Communication based workflow loop formalization using Temporal Logic of Actions (TLA)

José L. Caro¹, Antonio Guevara¹, Andrés Aguayo¹, and José L. Leiva¹

¹ Computer Science Department, Malaga University, 29071 Málaga, Spain

Abstract. The workflow map development for an organization is a highly complex process. Therefore, the workflow map should be tested and validated before it is implemented (into a WfMS). Most current workflow systems deal with this validation issue by using simulation modules that “execute” the model and examine the possible problems before it is truly “executed” and implemented. Although these simulation modules are very useful for the management team to detect problems in the business processes represented by the workflow, it would be advisable to find other more reliable methods. In this paper we propose a formal method based on Temporal Logic of Actions to formalize workflow maps (based on a communication modelling methodology).

1 Introduction

Actually there are two main research areas related to workflow technology [6]: (i) workflow management systems (WfMS) (the implementation of workflow technology), and (ii) workflow modelling techniques (the capability to express the process, work, information, methods, etc. for an organization into a specification called workflow map).

A workflow map could describe business processes at the conceptual level needed for the evaluation, understanding and design of such business processes, as well as capturing information process tasks at level that describes such process requirements for information systems and human skills [2]. This model should allow and facilitate the automated demonstration of properties. For example: will any workflow never be executed? Will this workflow ever be executed? Is the operation carried out with a specified time cost? Formal proving mechanisms will provide a practical solution to these kinds of problems [3].

In this paper, our objective is to develop a language/action paradigm formalization [5], based on an extension of temporal logic, known as Temporal Logic of Actions (TLA) [4]. This paper is organized as follows: we begin with a brief description of workflow (sec. 2). In section 3, the TLA elements needed for the formalization are described. In section 4, we develop the TLA formalization of the language/action paradigm. The last section summarizes the paper conclusions.

2 Workflow

Workflow has a wide range of possibilities related to group support and the automation of organizational processes. In general terms we can define workflow as [9]: “Workflow
is comprised by a set of activities dealing with the coordinated execution of multiple
tasks developed by different processing entities in order to reach a common objective”.
A good approach to workflow modelling techniques, that includes a formal method for
achieve demonstrations, can be found in [1, 7, 10, 6, 2].

Communication-based methodologies stem from the “Conversation for Action” model
developed by Medina-Mora, Winograd, and Flores [5]. The communication between
client (Cli) and server (Svr) can be defined in four steps (figure 1): request/preparation,
negotiation, development and acceptance. The language/action perspective modelling
is based on the “speech-act” theory developed by John Searle [8] that has been adapted
for workflow management (statechart).

The diagram starts at state 1, opening the conversation (by Cli), that triggers the
transition to state 2, where the server Svr has three options: promise, perform the work
(state 3); refuse, closes the conversation (state 8); counteroffer, negotiation of satis-
faction terms (state 6). The simplest path is going through the following transitions:
promise, the task is accepted (from state 2 to 3); report, the accepted task is done (from
state 3 to 4); declare, the service is accepted (from 4 to 5). Other option is if Svr
initiates a counteroffer (from state 2 to 6), Cli has three options: accept, state 3; coun-
teroffer, back to state 2; decline, the service is refused (state 8). From state 3 there are
this additional options renege, not performing (from 3 to 7) and withdraw the request
(state 3 to 9). After Svr reporting that the work is concluded the possible actions are:
declare, the work has not been concluded satisfactorily and Svr has to do it again (from
state 4 to 3) and withdraw the petition (from state 4 to 9).

3 Temporal Logic of Actions

Temporal Logic of Actions (TLA) combines two logics the action logic and temporal
logic [4]. All TLA formulas are TRUE or FALSE in a behavior σ defined as an infinite
sequence of states \(<s_0, s_1, s_2, \ldots >\).

3.1 Elements of State Logic in TLA

- Variables. The semantic of \([x](s)\) is the function of the value of \(x\) in the state \(s\).
- **State and predicate functions.** The semantics of $[[f]]$ is a mapping of states into values.

- **Actions.** We obtain $[[A]](s, t)$, by first replacing each variable $x$ with $[[x]](s)$ and each variable $x'$ with $[[x]](t)$ to later evaluate the expression. It is said that the state pair $(s, t)$ is a “A-step” iff $[[A]](s, t)$ is TRUE.

- **Active action in a state.** An action $A$ is said to be active in a state $s$ if there is a state $t$ such that $(s, t)$ is a A-step: $[[\text{Enabled } A]](s) \iff \exists t : [[A]](s, t)$.

### 3.2 Elements of Temporal Logic in TLA

In TLA, the system behavior is modelled as an infinite sequence of states. The basic elements are actions and temporal logic:

- **Predicates.** A behavior satisfies the predicate $P$ iff (if and only if) it is satisfied in the first state: $[[P]](<s_0, s_1, s_2, \ldots>) \Rightarrow [[P]](s_0)$. Similarly, a behavior satisfies the action $A$ iff the first pair of states of the given behavior is an A-step: $[[A]](<s_0, s_1, s_2, \ldots>) \Rightarrow [[A]](s_0, s_1)$.

- **“Always” ($\Box$) operator.** Given a formula $F$, $\Box F$ asserts that $F$ is always TRUE: $[[\Box F]](<s_0, s_1, s_2, \ldots>) \iff \forall n \geq 0 : [[F]](<s_n, s_{n+1}, s_{n+2}, \ldots>)$.

- **“Eventually” ($\Diamond$) operator.** The formula $\Diamond F$ asserts that $F$ is eventually TRUE: $[[\Diamond F]](<s_0, s_1, s_2, \ldots>) \iff \exists n \geq 0 : [[F]](<s_n, s_{n+1}, s_{n+2}, \ldots>)$.

- **Validity.** A formulae $F$ is valid iff it is satisfied for all behaviors ($\models F \equiv \forall \sigma \in S^\infty : [[F]](\sigma)$), where $S^\infty$ denotes the set of all possible behaviors: $\models F \equiv \forall \sigma \in S^\infty : [[F]](\sigma)$.

- **Specification in TLA.** A formal specification has the following general formula: $\Pi \models \text{Init} \land \Box (A_1 \lor A_2 \lor \ldots \lor A_n)$.

- **Fairness Operators.** The fairness operators are in charge of ensuring that “nothing abnormal will happen”:
  - **Weak fairness (WF).** Asserts that an action has to be infinitely executed frequently if it is continuously enabled for an infinitely long time: $WF_e(A) \equiv \Box(\Diamond(A) \lor \Box(\Diamond(\neg\text{Enabled}(A)))).$
  - **Strong fairness (SF).** Asserts that an action has to be infinitely executed frequently if it is often infinitely enabled: $SF_e(A) \equiv \Box(\Diamond(A) \lor \Box(\Diamond(\neg\text{Enabled}(A)))).$

- **Formal system specification.** A formal specification of a system, within fairness conditions, can be represented as: $\Pi \models \text{Init} \land \Box[N]_w \land L$.

### 3.3 Formal approach to state diagram modelling

$\Delta$ formula represents the statechart (equation 1).

$$
\begin{align*}
\text{Init}_\Delta & \doteq \exists n \in I : P_n \\
\mathcal{A}_n & \doteq \exists e \in E(n) : e \in E(n) \land P'_n \land \exists d(e) \\
\Delta & \doteq \text{Init}_\Delta \land \forall n \in N : \Box[P_n \Rightarrow \mathcal{A}_n]_w
\end{align*}
$$

(1)

Where, $N$: set of nodes; $I$: set of initial nodes; $E(n)$: set of edges originating at node $n$; $d(e)$: destination node of edge $e$; $P_n$: predicate labelling node $n$; $e$: action labelling edge $n$. 
### 4 Formalizing the language/action paradigm

In order to formalize in TLA the state diagram its edges are labelled $e_{ij}$ as shown in figure 1. Work $W_k$ is a quadruple expressed in the equation $W_k = \{I, H, P, SC, V\}$ where: $W_k.I$: information; $W_k.H$: tools and methods; $W_k.P$: human, role or agent to develop the task; $W_k.SC$: terms of work satisfaction; $W_k.V$: set of state variables belonging to the workflow. The set of nodes in our diagram will be denoted by $N_k$, where $k$ is the number of the current state (initial state $N_1$). In this way, and bearing in mind the equation to be satisfied in the initial state, we have $Init = \exists n \in I : P_n$. Therefore, to complete the definition of the initial state we have to give meaning to $P_{N_1}$ (equation 2).

$P_{N_1} = DetectTrigger(Type, Origin, Workflow)$ \[ (2) \]

The workflow IDWF is instantiated and the requested external work is assigned to the execution model that includes work $XW_k$, that will be assigned by the function $SelAgent(OrgDB, B)$ (assign the organizational knowledge database OrgDB) and the initial terms of satisfaction $XW_k.SC$. In the state $N_2$, the work has to be evaluated. The petition and the result of such evaluation is resent by $EvalWk(W_k, Agent)$, where $W_k$ is the work to be evaluated $W_k$. The response is obtained from $CounterOffer$ is possible to return: “commitment”, “counteroffer” (and the consequent new $W_k$), “decline” or “cancel” (eq. 4). From state $N_2$ we can advance through edges $e_{22}$, $e_{26}$, and $e_{28}$. Edge $e_{28}$ leads to state $N_8$, which is a final state of uncompleted termination (eq. 5). Consequently, a predicate that will abort the execution of the workflow will be used in $N_8$. This will trigger an exception so that the system can take appropriate actions, therefore, $P_{N_8} = Exception(IDWF, "abort", W_k)$. The function $Exception$ triggers the exception of aborting the task, and goes back to TRUE when completed. The edge $e_{26}$ corresponds to a counteroffer from $B$ after the evaluation done in $P_{N_2}$ (eq. 6).

$P_{N_8} = S = EvalWk(W_k, B) \land (S = "commit." \lor S = "c.off." \lor S = "dec.")$ \[ (4) \]

$e_{26} = S = "decl." \land W_f.P' = "prep." \land W_f.S' = "abort" \land W_k.S' = "Svr\ab" \land W_k.S' = "neg." \land W_k.S' = "neg." \land W_k.S' = CounterOffer(W_k, W_f, Svr)$ \[ (5) \]
In \( N_0 \) another evaluation is carried out by \( A \) (equation 7). We have three options: reconsidering the offer i.e., transit to \( N_2 \) (eq. 8), refuse the offer and raise an exception (transit to \( N_8 \), eq. 9).

\[
P_{N_0} \triangleq S = \text{EvalWk}(W_k, W_f, \text{Clt}) \land (S = \text{“ac.”} \lor S = \text{“c.off.”} \lor S = \text{“docl.”}) \tag{7}
\]

\[
e_{62} \triangleq S = \text{“c.off.”} \land W_fP' = \text{“neg.”} \land W_fS' = \text{“act.”}
\land W_kS' = \text{“neg.”} \land W_k' = \text{ComunterOffer}(W_k, W_f, \text{Clt}) \tag{8}
\]

\[
e_{68} \triangleq S = \text{“dec.”} \land W_fP' = \text{“neg.”} \land W_fS' = \text{“abort”} \land W_kS' = \text{“Clt refusal”} \tag{9}
\]

The third option is to accept the work \( W_k \) and transit to \( N_3 \) (eq. 10). Similarly, if the evaluation leads to \( B \) accepting the work, we could directly transit from \( N_2 \) to \( N_3 \) (eq. 11).

\[
e_{63} \triangleq S = \text{“ac.”} \land W_fP' = \text{“devel.”} \land W_fS' = \text{“exec.”} \land W_kS' = \text{“acc.”} \tag{10}
\]

\[
e_{23} \triangleq S = \text{“ac.”} \land W_fP' = \text{“devel.”} \land W_fS' = \text{“exec.”} \land W_kS' = \text{“acc.”} \tag{11}
\]

To define the semantic of \( P_{N_3} \), we need the following functions: \( \text{Trigger}(t, W_f) \): trigger \( t \) to enable the sub-workflow \( W_f \); \( \text{Completed}(W_f) \): TRUE if \( W_f \) has been satisfactorily completed; \( \text{Aborted}(W_f) \): TRUE if \( W_f \) has terminated as abort; \( \text{Abort}(X) \): TRUE if \( X \) aborts the workflow. In \( P_{N_3} \), all workflow subtasks must to be executed. The following case has to take place: a) all subtasks \( a_i \) have to be completed; b) the incorrect termination of some subtasks has to be detected; or c) the client aborts the workflow. Let \( W_k, a_i \) each of the subtasks comprising the task of \( W_k \), then \( P_{N_3} \) is defined as the equations 12 and 13:

\[
P_{N_3} \triangleq \forall (a_i \in W_k, t_i/t_i = \text{Trg}(a_i)) \text{Trigger}(t_i, W_k, a_i) \land (\text{ExecP}) \tag{12}
\]

\[
\text{ExecP} \triangleq \exists a_i \in W_k/\text{Completed}(W_k, a_i)
\lor \exists a_i \in W_k/\text{Aborted}(W_k, a_i) \lor \text{Abort}(W_k, \text{Clt}) \tag{13}
\]

At this point two options are possible: aborting the execution and transit to state \( N_7 \) or \( N_6 \) depending on who aborted or transit to evaluation state \( N_4 \) (eq. 14, 15, 16 and 17).

\[
e_{39} \triangleq \exists a_i \in W_k/(W_k, a_i, W_f S = \text{“aborted”} \land W_k, a_i, W_h S = \text{“Svr ref. Wk”})
\land (W_f P' = \text{“exec.”} \land W_f S' = \text{“aborted”} \land W_k S' = \text{“Svr refusal”}) \tag{14}
\]

\[
e_{37} \triangleq \exists a_i \in W_k/(W_k, a_i, W_f S = \text{“aborted”} \land W_k, a_i, W_h S = \text{“Clt refusal”})
\land (W_f P' = \text{“exec.”} \land W_f S' = \text{“aborted”} \land W_k S' = \text{“Clt refusal”}) \tag{15}
\]

\[
e_{34} \triangleq \forall a_i \in W_k/(W_k, a_i, W_f S = \text{“accepted”} \land W_k, a_i, W_h S = \text{“completed”})
\land (W_f P' = \text{“eval.”} \land W_f S' = \text{“active”} \land W_k S' = \text{“Clt’seval.”}) \tag{16}
\]

\[
P_{N_4} \triangleq \text{EvalReport}(W_k, B) = \text{“correct”} \lor \text{EvalReport}(W_k, B) = \text{“incorrect”}
\lor \text{Exception}(\text{IDWF}, \text{“abortal”, } W_k) \tag{17}
\]
If the work satisfies the terms, there is a transition to successful state $N_5$. On the other hand, if it does not, we go back to the subtask execution state (this path is optional) and if it is aborted, it leads to $N_9$ (eq. 18, 19, and 20).

$$e_{49} \equiv W_f P' = \text{"evaluation"} \land W_f S' = \text{"aborted"} \land W_k S' = \text{"Cliejects"}$$

$$e_{45} \equiv W_f P' = \text{"complete"} \land W_f S' = \text{"completed"} \land W_k S' = \text{"Wkaccept"}$$

$$e_{43} \equiv W_f P' = \text{"execution"} \land W_f S' = \text{"active"} \land W_k S' = \text{"reexecute"}$$

State $N_5$ only has to send a completed signal: $P_{N_5} \equiv \text{Signal(IDWF, "WF completed", W_k)}$.

Once these definitions are completed and the model equations are applied to formalize a state diagram (eq. 1) we obtain the formal representation.

5 Conclusions

The use of formal methods based on logic in workflow modelling can establish an automated, formal, and robust reasoning mechanism that will successfully provide insight into these issues (conflict, deadlock, reachability, reliability and satisfiability). The application of TLA to workflow management systems provides three fundamental bases [3]: (i) theory: providing a theory with a valid and robust basis to carry out analysis; (ii) formalization: expressing workflow maps as TLA expressions; and (iii) analysis: providing a mechanism for the automated demonstration of workflow model properties.

References