PARTIAL RECOGNIZING ALGORITHM FOR VERIFICATION OF WORKFLOW PROCESSES

Armen Kostanyan
Anna Varosyan

Educational and Research Center of Information Technologies
Yerevan State University
1 A.Manoogian Str., Yerevan, Armenia, 0049
Email: armko@ysu.am, annavarosyan@yahoo.com

KEYWORDS
Model analysis, verification, approximation techniques.

ABSTRACT
The paper presents the formal definition of workflow process, its semantics and correctness assertion. A partial recognizing algorithm for formal verification of acyclic workflow processes is suggested. The algorithm uses molds that represent upper and lower approximations of environment state sets. The consideration of mold transformations on all branching paths allows to avoid the process execution and examination per each input. At the same time the use of molds makes the verification algorithm as a partially recognizing one: there is a possibility that the assertion about the workflow process correctness is neither confirmed nor rejected. The algorithm is applied to verify the workflow library processes provided by IBM for IT Infrastructure.

INTRODUCTION
Business Process Management is a key technology in the overall enterprise strategy of an organization (Leymann and Roller 2000). The business processes have to be specified by a language called workflow specification language (IBM 2001, MQSeries Workflow). It is very important to verify the business process correctness prior to the operation (Khoshafian 2004). The business process correctness can be checked by the process graph structural verification, such as cycle analysis, branching path synchronization and so on. The formal verification can be considered as another case of verification to verify the process semantic correctness. The formal verification requires knowledge about the underlying business process (internal structure of tasks, data flow, etc.). Two invariant conditions have to be specified for the process. First, the process precondition has to be satisfied prior to the process execution. Second, the postcondition has to be checked on each process execution end. The process is considered correct if the postcondition is of “true” value for all possible executions.

Currently, the most approaches to validate workflow systems have restrictions on workflow control structures that reduce the verification complexity. An idea of limiting control flow graph size gives linear complexity for the verification paradigm on a transactional logic based workflow (Davulcu et al. 1998). Another idea is proposed for formally modeling workflows by Petri nets. It is shown that when verifying certain soundness properties a well structured or free-choice (Van der Aalst et al. 2000) control flow leads to a polynomial verification complexity. There were many results in the area of workflow graph structural verification, such as the lack of synchronization, presence of deadlocks and cycles without exit. Most of results have used verification methods based on calculus (Xu and Yu 2007), graph search techniques (Perumal and Mahanti 2007), model checking (Pfeiffer et al. 2004) and propositional logic (Bi and Zhao 2004). However, the main interest was in general control structure properties, while the semantics of each specific task was not considered. No researches have been done by now on the verification to cover every aspect of workflow systems.

In this paper a method of partial verification of acyclic workflow processes is suggested that is based on molds (primitives for the description of environment state subsets). The idea of the method was applied to the verification of linear programs processing Boolean data (Kostanian 1995.) and tests for computational system components (Shoukourian et al. 1996, Shoukourian and Kostanian 1997). The mold technique also has been used for the verification of special type object-oriented programs (Kostanyan and Asatryan 2001).

The paper presents the formal definition of workflow process, its semantics and correctness assertion. Also, an extension of workflow process semantics is constructed that allows to use an acyclic workflow process control graph for the stepwise transformation of an initial mold into a final one. A conclusion about the workflow process partial correctness is made based on the match of final mold to the postcondition.

W-PROCESS
Definitions
The short description of formal workflow/BPM model which underlies a model of IBM’s MQSeries Workflow (Leymann and Roller 2000), is presented in this section.

The main model components are activities and connectors. The activities are associated with a context being defined
as data passing to an activity. It is called input container. An activity also returns data called output container. Some output container elements of activity can be passed to the input container elements of other activities or to the external memory. All data elements are collected in the set V. Control and data connectors provide connections between activities. A control connector has an associated Boolean predicate called transition condition.

A directed graph based on sets of activities and control connectors is called control flow of workflow process (W-process). The execution of W-process can be described in following general stages. At first, initial activities have to be executed and transition conditions of their outgoing control connectors should be calculated. Then, the set of ready activities have to be determined by the selecting activities whose all incoming transition conditions have been calculated. The most priority activity should be selected from this set, its join condition has to be calculated and the activity must be executed in the case of join condition truth-value. The execution of W-process will be stopped if the set of ready activities is empty.

Let give the formal definition of non-hierarchical workflow processes using following designations:

- If x=<xi1, ..., xi2> is a tuple, and 1≤i1, ..., i2≤n is a set of indices, then πi1,...,i2(x) denotes the tuple <xi1, ..., xi2>.
- If X is a set, the ϕ(X) denotes the set of all subsets of X.

**Definition:** A tuple P = <T, N, Φ, i, o, Ψ, Δ, M> is called W-process, if

1. T is a finite set of types having own set of values. The set of values for t ∈ T is denoted by DOM(t) and is called the domain of t. Each variable used in the model belongs to any type from T. Let designate the set of v variable values by DOM(v) or by DOM(t) if v belongs to the type t.
2. N is a finite set of activities having some priorities.
3. C is the finite set of transition conditions.
4. The set E ⊆ N × N × C is the set of connectors with the following restrictions:
   - E is unified: ∀e, e′ ∈ E : πi1,e(e) = πi1,e′(e) ⇒ πi1,e(e) = πi1,e′(e).
   - The G=(N, E) graph is acyclic.
5. Φ : N → Γ, where Γ is the set of all activity join conditions. The join condition of activity AεN is defined as a Boolean expression Φ(A) depending on elements of set A^c={eεE | πi2(e)=A}. Suppose Φ(A)=1, if A^c=∅.
6. i : N ∪ C → IV is input container map where IV is the set of input variable tuples of activities and conditions. Each input variable is unique and belongs to any type from T.
7. o : N → OV is output container map where OV is the set of output variable tuples of activities. Each output variable is unique and belongs to any type from T.
8. Ψ : N ∪ C → E, where E is the set of activities and conditions of all possible implementations. The implementation of activity AεN is defined as a map Ψ(A) : xεE, DOM(v) → xεE, DOM(v).

**The implementation of condition p ε C is defined as a map Ψ(p) : xεE, DOM(v) → {0, 1}.**

9. Δ : N × (N ∪ C ∪ {M}) → ∪AεN. bεN, εC Φ(A) × (i(A), B) is data map with following restrictions:
   a. [AεN, BεN ∪ C] ⇒ Δ(A, B) ⊆ Φ(A) × i(B).
   b. AεN ⇒ Δ(A, M) ⊆ Φ(A) × M.
   c. [AεN, BεN ∪ C ∪ {M}, <v1, v'> ∈ Δ(A, B), <v2, v''> ∈ Δ(A, B)] ⇒ v1=v2.
   d. ∀<v, v'> ∈ Δ(X, Y), DOM(v)=DOM(v').

10. M is an external memory. Each variable from M is unique and belongs to any type from T.

Let A^+= [eεE | πi2(e)=A] be the set of outgoing transition conditions from the activity A.

Suppose P is a workflow process and X ∈ N ∪ C. Let define the set of all states for X as States(X)=xεE,DOM(v), and the set of external memory states as States(M)=xεE,DOM(v). Let call B=xεN,εC∪{M} States(X) as a set of states for P and designate as b(X) the state of X ∈ N ∪ C ∪ {M} for bεB.

Suppose bεB, AεN. Let define the result of applying activity A to the state b as a state bA=b’ which can be presented as following:

(b’(X))= \{
(b(X)), \; if \; \forall v \in Φ(A) \{ < v, v' > \in Φ(A, X) \}
\}

for all XεN∪C∪{M}.

Let present an algorithm that describes the recursive semantics of W-process assuming that each activity has a unique priority. In general, the W-process semantics might be based on the nondeterministic choice of an activity from a set of activities having the same priority. It will make the semantic determination algorithm as nondeterministic. After some modifications the suggested approach of the process verification can be adjusted to this case.

**Semantics of W-Process**

Each W-process P calculates a mapping Ψ(P) : B→B, defining by the following algorithm.

**Algorithm S. Semantics of W-process.**

**Input:** Initial state initState∈B.

**Output:** Final state finState∈B.

**Remark:** Creates an initialization and calls SR.

**Method:**

1. S1. For all AεN do
   - counter(A) ← |A^+|.
   - If counter(A)=0, then insert A into the activities priority queue front that is sorted in non-decreasing order of priorities.
2. S2. For all eεE assume that μ(e) is undefined.

**Algorithm SR.**

**Input:**
- Current state currState∈B.
• priority queue \texttt{front} of activities,
• mapping \texttt{counter}: N \rightarrow \{0, 1, 2, \ldots\},
• partially defined mapping \(\mu \subseteq E \rightarrow \{0, 1\}\).

\textbf{Output:} Final state \texttt{finState} \(_E\).

\textbf{Method:}

SR1. If \texttt{front} is empty, then return \texttt{currState}.

SR2. Select most priority activity \(A \in \texttt{front}\) and remove \(A\) from \texttt{front}.

SR3. If \(\Phi(A) \land \langle \mu(e) \mid e \in \texttt{front}\rangle = 0\), then
- \texttt{nextState} \leftarrow \texttt{currState}.

SR4. Else, if \(\Phi(A) \land \langle \mu(e) \mid e \in \texttt{front}\rangle = 1\), then
- \texttt{nextState} \leftarrow (\texttt{currState}) \(_A\).
- For all \(e \in \texttt{front}\), evaluate \(\Phi\) and do a-c)
  a) \(\mu(e) \leftarrow \Psi(p)\) (nextState),
  b) \(\texttt{counter}(A') \leftarrow \texttt{counter}(A') - 1\),
  c) If \(\texttt{counter}(A') = 0\), then insert \(A'\) into \texttt{front}.

SR5. Return \(\texttt{SR(newState, front, counter, } \mu)\).

Let designate \(\Psi(p)(b) = bP\) and call the \(W\)-processes \(P_1\) and \(P_2\) equivalent, if \((bP_1)(\mathcal{M}) = (bP_2)(\mathcal{M})\).

\textbf{Correctness of \(W\)-Process}

\textbf{Definition} Let \(\alpha, \beta\) are two predicates depending on variables from \(\mathcal{M}\). Let call \(\alpha\) and \(\beta\) a precondition and a postcondition correspondingly. In this case, \(P\) is correct with respect to \(\alpha\) and \(\beta\), if
\[(\forall b \in B) \left( \alpha(b(\mathcal{M})) = 1 \Rightarrow \beta((bP)(\mathcal{M})) = 1 \right).\]

\textbf{EXTENSION OF SEMANTICS OF \(W\)-PROCESS}

\textbf{Definitions}

Let extend semantics of \(W\)-processes in following way. Suppose \(\mathcal{C} \subseteq B\), \(A \in \mathcal{C}\), and define an action of activity \(A\) applied to \(s\) as a set \(sA = s' \subseteq \mathcal{C}\) such that
\[s'(x) = \left\{ \begin{array}{ll} s(x), & \text{if } x = (A(x), X) \land v < v' \in A(X), \\ \Psi(A)\left(s(A(x))\right), & \text{if } v < v' \in A(X), \end{array} \right.\]
for all \(x \in \mathcal{N} \cup \mathcal{C} \cup \mathcal{M}\).

Let introduce some designations. Suppose \(p \in \mathcal{C}\) and \(\text{supp}(p) = \{b \in B\} \subseteq \Psi(p)\) (\(\tau = 1\)). Let \(F(x_1, \ldots, x_n)\) be a Boolean expression depending on \(x_1, \ldots, x_n\) and \(<s_1, \ldots, s_n>\) is a tuple of subsets of \(B\). Let designate
\[F(s_1, \ldots, s_n) = \bigcup_{s_1^{e_1}, \ldots, s_n^{e_n}} s_1^{e_1} \cap \cdots \cap s_n^{e_n},\]
where \(s_1^{e_1} = s_1, \ldots, s_n^{e_n} = B \setminus s\).

Let extend semantics of \(W\)-process that calculates a mapping \(2^B \rightarrow 2^B\) defining through following algorithm.

\textbf{Algorithm E: Extended Semantics of \(W\)-process.}

\textbf{Input:} Initial subset \(\texttt{initSet} \subseteq B\).

\textbf{Output:} Final subset \(\texttt{finSet} \subseteq B\).

\textbf{Remark:} Creates an initialization and calls \texttt{ER}.

\textbf{Method:}

E1. For all \(A \in \mathcal{C}\) do
- \(\text{counter}(A) \leftarrow \lceil A^\tau \rceil\).
- If \(\text{counter}(A) = 0\), then insert \(A\) into the activities priority queue \texttt{front} that is sorted in non-decreasing order of priorities.

E2. For all \(e \in E\) assume that \(\mu(e)\) is undefined.

E3. Return \(\texttt{ER(initSet, front, counter, } \mu)\).

\textbf{Algorithm ER.}

\textbf{Input:}
- Current subset \(\texttt{currSet} \subseteq B\),
- priority queue \texttt{front} of activities,
- mapping \texttt{counter}: N \rightarrow \{0, 1, 2, \ldots\},
- partially defined mapping \(\mu \subseteq E \rightarrow 2^B\).

\textbf{Output:} Final subset \(\texttt{finSet} \subseteq B\).

\textbf{Method:}

ER1. If \(\texttt{front}\) is empty, then return \(\texttt{currSet}\).

ER2. Select most priority activity \(A \in \texttt{front}\) and remove \(A\) from \texttt{front}.

ER3. \(s_1 = \Phi(A) \land \langle \mu(e) \mid e \in \texttt{front}\rangle, s_0 = B \setminus s_1\).

ER4. If \(s_0 \neq \emptyset\), then do
\[\text{front}' \leftarrow \text{front} \lor \text{counter}', \mu' \leftarrow \mu, \text{finSet}' \leftarrow \text{ER(currSet, front', counter', } \mu)\).

Else, if \(s_0 = \emptyset\), then \(\text{finSet}' \leftarrow \emptyset\).

ER5. Return \(\text{finSet}' \cup \text{finSet}''\).

Extended Semantics and Correctness of \(W\)-Process

Let \(\alpha\) and \(\beta\) are precondition and postcondition for a \(W\)-process \(P\). Let \(B(\alpha) = \{b \in B\} \subseteq \Psi(\alpha)\) (\(\tau = 1\)), \(B(\beta) = \{b \in B\} \subseteq \Psi(\beta)\) (\(\tau = 1\)). The following assertion holds.

\textbf{Theorem 1}

\(P\) is correct with respect to \(\alpha\) and \(\beta\), if \(\alpha \Rightarrow \Psi(P(\alpha)) \subseteq B(\beta)\).

\textbf{APPROXIMATIVE CALCULATIONS VIA MOLDS}

\textbf{Definitions}

Let \(\Sigma\) be a predefined set of \(B\) subsets ordered by the inclusion relation. Suppose that
- \(\emptyset, B \in \Sigma\),
- \(\Sigma\) is close with respect to intersection operations.

Suppose \(s \subseteq B\). Let denote by \(\lceil s \rceil\) the unique minimum element of \(\Sigma\) containing \(s\), and by \(\lfloor s \rfloor\) any minimum element of \(\Sigma\) containing \(s\). Let call a pair \(\text{mold} = \langle s_1, s_2 \rangle\) so that \(s_1, s_2 \in \Sigma\), a mold and denote the set of all molds by \(M\). Let use the designation \(\Lambda\) for mold \(\langle s_1, s_2 \rangle\), if \(s_1 \supseteq s_2\), and identify such a mold as \(\emptyset, \emptyset\).

Let \(A \in \mathcal{N}\). Define an action \(A: M \rightarrow M\) so that \(\langle \text{mold}\rangle A = \langle \pi_1(\text{mold})A, \pi_2(\text{mold})A \rangle\) for all \(\text{mold} \in M\). Similarly, for \(p \in \mathcal{C}\) define an action \(p: M \rightarrow M\).
Let the designation \((\text{mold}) \text{ P}\) for the value \(\Psi(\text{P})(\text{mold})\).

**Approximative Calculations and Correctness of W-Process**

Let \(\alpha\) and \(\beta\) be precondition and postcondition of W-process \(\text{P}\), \(\supp(\alpha) = \{b\, |\, \alpha(b) = 1\}\), \(\supp(\beta) = \{b\, |\, \beta(b) = 1\}\). Let 
\[
\text{mold}(\alpha) = \{\supp(\alpha)\}, \quad \text{mold}(\beta) = \{\supp(\beta)\}.
\]

The following statement about the correctness of \(\text{P}\) with respect to \(\alpha\) and \(\beta\), based on comparison of basic sets, holds.

**Theorem 2**

a) If \(\pi_1(\text{finalMold}) <\pi_2(\text{mold}(\beta))\), then \(\text{P}\) is correct with respect to \(\alpha\) and \(\beta\).

b) If \(\pi_1(\text{finalMold}) <\pi_2(\text{mold}(\beta))\), then \(\text{P}\) is incorrect with respect to \(\alpha\) and \(\beta\).

c) Otherwise, the assertion about correctness of \(\text{P}\) with respect to \(\alpha\) and \(\beta\) is undefined.

**SPLITTING MOLDS**

Let describe a possible determination of basis having increased effectiveness of Boolean operators on the molds. For this purpose a basic set \(\text{se}\Sigma\) has to be defined as a Cartesian product of variable value subsets generating the state of environment, in such way \(\text{se}\Sigma \Rightarrow \text{S} \times \bigtimes \text{v}\), where

- \(\text{V}\) is an ordered set of activity input variables, transition condition input variables and external memory variables. Suppose the variables of \(\text{V}\) are ordered in some predefined order.

- \(\text{se}\Sigma \subseteq \text{DOM}(\text{v})\).

Notice that \(\text{S} = \emptyset\) if \(s(\text{v}) = \emptyset\) for some \(s\in\text{V}\).

Let call above described subsets of \(\text{B}\) splitting sets and molds constructed on the basis of splitting sets, as splitting molds. Let introduce an ordering relationship for the set of splitting subsets as: \(\text{s}\subseteq\text{s'}\), if \(s(\text{v}) \subseteq s'(\text{v})\) for all \(s(\text{v}) \in \text{V}\). Let \(s'\) and \(s''\) are splitting subsets, \(\text{v}\in\text{V}\). Let consider following operations:

- \(s' \oplus s'' = \bigtimes \in v \in V (s'(v) \cup s''(v))\).
- \(s' \otimes s'' = \bigtimes \in v \in V (s'(v) \cap s''(v))\).
- \(s' \cup_p s'' = \bigtimes \in v \in V \{s'(v) \cup s''(v)\} \), where \(s(\text{v}) = \{s'(v), v \neq p\} \cup \{s''(v), v = p\}\).
- \(s' \setminus_p s'' = \bigtimes \in v \in V \{s'(v), v \neq p\} \), where \(s(\text{v}) = \{s'(v), v = p\} \cup \{s''(v), v \neq p\}\).

**Theorem 3** Let \(s'\) and \(s''\) are splitting sets,

\[
\text{s'} = \bigtimes \in v \in V (s'(v)), \quad \text{s''} = \bigtimes \in v \in V (s''(v)).
\]

The following assertions hold:

a) \([s' \cap s''] = \bigtimes \in v \in V (s' \cap s'')\).

b) \([s' \cup s'']\) could be defined as
• \( s \cup \neg s \), if \((\exists p \in \mathcal{V}) [s’(p)\neg s’(p)\neq\emptyset, s’(p)\neg s’(p)\neq\emptyset].\)

• \( s \), if \( s \subseteq s’ \), or \((\exists p \in \mathcal{V})|s’(p)\subseteq s’(p)\wedge (\forall v \in \mathcal{V}, \forall p |s’(v)\subseteq s’(v)]\).

• \( s \), if \( s \subseteq s’ \), or \((\exists p \in \mathcal{V})|s’(p)\subseteq s’(p)\wedge (\forall v \in \mathcal{V}, \forall p |s’(v)\subseteq s’(v)]\).

Applying the algorithm to the given process a result will be that the activity 7 cannot be activated and the value of the variable bResolved for the activity 7 and memory will remain “Unknown”. As a result, the condition for incorrect processes will be satisfied (See Theorem 2). This sample illustrates a very simple model of the discussed flow with some shortening so the incorrect design (2 mutually exclusive execution paths with one synchronization point) can be easily found. The control connector between activities 4 and 7 has to be removed to correct the modified template logic. The postcondition has to be also changed to the PostCon = bCanceled OR (bWorkaroundOrFixProvided AND bIncidentResolutionPlanCreated) AND bResolved.

At present, the works on the tool for applying the algorithm to all library processes are under development.
CONCLUSION

A partial recognizing algorithm for the formal verification of acyclic processes, which underlies IBM/Workflow model, is suggested. The algorithm uses the molds for upper and lower approximation of environment possible state sets. The consideration of mold transformations on all branching paths allows to avoid the process execution and examination per each input. The algorithm instead examines all branching paths through the molds. The weakness of the algorithm is the possibility of getting uncertain answer to the question regarding the program correctness. Additional investigations on specific cases should be performed to estimate the probability of such a result.

The presented approach is now used as a basis for the tool development. The tool is suggested to verify the processes for IMB PRM-IT library.

Works on analyzing the algorithm effectiveness through the applying the algorithm to some popular libraries and building more accurate approximations are in progress.

REFERENCES


ACKNOWLEDGEMENTS

The authors express their acknowledgement to Prof. Samvel Shoukourian who inspired them to perform investigations in the area of workflow process formal verification. Authors also appreciate the International Scientific Technology Center- ISTC A1451 Project for supporting this research.

BIOGRAPHY

ARMEN KOSTANYAN is an associate professor of Yerevan State University. He is also a visiting associate professor of American University of Armenia. His main research interests are in the computation system analysis and design area. He participates in the “Development of Scientific Computing Grid on the Base of Armcluster for South Caucasus Region” and “Problems of Analysis and Optimization of Processes in Distributed Environments” research programs.

ANNA VAROSYAN is a lecturer of Yerevan State University. She is also a design engineer in Virage Logic International Corporation. Her research interests include verification and validation of languages and software tools. She participates in the “Development of Scientific Computing Grid on the Base of Armcluster for South Caucasus Region” research program.