An Approach to Compose Viewpoints of Different Stakeholders in the Specification of Probabilistic Systems

Mahboubeh Samadi*
Faculty of Electrical and Computer Engineering, Shahid Beheshti University G. C. Tehran, Iran
mbh_samadi@yahoo.com
Hasan Haghighi
Faculty of Electrical and Computer Engineering, Shahid Beheshti University G. C. Tehran, Iran
h_haghighi@sbu.ac.ir

Received: 19/May/2013 Accepted: 13/Jan/2014

Abstract
Developing large and complex systems often involves many stakeholders each of which has her own expectations from the system; hence, it is difficult to write a single formal specification of the system considering all of stakeholders' requirements at once; instead, each stakeholder can specify the system from her own viewpoint first. Then, the resulting specifications can be composed to prepare the final specification. Much work has been done so far for the specification of non-probabilistic systems regarding viewpoints (or expectations) of different stakeholders; however, because of big trend to apply formal methods on probabilistic systems, in this paper, we present an approach to compose viewpoints of different stakeholders in the specification of probabilistic systems. According to this approach, different viewpoints are separately specified using the Z notation. Then, the resulting specifications are composed using some new operators proposed in this paper. We show the applicability of the presented approach by performing it on a known case study.

Keywords: Formal Methods, Formal Specification, Probabilistic Systems, Partial Models, Multiple Viewpoints.

1. Introduction

Developing large and complex systems often involves many stakeholders. However, each of these stakeholders has her own viewpoints (or expectations) when the system is specified. To collect requirements of all stakeholders, [1,2] proposed the viewpoint-oriented requirements engineering. From another point of view, modern systems are big and complex, resulting from assembling multiple components. "Components are designed by teams, working independently but with a common agreement on what the interfaces of each component should be [3]".

Considering both of the above mentioned cases, parallel and logical compositions should be done to produce a final specification of the system. "System specification through parallel composition is done by putting specifications of various components together. Logical composition (or merging), however, is used to merge viewpoints of different stakeholders to obtain the specification of a single component or system [4]".

As our review in section 2 shows, much of the related work has focused on parallel composition of partial specifications of non-probabilistic systems. Since there is a big trend to the formal specification and development of probabilistic systems, we present a formal method to compose viewpoints of different stakeholders when specifying a single probabilistic component or system (i.e., logical composition).

To achieve this goal, we first use the Z notation to specify viewpoints of different stakeholders separately. As shown in the paper, Z schema calculus operations do not work on merging resulting specifications; hence, we define a new set of operators, including "m_conjunction", "m_disjunction" and "m_hiding" to compose specifications obtained after the first step. Names of operators begin with "m" which abbreviates for "merge".

Section 2 reviews the related work. In section 3 a probabilistic system is specified from viewpoints of different stakeholders. Section 4 first shows that Z schema calculus operations are not sufficient to compose specifications of different stakeholders and then presents a set of new operators. The applicability of our method is demonstrated using a known case study in section 5. And finally, section 6 is devoted to the conclusion and directions for future work.

2. Related Work

We categorize the works on the system specification from viewpoints of different stakeholders into two groups: specification of non-probabilistic systems and specification of probabilistic systems. There is little work on the latter category. Moreover, the presented methods are based on behavioural models.

Instead, much work has been done so far to specify non-probabilistic systems from viewpoints of different stakeholders. In [5], it is shown that partial models can be used to specify a system from viewpoints of different stakeholders. In this way, each model satisfies certain system requirements (from a certain stakeholder's point

* Corresponding Author

of view). To make the final specification of the system, these models should be elaborated through both logical and parallel compositions in order to yield a system model that preserves the properties of the initial viewpoints altogether. In [8] and [9], a new operator, called conjunction, is introduced for the composition of different stakeholders' specifications. The behaviour of this operator is similar to the merge operator introduced in [5], but it is only defined for Modal Transition Systems (MTSs).

In [12] the modal interface framework, a unification of interface automata and modal specification is presented. The goal of this work is to compose specifications of interfaces from different viewpoints. The result of this work is a complete theory with a powerful composition algebra that includes operations such as conjunction (for requirements composition) and residuation (for components reuse that in addition assumes/guarantees contract-based reasoning [11]).

In [13], viewpoints are shown as partial specifications of functionality, written in Z but by different people, to be reconciled later. The focus of this work is on reconciliation and amalgamation of partial specifications and not the structure of these specifications themselves. By reconciliation, partial specifications become ready for the composition, and by amalgamation, real composition is done.

For collections of partial specifications to be meaningful, consistency between them has to be committed. In [15], it is described how to check consistency between partial specifications in Z, and how to ensure that different partial specifications of one system do not impose contradictory requirements; in [10] a solution to handle inconsistency between different specifications is introduced.

Besides the above mentioned work on non-probabilistic systems, a number of works have been done in the area of multi viewpoints specification of probabilistic systems. Interval Markov Chains (IMCs) and Constraint Markov Chains (CMCs) introduced in [16,17] can be used for non-functional analysis of multi viewpoints probabilistic systems [4]. In [4], it is shown that IMC is not a proper formalism for compositional specification. Thus, [4] and [17] introduce CMC for component based design of probabilistic systems. CMCs are a further extension of IMCs allowing rich constraints on the next-state probabilities from any state.

Larsen et. al. [18] further explore the influence of non-deterministic behaviour by mixing CMCs and MTSs and considering Probabilistic Automata (PA). In their model, state changes are additionally guarded by actions [4]. They present a specification theory for PAs, namely Abstract Probabilistic Automata (APA) which can serve as a specification theory for systems with both non-deterministic and stochastic behaviours. APA like any usable specification theory is equipped with a conjunction operator that allows combining multiple requirements into a single specification, and a composition operator that allows specifications to be combined structurally [17].

As described, there is not much work in the specification of probabilistic systems from viewpoints of different stakeholders. Moreover, the existing works on probabilistic systems use behavioural models to specify such systems while benefits of using well-known functional specification languages, such as Z, encourage us to specify probabilistic systems from viewpoints of different stakeholders using a Z-based formalism.

3. Specification of probabilistic systems from viewpoints of different stakeholders

In this section, we use the Z notation to write separate specifications (of a probabilistic system) describing viewpoints of different stakeholders. We propose our specification method through an illustrative example [17]. In this example, a customer and a manufacturer are considered as stakeholders of the system.

3.1 Example

Two parties, a customer and a manufacturer, are discussing a design of a relay for an optical telecommunication network. The relay should have several modes of operation, modelled by four dynamically changing properties and specified by atomic propositions a, b, c, and d as follows:

a: The Bit error rate is less than 1 per billion bits transmitted.

b: The Bit rate is higher than 10 Gbits/s.

c: Power consumption is less than 10 W.

d: The relay is not in the transmission mode (is in the standby state).

At first, informal specifications of the relay from customer's and manufacturer's viewpoints are presented.

- *Customer Specification:* In the initial state, the relay is in the standby mode (i.e., proposition "d" holds). Then, with a probability more than 0.7, it can move to state s_2 which is specified as {{a, b}, {a, c}, {b, c}, {a, b, c}}; this set means that in state s_2, at least two of properties "a", "b" and "c" hold, and "d" does not hold. With an unknown probability, the relay can move from the initial state to state s_3 specified as {{a}, {b}, {c}}; this means that in state s_3, exactly one of properties "a", "b" and "c" holds. The relay comes back to the initial state from states s_2 and s_3 with probability 1. Finally, there is no transition with the same source and destination (Figure 1).
- *Manufacturer Specification:* In the initial state, the relay is in the standby mode. Then with a probability more than 0.2, it can move to state s_3 where "a" and "d" do not hold. And with an unknown probability, it can move from the initial state to state s_2 where at least proposition "a" holds, and "d" does not hold. This relay comes back to

the initial state from states s_2 and s_3 with probability 1. Finally, there is no transition with the same source and destination (Figure 2).

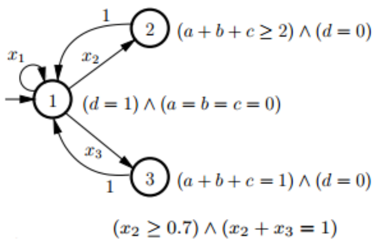

Fig. 1. The Customer's Specification

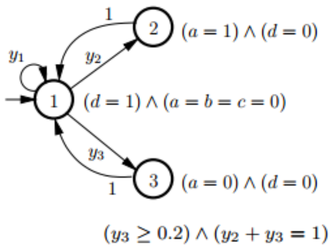

Fig. 2. The Manufacturer's Specification

3.2 Formal specification of the relay

Here is the formal specification of every probabilistic system from one stakeholder's point of view:

```
Statenum:ℕ
─────────────
Statenum≥0
```

```
State:ℙℕ
─────────────
State=1..Statenum
```

```
[Property]
```

```
StakeholderSpec
Transition:seq State × seq State →0..10^d
PropertyFunc: seq State →ℙℙproperty
InitialState: seq State
States: seq seq State
─────────────
InitialState≠<>
InitialState ∈ States
dom Transition = ran States × ran States
dom PropertyFunc = ran States
```

Fig. 3. The Stakeholder's Specification

Property shows a given type of properties (such as a, b, c, and d in our example) that could be true in each state. *StakeholderSpec*, as the state schema of the system, specifies all system states and their related properties and transitions from the stakeholder's point of view. It also shows the initial state of the system. Although each stakeholder prefers a set of desired bindings of the state schema, we assume that all of them agree on the initial state (*InitialState*) and the set of states (*States*). Since states in the final (composite) specification are combinations of states specified by each stakeholder, we consider a sequence of numbers (each number corresponds to a state in one stakeholder's specification before combination) per each state, either it is composite or simple; for simple states, i.e., when we are considering the specification from a single stakeholder's viewpoint, this sequence has only one element. As an example of composite states, if the initial state specified by each of customer and manufacturer is <1>, the initial state in the final, composite specification will be shown as <1, 1>.

For two states s_i and s_j, *Transition* (s_i, s_j) is the probability of transition from s_i to s_j. Since floating-point numbers cannot be shown in the Z notation, transition probabilities are converted to natural numbers by multiplying them with 10^d. Thus, transition probabilities are shown as $0..10^d$. Considering all existing probabilities, d is the maximum number of digits to the right of the floating point. *PropertyFunc* is a function that assigns a set of sets of properties to each state; for example, consider set {{a, b}, {a, c}, {b, c}, {a, b, c}} for state s_2 in the customer's specification of the relay. The last two constraints in the schema guarantee that both *Transition* and *PropertyFunc* are defined on all of available states and anything else.

Regarding *StakeholderSpec* above, the formal specification of the relay from viewpoints of the customer and manufacturer is shown as *CustomerSpec* and *ManufacturerSpec* schemas, respectively.

```
CustomerSpec
StakeholderSpec
─────────────
InitialState=<1>
States=<<1>,<2>,<3>>
PropertyFunc(InitialState)= {{d}}
PropertyFunc(< 2 >)={{a,b},{a,c},{b,c},{a,b,c}}
PropertyFunc(< 3>)= {{a},{b},{c}}
Transition(<1>,<2>) ≥7
Transition(<1>,<1>) =0
Transition(<2>,<1>) =10
Transition(<3>,<1>) =10
Transition(<1>,<1>) + Transition(<1>,<2>) + Transition(<1>,<3>) =10
Transition(<2>,<1>) + Transition(<2>,<2>) + Transition(<2>,<3>) =10
Transition(<3>,<1>) + Transition(<3>,<2>) + Transition(<3>,<3>) =10
```

Fig. 4. The Customer's Specification

```
ManufacturerSpec
StakeholderSpec
─────────────
InitialState=<1>
States=<<1>,<2>,<3>>
PropertyFunc(InitialState)= {{d}}
PropertyFunc(< 2 >)={{a },{a,b},{a,c},{a,b,c}}
PropertyFunc(< 3>)= {{b,c},{b},{c}}
Transition(<1>,<3>) ≥2
Transition(<1>,<1>) =0
Transition(<2>,<1>) =10
Transition(<3>,<1>) =10
Transition(<1>,<1>) + Transition(<1>,<2>) + Transition(<1>,<3>) =10
Transition(<2>,<1>) + Transition(<2>,<2>) + Transition(<2>,<3>) =10
Transition(<3>,<1>) + Transition(<3>,<2>) + Transition(<3>,<3>) =10
```

Fig. 5. The Manufacturer's Specification

4. Operators to compose viewpoints

As a simple example, suppose that we use the Z conjunction operator to compose *CustomerSpec* and *ManufacturerSpec*. The constraint part of the resulting schema will be *false* because, just as one example, *PropertyFunc(<2>)* is in the left hand of two equalities with different right hand sides in *CustomerSpec* and *ManufacturerSpec*; the conjunction of these two equalities and thus the conjunction of schemas constraints will be *false*. Such a *false* constraint obviously satisfies neither the customer nor the manufacturer. Related to this example, it should be noticed that at least two of properties "a", "b" and "c" hold in state s_2 from the customer's viewpoint. On the other hand, at least proposition "a" holds in state s_2 from the manufacturer's viewpoint. Thus, we could have a composite state (in the final, composite specification) where "a" holds, and at least one of "b" and "c" holds. Such a state satisfies both the customer and the manufacture; similar examples can be given for the composition of other states and also transitions between states.

In summary, while we could have a composite specification satisfying both the customer and the manufacture, the Z conjunction operator is not able to generate such specification. Similarly, it can be shown that the disjunction operator in Z is not sufficient for merging specifications of different stakeholders in order to meet at least one of the viewpoints. Therefore, we are to define new operators to merge specifications of different stakeholders.

4.1 m_conjunction: meeting both viewpoints

The following definitions, that show how transition probabilities are determined in composite specifications, will be used when introducing the new operator m_conjunction.

Definition 1. For two states $< x_i >$ and $< x_j >$ in the first stakeholder's specification, if $\mathrm{Transition}(< x_i >, < x_j >) \geq a$, then for every states $< y_k >$ and $< y_{k'} >$ in the second stakeholder's specification for which $\mathrm{Transition}(< y_k >, < y_{k'} >) \geq 0$, we have $\sum_{y_k, y_{k'}} \mathrm{Transition}\big(< x_i, y_k >, < x_j, y_{k'} >\big) \geq a$ in the composite specification.

Definition 2. For two states $< y_i >$ and $< y_j >$ in the second stakeholder's specification, if $\mathrm{Transition}(< y_i >, < y_j >) \geq a$, then for every states $< x_k >$ and $< x_{k'} >$ in the first stakeholder's specification for which $\mathrm{Transition}(< x_k >, < x_{k'} >) \geq 0$, we have $\sum_{x_k, x_{k'}} \mathrm{Transition}\big(< x_k, y_i >, < x_{k'}, y_j >\big) \geq a$ in the composite specification.

Definition 3. For two states $< x_i >$ and $< x_j >$ in the first stakeholder's specification and for two states $< y_i >$ and $< y_j >$ in the second stakeholder's specification, if $\mathrm{Transition}\big(< x_i >, < x_j >\big) = k$ and $Transition\big(< y_i >, < y_j >\big) = k'$, then $\mathrm{Transition}\big(< x_i, y_i >, < x_j, y_j >\big) = k * k'$.

The rationality behind Definition 1 is that, suppose based on the first stakeholder's viewpoint, the transition probability from state $< x_i >$ to $< x_j >$ is greater than a. Now, based on the final, composite specification, the system should be able to move from states which start with $< x_i >$ to states which start with $< x_j >$ with a probability greater than a totally. In this way, the first stakeholder is satisfied by the final specification. A same reason can be given for Definition 2. Notice that two more definitions should be considered for ≤ the same as those presented for ≥.

Schema *CompositeSpec* below is used to compose specifications of different stakeholders in the form of *StakeholderSpec* (see subsection 3.2). We assume that we have two schemas of two stakeholders, called *FirstStakeholderSpec* and *SecondStakeholderSpec*. In addition, *StakeholderSpec* given in declaration part of *CompositeSpec* is the final, composite specification.

Since *FirstStakeholderSpec* and *SecondStakeholderSpec* have identifiers with the same name, in the declaration part of *CompositeSpec* their identifiers are renamed to avoid variable capturing: all identifiers are replaced with the same names but ended by F (for identifiers of *FirstStakeholderSpec*) and S (for identifiers of *SecondStakeholderSpec*). We use notation *FirststakeholderSpec[identifier/identifierF]* to show that one "F" is appended to the name of all identifiers of *FirstStakeholderSpec*. A similar notation is used for renaming identifiers of *SecondStakeholderSpec*.

Since the constraint part of *CompositeSpec* is almost long, we present it gradually via different parts. The informal description of each part is also given accordingly. For this reason, we do not draw the bottom line of *CompositeSchema* in Figure 6. As the first line of the constraint part, it is mentioned that the initial state of the system is obtained by concatenating the initial states of two stakeholders' viewpoints.

CompositeSpec

FirstStakeholderSpec[identifier/identifierF]
SecondStakeholderSpec[identifier/identifierS]
StakeholderSpec

InitialState= InitialStateF ⁀ InitialStateS

Fig. 6. Composite Specification – part 1

Besides the given equality for *InitialState*, *CompositeSpec* has the following constraints, too:

$\forall\ spindex_F$: 1 .. #StatesF; $index_F$: 1 .. #StatesF;
$\forall spindex_S$: 1 .. #StatesS; $index_S$: 1 .. #StatesS |
PropertyFuncF(StatesF. $spindex_F$) ∩
PropertyFuncS(StatesS. $spindex_S$) =∅ •
Transition(StatesF. $index_F$ ⁀ StatesS. $index_S$,
StatesF. $spindex_F$ ⁀ StatesS. $spindex_S$)=0
∧ *Transition*(StatesF. $spindex_F$ ⁀StatesS. $spindex_S$,
StatesF. $index_F$ ⁀StatesS. $index_S$)=0

Fig. 7. Composite Specification – part 2

Before probing on the above predicate, we should mention that in the specification which supports viewpoints of both stakeholders, the set of states is as the Cartesian product of states in the specification of each stakeholder. More precisely, if sets of states in two stakeholder's specifications are $1..k_1$ and $1..k_2$, the set of states in the final specification is $(1..k_1)\times(1..k_2)$. Of course, the composition of two states will lead to an inconsistent state if the conjunction of their related properties is false, or in other words, the intersection of sets resulted from applying *PropertyFunc* to those two states is $\varnothing$. Inconsistent states will not appear in the final specification.

The predicate in Figure 7 says that probabilities of all transitions from\to an inconsistent state are 0. In other words, the system cannot move from\to inconsistent states; figures 8 and 9, in contrast, specify transition probabilities for consistent states. Based on the predicate given in Figure 8, if the transition probability from state s_1 to state s_2 in the specification of one stakeholder is in interval $[p_1,p_2]$, the total sum of transition probabilities from those states corresponding to s_1 to those states corresponding to s_2 in the final specification should be in interval $[p_1,p_2]$. This predicate is based on Definitions 1 and 2.

As another point, the utility function "sum" defined using an axiomatic definition in Figure 11 calculates the sum of elements in a sequence of natural numbers.

$$
\begin{array}{l}
\forall\, spindex_F: 1 \,..\, \#StatesF;\ index_F: 1 \,..\, \#StatesF;\\
\forall\, spindex_S: 1 \,..\, \#StatesS;\ p_1,p_2{:}\mathbb{N} \mid\\
ProperyFuncF(StatesF.\,spindex_F) \cap\\
PropertyFuncS(StatesS.\,spindex_S) \neq \varnothing \ \wedge\\
p_1 \leq TransitionF\ (StatesF.\,spindex_F, StatesF.\,index_F) \wedge\\
TransitionF\ (StatesF.\,spindex_F, StatesF.\,index_F) \leq p_2 \ \bullet\\
\exists\, s{:}seq\mathbb{N}\ ;\ \forall\, i{:}1 \,..\, \#StateS \bullet\\
s.i = Transition(StatesF.\,spindex_F \frown StatesS.\,spindex_S,\\
StatesF.\,index_F \frown StatesS.i) \wedge\ p_1 \leq sum(s) \wedge sum(s) \leq p_2\\
\forall\, spindex_F: 1 \,..\, \#StatesF;\\
\forall spindex_S: 1 \,..\, \#StatesS;\ index_S: 1 \,..\, \#StatesS\ ;\ p_1,p_2{:}\mathbb{N} \mid\\
ProperyFuncS(StatesS.\,spindex_S\,) \cap\\
PropertyFuncF(StatesF.\,spindex_F) \neq \varnothing \wedge\\
p_1 \leq TransitionS\ (StatesS.\,spindex_S,\ StatesS.\,index_S) \wedge\\
TransitionS\ (StatesS.\,spindex_S,\ StatesS.\,index_S) \leq p_2 \ \bullet\\
\exists\, s{:}seq\mathbb{N}\ ;\ \ \forall\, i{:}1 \,..\, \#StatesF \bullet\\
s.i = Transition(StatesF.\,spindex_F \frown StatesS.\,index_S,\\
StatesF.i \frown StatesS.\,index_S) \wedge\ p_1 \leq sum(s) \wedge sum(s) \leq p_2
\end{array}
$$

Fig. 8. Composite Specification – part 3

Unlike predicates in Figure 8 which consider transitions with variable probabilities, the following predicates regard transitions with constant probabilities (see Definition 3):

$$
\begin{array}{l}
\forall\, spindex_F{:}1 \,..\, \#StatesF;\ index_F{:}\ 1 \,..\, \#StatesF;\\
\forall spindex_S{:}\ 1 \,..\, \#StatesS;\ index_S{:}\ 1 \,..\, \#StatesS;\ p_1,p_2{:}\mathbb{N} \mid\\
PropertFuncF(StatesF.\,spindex_F) \frown\\
PropertyFuncS(StatesS.\,spindex_S) \neq \varnothing \wedge\\
TransitionF\ (StatesF.\,spindex_F,\ StatesF.\,index_F) = p_1 \wedge\\
TransitionS\ (StatesS.\,spindex_S, StatesS.\,index_S) = p_2 \ \bullet\\
Transition(StatesF.spindex_F \frown StatesS.\,spindex_S,\\
StatesF.\,index_F \frown StatesS.\,index_S) = p_1 \times p_2
\end{array}
$$

Fig. 9. Composite Specification – part 4

In the following predicate, it is emphasized that the sum of probabilities of transitions from one state in the final specification should be 1 (or in fact 10^d since we multiplied all probabilities with 10^d); the utility function "bind" defined using an axiomatic definition in Figure 11 constructs a sequence through the concatenation of all sequences existing in a sequence of sequences.

$$
\begin{array}{l}
\exists\, s{:}seq\mathbb{N}\,;\ \forall\, spindex_F{:}1 \,..\, \#StatesF;\ i{:}\ 1 \,..\, \#StatesF;\\
\forall spindex_S{:}\ 1 \,..\, \#StatesS; j{:}\ 1 \,..\, \#StatesS \bullet\\
(s.i).j = Transition(StatesF.\,spindex_F \frown StatesS.\,spindex_S,\\
StatesF.\,i \frown StatesS.j) \wedge\ sum(bind(s)) = 10^d
\end{array}
$$

Fig. 10. Composite Specification – part 5

$$
\begin{array}{|l}
sum{:}\ seq\mathbb{N} \rightarrow \mathbb{N}\\
\hline
sum\langle\rangle = 0\\
\forall\, s{:}seq\mathbb{N} \mid s \neq \langle\rangle \bullet sum(s) = head(s) + sum(tail(s))
\end{array}
$$

$$
\begin{array}{|l}
bind{:}\ seqseq\mathbb{N} \rightarrow seq\mathbb{N}\\
\hline
bind\langle\rangle = \langle\rangle\\
\forall s{:}seqseq\mathbb{N} \mid s \neq \langle\rangle \bullet bind(s) = head(s) \frown bind(tail(s))
\end{array}
$$

Fig. 11. Utility Functions

The following predicate says that state properties in the final specification are the conjunction of state properties specified by each stakeholder.

$$
\begin{array}{l}
\forall\, si{:}\ 1..\#StatesF; sj{:}\ 1\,..\#StatesS \bullet PropertyFunc(StatesF.si \frown StatesS.sj) =\\
PropertyFuncF(StatesF.\,si) \cap PropertyFuncS(StatesS.sj)
\end{array}
$$

Fig. 12. Composite Specification – the last part

Now, since the final specification should only consist of composed states and related transitions, specifications of the first and the second stakeholders should be hidden. Consequently, the new operator for merging viewpoints of two stakeholders (in order to satisfy both of them) is specified as follows:

$$
\begin{array}{l}
FirstStakeholderSpec\ \wedge_{VoS}\ SecondStakeholderSpec \mathrel{==}\\
\exists\, FirstStakeholderSpec[identifier/identifierF],\\
SecondStakeholderSpec[identifier/identifierS] \bullet CompositeSpec
\end{array}
$$

Fig. 13. m_conjunction of Two Schemas

Here, $\wedge_{VoS}$ is the symbol of m_conjunction in which VoS abbreviates for "Viewpoints of Stakeholders".

4.2 m_disjunction: meeting at least one of viewpoints

Sometimes, it is important that the final specification satisfies the concerns of at least one stakeholder. We introduce operator “m_disjsunction” to support this requirement. In the final specification, the states are all of the states specified by all stakeholders. Since states in two specifications may include identical numbers, and we are going to consider all states in the final specification, schema *Rename* in Figure 14 specifies the change of numbers used in states of the second specification. This change should be done before we merge the two specifications.

Rename

$\mathit{FirstStakeholderSpec}[\mathit{identifier}/\mathit{identifierF}]$

$\Delta \mathit{SeconStakeholderSpec}[\mathit{identifier}/\mathit{identifierS}]$

$\mathit{StatesS}' = \mathit{Add2ToAllElems}\ (\mathit{StatesS})$

$\mathit{TransitionS}' = \mathit{Add2ToAllTStates}\ (\mathit{TransitionS})$

$\mathit{PropertyFuncS}' = \mathit{Add2ToAllPStates}\ (\mathit{PropertyFuncS})$

$$\mathit{FinalCompositeSpec} = \mathit{Rename} \mathbin{⨾} \mathit{LeastCompositeSpec}$$

Fig. 14. Change of Numbers in the Second Specification

Three functions *Add2ToAllElems*, *Add2ToAllTStates* and *Add2ToAllPStates* are supposed to add number “2” at the beginning of numbers used in the states of the second specification. For example, state <1> is changed to <21>. Due to the space limitation, we do not define these functions here. The final specification is obtained by sequential composition of *Rename* and *LeastCompositeSpec* specified below.

Schema *LeastCompositeSpec* specifies the composition of stakeholder specifications using m_disjunction. This schema includes the schemas of two stakeholders. Similar to what we did for m_conjunction, we rename identifiers of *FirstStakeholderSpec* and *SecondStakeholderSpec* to avoid variable capturing here.

LeastCompositeSpec

$\mathit{FirstStakeholderSpec}[\mathit{identifier}/\mathit{identifierF}]$

$\mathit{SecondStakeholderSpec}[\mathit{identifier}/\mathit{identifierS}]$

$\mathit{StakeholderSpec}$

$\mathit{InitialState} = \langle 0 \rangle$

$\mathit{States} = \langle \mathit{InitialState} \rangle \frown \mathit{StatesF} \frown \mathit{StatesS}$

$\mathit{Transition} = \mathit{TransitionF} \cup \mathit{TransitionS} \cup$

$\{((\mathit{InitialState},\ \mathit{InitialStateF}),\ 0.5 \times 10^{d})\} \cup$

$\{((\mathit{InitialState},\ \mathit{InitialStateS}),\ 0.5 \times 10^{d})\}$

$\mathit{PropertyFunc} = \mathit{PropertyFuncF} \cup \mathit{PropertyFuncS} \cup \{(\mathit{InitialState},\ \emptyset)\}$

Fig. 15. *LeastCompositeSpec* Schema

We define new state <0> as the initial state. This state has no property (see the last line of the constraint part). Since the states in the composite system should be all of the states specified by both stakeholders, states are specified as the concatenation of states specified by the first and the second stakeholder and also the new defined state (i.e., <0>). To consider the two viewpoints in the same way, we add two new transitions with probability 0.5 from the new initial state: one to the initial state in *FirstStakeholderSpec* and the other one to the initial state in *SecondStakeholderSpec*.

Finally, the new operator for composing viewpoints of two stakeholders (in order to meet at least one of them) is specified as follows ($\vee_{VoS}$ is the symbol of m_disjunction):

$\mathit{FirstStakeholderSpec} \vee_{VoS} \mathit{SecondStakeholderSpec} ==$

$\exists\, \mathit{FirstStakeholderSpec}[\mathit{identifier}/\mathit{identifierF}]$

$,\ \mathit{SecondStakeholderSpec}[\mathit{identifier}/\mathit{identifierS}]$

$\bullet\ \mathit{FinalCompositeSpec}$

Fig. 16. m_disjunction of Two Schemas

4.3 m_hiding: hiding states

Sometimes it is required to hide one state before using the system specification from one stakeholder’s viewpoint. Here are some examples:

- A stakeholder would not rather see one state of the system that another stakeholder specifies.
- To apply some change to a given specification in order to make it reusable in another situation.
- Each stakeholder may change her specification according to her new viewpoint only by hiding states.
- Before composing specifications using the m_conjunction operator, it may be necessary to change one or both specifications by hiding states.

Hide schema is as follows:

Hide

$\Delta\, \mathit{StakeholderSpec}$

$\mathit{HiddenState}: \mathit{StateSet}$

$\mathit{HiddenState} \neq \mathit{InitialState}$

$\exists\, x: 1..\#\mathit{States} . \mathit{Transition}(\mathit{HiddenState}, \mathit{States}.x) > 0$

$\mathit{HiddenState} \neq \langle\rangle$

$\forall\, x: 1..\#\mathit{States} \mid \mathit{States}.x \neq \mathit{HiddenState} . \mathit{States}.x \in \mathrm{ran}\ \mathit{States}'$

$\#\mathit{States}' = \#\mathit{States} - 1$

Fig. 17. *Hide* Schema - part 1

In the declaration part of this schema, the state being hidden is specified as *HiddenState*. In the constraint part, it is mentioned that the hidden state should not be the initial state of the specification. In addition, it should be the source of at least one transition. These two constraints are given to guarantee that no probability value is missed after removing the hidden state. The last two predicates describe removing the hidden state.

$\forall\, x: 1..\#\mathit{States} . \mathit{Transition}'(\mathit{States}.x, \mathit{HiddenState}) = 0$

$\forall\, x: 1..\#\mathit{States} . \mathit{Transition}'(\mathit{HiddenState}, \mathit{States}.x) = 0$

Fig. 18. *Hide* Schema – part 2

The above predicate says that every transition whose source or destination is the hidden state should be discarded. Instead, for arbitrary states x and y, if there is one transition from x to the hidden state and one transition from the hidden state to y, the multiplication of probabilities of these two transitions should be added to the current probability of the transition from x to y. This constraint is described in Figure 19.

$$
\begin{aligned}
&\forall x,y: 1 \,..\, \#States;\ p_1,p'_1,p_2,p'_2 : \mathbb{N} \bullet \\
&p_1 \leq Transition(States.\ x,\ HiddenState)\ \wedge \\
&Transition(States.\ x,\ HiddenState) \leq p_2\ \wedge \\
&p'_1 \leq Transition(HiddenState,\ States.y)\ \wedge \\
&Transition(HiddenState,\ States.y\ ) \leq p'_2 \longrightarrow \\
&p_1 \times p'_1 + Transition(States.x,\ States.y) \leq Transition'(\ States.\ x,\ States.y) \\
&\wedge\ Transition'(\ States.\ x,\ States.y) \leq p_2 \times p'_2 + Transition\ (States.x,\ States.y)
\end{aligned}
$$

Fig. 19. *Hide* Schema – the last part

At last, the new operator for hiding a state from one specification is specified as follows:

$$StakeholderSpec \setminus_{VoS} HiddenState \mathrel{\hat{=}} \exists\ Stakeholder \bullet Hide \setminus HiddenState$$

Fig. 20. m_hiding Definition

Here, $\setminus_{VoS}$ is the symbol of m_hiding.

5. Case Study

In subsection 3.2, a relay for an optical telecommunication network was specified from viewpoints of different stakeholders, i.e., a customer and a manufacturer. The constraint part of *CustomerSpec* $\wedge_{VoS}$ *ManufacturerSpec* below specifies states properties and transitions probabilities in the relay from both stakeholders' viewpoints. Properties of each state are conjunction of properties specified by each stakeholder. It is worth noting that inconsistent states and their relevant transitions have not been shown in the schema.

```
__CustomerSpec ∧VoS ManufacturerSpec__________
StakeholderSpec
______________________________
InitialState=<1,1>
States=<<1,1>,<2,2>,<3,3>,<2,3>,<3,2>,<1,2>,<1,3>>
PropertyFunc(<1,1>)={{d}}
PropertyFunc(<2,2>)={{a,c},{a,c},{a,b,c}}
PropertyFunc(<2,3>)={{b,c}}
PropertyFunc(<3,2>)={{a}}
PropertyFunc(<3,3>)={{c},{b}}
Transition(<1,1>,<2,2>)+ Transition(<1,1>,<2,3>)>7
Transition(<1,1>,<2,3>)+ Transition(<1,1>,<3,3>)>2
Transition(<1,1>,<1,1>)+ Transition(<1,1>,<1,2>)+ Transition(<1,1>,<1,3>)=0
Transition(<1,1>,<1,1>)+ Transition(<1,1>,<2,1>)+ Transition(<1,1>,<3,1>)=0
Transition(<2,2>,<1,1>)=10
Transition(<2,3>,<1,1>)=10
Transition(<3,2>,<1,1>)=10
Transition(<3,3>,<1,1>)=10
Transition(<1,1>,<2,2>)+ Transition(<1,1>,<2,3>)+
Transition(<1,1>,<3,3>)+ Transition(<1,1>,<3,2>)=10
______________________________
```

Fig. 21. *CustomerSpec* $\wedge_{VoS}$ *ManufacturerSpec*

Sometimes, it is required that the concerns of at least one stakeholder are satisfied. To achieve this goal, m_disjunction of schemas is useful. Figure 22 indicates the application of m_disjunction to *CustomerSpec* and *ManufacturerSpec*.

```
__CustomerSpec ∨VoS ManufacturerSpec_____
StakeholderSpec
______________________________
InitialState=<0>
States=<<0>,<1>,<2>,<3>,<21>,<22>,<23>>
PropertyFunc<0>={}
PropertyFunc<1>={{d}}
PropertyFunc<2>={{a,b},{a,c},{b,c},{a,b,c}}
PropertyFunc<3>={{a},{b},{c}}
PropertyFunc<21>={{d}}
PropertyFunc<22>={{a},{a,b},{a,c}{a,b,c}}
PropertyFunc<23>={{b,c},{b},{c}}
Transion(<0>,<1>)=5
Transion(<0>,<21>)=5
Transition(<1>,<2>) ≥7
Transion(<1>,<1>)=0
Transion(<2>,<1>)=10
Transion(<3>,<1>)=10
Transition(<1>,<1>)+Transition(<1>,<2>) +Transition(<1>,<3>)=10
Transition(<2>,<1>)+Transition(<2>,<2>) +Transition(<2>,<3>)=10
Transition(<3>,<1>)+Transition(<3>,<2>) +Transition(<3>,<3>)=10
Transition(<21>,<23>)⩾2
Transition(<21>,<21>)=0
Transition(<22>,<21>)=10
Transition(<21>,<21>)+Transition(<21>,<22>) +Transition(<21>,<23>)=10
Transition(<22>,<21>)+Transition(<22>,<22>) +Transition(<22>,<23>)=10
Transition(<23>,<21>)+Transition(<23>,<22>) +Transition(<23>,<23>)=10
______________________________
```

Fig. 22. *CustomerSpec* $\vee_{VoS}$ *ManufacturerSpec*

Besides the new initial state, i.e., <0>, the final specification consists of all states in the initial specifications. Also, regardless of the new transitions starting from the new initial state, transition probabilities remain unchanged. The resulting specified relay can behave like customer's specification or manufacturer's specification.

Sometimes, it is required to hide one state before using the system specification from one stakeholder's viewpoint. Figure 23 indicates the application of m_hiding to *CustomerSpec* to hide state <2>.

```
_CustomerSpecAfterHiding__________________
Stakeholderspec
______________________________
InitialState=<1>
PropertyFunc<1>={{d}}
PropertyFunc<3>={{a},{b},{c}}
Transition(<1>,<1>) ≥7
Transition(<3>,<1>) =10
Transition(<1>,<1>) + Transition(<1>,<3>) =10
Transition(<3>,<1>) + Transition(<3>,<3>) =10
______________________________
```

Fig. 23. Hiding State <2> in *CustomerSpec*

6. Conclusions

In this paper, an approach to compose viewpoints of different stakeholders in the specification of probabilistic systems was presented. The main contribution of this approach is introducing three new schema operators to manipulate specifications written by different stakeholders. In the extended version of this paper, we are going to formally prove that the proposed operators are sound. Also, as another future work, we will present an approach to specify component based probabilistic systems. To achieve this goal, specifications of different stakeholders should be first merged to construct the specification of each component (logical composition), and then specifications of different components should be combined to construct the specification of the whole system (parallel composition).

References

[1] B. Nuseibeh, J. Kramer, A. Finkelstein, "A Framework for Expressing the Relationships between Multiple Views in Requirements Specification," *In IEEE Transaction on Software Engineering*, 1994, vol. 20, no. 1, pp: 760-773.

[2] G. Kotonya, and I. Sommerville, "Requirements Engineering with Viewpoints," *Journal in Software Engineering*, 1996, vol. 11, no. 2, pp: 58-66.

[3] S. Rui-feng, Y. Chao, X. Jie, "Acquire Multi-Viewpoint from Domain," In *3rd International Conf. on Advance Comp. Theory and engineering*, 2010, vol. 4, pp.598-602.

[4] B. Caillaud, B. Delahaye, K.G. Larsen, A. Legay, M. Pederson, A. Wasowski, "Compositional Design Methodology with Constraint Markov Chains," *In 7th International Conference on the Quantitative Evaluation of Systems*, 2010, pp 123-132.

[5] A. Benveniste, B. Caillaud, R. Passerone, "Multi-Viewpoint State Machines for Rich Component Models," In *Model-Based Design of Heterogeneous Embedded Systems*, Pieter Mosterman, Gabriela Nicolescu (eds.), 2009.

[6] S. Uchitel, and M. Chechik, "Merging Partial Behavioural Models," In *12th ACM International symp. on foundation of soft. Engineering*, 2004, vol. 29, no. 6, pp. 43-52.

[7] G. Brunet, M. Chechik, S. Uchitel, "Properties of Behavioural Model Merging," In *14th International Conference on Formal Methods*, 2006, pp. 98-114.

[8] K.G. Larsen, B. Steffen, C. Weise, "A Constraint Oriented Proof Methodology Based on Modal Transition Systems," In *first International workshop on Tools and Algorithms for the Construction and Analysis of Systems*, 1995, pp. 17-40.

[9] M. Huth, R. Jagadeesan, and D. Schmidt, "A Domain Equation for Refinement of Partial Systems," In *Mathematical Structures in Computer Science*, 2004, vol. 4, no. 3.

[10] C, Anthony, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh, "Inconsistency Handling in Multiperspective Specifications," In *IEEE Transaction on Software Engineering*, 1994, vol 20.no X A1 Gl ST.

[11] D. Fischbein, G. Brunet, N. D'Ippolito, M. Chechik, and S. Uchitel, "Weak Alphabet Merging of Partial Behaviour Models," In *ACM Transactions on Software Engineering and Methodology*, 2011, vol. 21, pp: 1-49.

[12] A. Benveniste1, B. Caillaud1, and A. Ferrari, Multiple Viewpoint Contract-Based Specification and Design. In *FMCO*, Springer, 2008, , pp: 200–225.

[13] JB. Raclet, A. Benveniste, A. Legay, and et al, "A Modal Interface Theory for Component-based Design," In *Fundamenta Informaticae*, 2010.

[14] D. Jackson, "Structuring Z Specifications with Views," In *ACM Transactions on Software Engineering and Methodology*, 1995, vol. 4.

[15] M. Ainsworth, AH. Cruickshank, L.J. Groves and P.J.L. Wallis, "Formal Specification via Viewpoints," In *Proc. 13th New Zealand Computer Conference*, 1993.

[16] E. Boiten, J. Derrick, H. Bowman, M. Steen, "Constructive Consistency Checking for Partial Specification in Z," In *Science of Comp. Prog*., 1995, vol. 35, pp: 29-75.

[17] B. Delahaye, K.G. Larsen, A. Legay, M. Pedersen, A. Wasowski, "Decision Problems for Interval Markov chains," In *5th International Conference on Language and Automata Theory and Applications*, 2011, pp. 274-285.

[18] B. Caillaud, B. Delahaye, K.G. Larsen, A. Legay, M. Pederson, A. Wasowski, "Constraint Markov Chains," *Journal In Theoretical Computer Science*, 2011, vol. 412, no. 5, pp.4373-4404.

[19] B. llaud, B. Delahaye, K.G. Larsen, A. Legay, M. Pederson, A. Wasowski, "New Results on Abstract Probabilistic Automata," In *11th International Conference on Application of concurrency to system design*, 2011, pp. 118-127.

Mahboubeh Samadi received her B.Sc. and M.Sc. degrees in software engineering from the Faculty of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran, in 2010 and 2012, respectively. Her research interests include software engineering and formal methods.

Hasan Haghighi received his B.Sc., M.Sc., and Ph.D. degrees from the Computer Engineering Department, Sharif University of Technology, Tehran, Iran, in 2002, 2004, and 2009, respectively. Since 2009, he has been with the Faculty of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran. His research interests include formal methods, software engineering, software architecture and software test.