ensure a locally sound AI-net. The agent must also be a-live with respect to its external environment, i.e., the organizations it is interacting with. In other words, the IOWF-net must also be sound.

Given an agent itinerary, we provide the following steps for agent verification: (1) Translate the itinerary to an AI-net. (2) Construct the WF-nets of the organizations the agent is interacting with. (3) Determine the synchronising relations between the AI-net and the WF-nets, and fuse the appropriate transitions to form an IOWF-net. (4) Unfold the IOWF-net. (5) Verify IOWF soundness by using Petri net’s liveness and boundedness checks.

The soundness property of any large and complex WF-net can be difficult to verify, especially for an IOWF-net, involving different organizations with a large number of states and tasks. It has been pointed out that IOWF-soundness is decidable but EXSPACE-hard [1]. To make verification methods more efficient, workflow processes can be specified in terms of two subclasses of Petri nets: free-choice and well-structured Petri nets [1]. Structural analysis methods [5, 1] without the need to generate large state spaces can then be employed. Analysis tools such as Woflan [11] dedicated to the analysis of workflows specified in Petri nets can be used. It has been shown that the soundness property of such nets can be verified in polynomial time.

4. Conclusion

We have defined agent liveness in the context of a mobile agent enacting an interorganizational workflow by moving among the participating organizations. We have also studied formal relationships between agent liveness and IOWF-net soundness, and outlined an algorithm for our verification method. We believe that our formal apparatus provides a sound basis for building mobile agent enabled interorganizational workflows that work.

Further investigation will seek to extend our analysis to involve multiple mobile agents in interorganizational workflows. We already have concurrency with cloning of agents but aim to consider itineraries which mention multiple agents as possible in theory in our itinerary algebra.

References


