HIERARCHICAL STOCHASTIC ACTIVITY NETWORKS: FORMAL DEFINITIONS AND BEHAVIOUR

MOHAMMAD ABDOLLAHI AZGOMI AND ALI MOVAGHAR

Department of Computer Engineering,
Sharif University of Technology,
Tehran 11365, Iran

azgomi@mehr.sharif.edu *and* movaghar@sharif.edu

Abstract: *Stochastic activity networks* (SANs) are a powerful and flexible extension of Petri nets. These models can be used for the modelling and analysis of various kinds and different aspects of distributed real-time systems. Similar to other classical extensions of Petri nets, SANs have some limitations for modelling complex and large-scale systems. In order to remove some of these limitations and provide high-level modelling constructs, we have defined a new extension for SANs, called *hierarchical stochastic activity networks* (HSANs). HSAN models provide a construct for composing a hierarchy of SAN submodels that is called *macro activity*. HSANs encapsulate hierarchies and a key benefit of these models is the possibility of automatic selection and usage of techniques for model construction with reduced state spaces by their modelling tools. In this paper, we will present the informal and formal definitions, behaviour and state process of HSANs. We will also introduce methods for the solution of HSAN models by state space analysis and discrete-event simulation techniques.

Keywords: stochastic Petri nets, stochastic activity networks, hierarchical modelling

1. INTRODUCTION

Stochastic activity networks (SANs) [Movaghar and Meyer, 1984] are a stochastic generalization of *Petri nets* [Peterson, 1982]. These models are more powerful and flexible than most other stochastic extensions of Petri nets such as *stochastic Petri nets* (SPNs) [Molloy, 1982] and *generalized stochastic Petri nets* (GSPNs) [Ajmone; Balbo and Conte 1984]. SANs permit the representation of concurrency, timeliness, fault-tolerance and degradable performance in a single model [Movaghar, 1985]. SAN models have been used to evaluate performance and dependability of a wide range of systems and are supported by several powerful modelling tools such as UltraSAN [Sanders et al, 1995] and Möbius [Deavours et al, 2002].

Similar to other classical extensions of Petri nets, SANs have some limitations for modelling complex and large-scale systems. In order to remove some of these restrictions, we have introduced a new extension for SANs called *hierarchical stochastic activity networks* (HSANs) [Abdollahi and Movaghar, 2003]. HSAN models encapsulate hierarchies and a key benefit of these models is the possibility of automatic selection and usage of techniques for model construction with reduced state spaces.

In this paper we will present the informal and formal definitions, behaviour and state process of HSANs. We will also basically introduce methods for state space analysis and simulation of HSAN models and show how the existing methods can be used for these purposes.

The rest of this paper is organized as follows. In Sec. 2, motivations of this work are described. In Sec. 3, the informal definitions and an example of HSAN models are presented. The formal definitions, behaviour and state process of HSANs are presented in Sec. 4. Methods for the transformation, analytic solution and simulation of HSANs are introduced in Sec. 5. Finally, some concluding remarks are mentioned in Sec. 6.

2. MOTIVATIONS

In the original definition of stochastic activity networks [Movaghar and Meyer, 1984], these models have been defined as a flat network of primitives. However, a few techniques have been introduced for constructing hierarchical and composed SAN models in UltraSAN and Möbius modelling tools.

An important state space reduction technique, which is implemented in UltraSAN and Möbius modelling tools, is the *Replicate/Join construct* [Sanders and Meyer, 1991]. This construct provides a hierarchical, tree-like method of combining submodels to form a larger, composed model. As the name may imply, there are two main methods that can be used to combine submodels: *Replicates* and *Joins*. *Replicates* are used to replicate a model any number of times, often having one or more state variables shared among all of the replicas as a means of connection. *Joins* are used to bring together two or more dissimilar models, connecting

them by sharing certain state variables between them [Stillman, 1999].

There is a set of issues with this construct. It is limited to a tree-like structure and an arbitrary symmetric model structure, such as ring or mesh cannot be expressed by this construct.

Replicate/Join construct has been extended and generalized in the work of [Stillman, 1999] on the *graph composition formalism* in the *Möbius modelling framework* [Deavours, 2001]. This formalism does not limit the model hierarchy to a tree-like structure and any arbitrary structure (such as ring or mesh) is possible.

Using the above techniques, a composed model is constructed in a *bottom-up* manner using some operations (such as *Replicate* and *Join*) [Sanders and Meyer, 1991]. While *top-down* model construction, especially, for large models, is more appropriate. On the other hand, the use of these composition formalisms is specific to the modelling tools (UltraSAN or Möbius) and not SANs in general. They have not been defined formally along with the definition of SANs.

Related to the above difficulties, SANs have two other restrictions:

- There is a lack of facilities for constructing complex models incrementally, by starting with abstract components and easily replacing them with detailed and enhanced components. Here, again a top-down paradigm for model construction is needed.
- Since SAN models are flat, using a part of an existing model, as a component for constructing a new one is difficult. Specially, building and sharing a repository of submodels with well-defined interfaces is not easily possible.

On the other hand, programming languages, especially object-oriented languages, have simple and intuitive ways of encapsulating hierarchies. Keeping in mind these ways and to overcome the above difficulties and restrictions of SANs, we have introduced *hierarchical stochastic activity networks* (HSANs). We will introduce these models in the next sections of this paper.

3. DEFINITIONS OF HIERARCHICAL STOCHASTIC ACTIVITY NETWORKS

In this section we will define SAN and HSAN models. The formal definition and behaviour of HSANs will be presented in the next section.

3.1 A New Definition of SANs

There are two definitions for SAN models: The original definition of SANs [Movaghar and Meyer, 1984 or Movaghar, 1985] and a new definition of SANs [Movaghar, 2001]. We will use the latter definition as the base model throughout this paper. Therefore, we will introduce a new definition of SANs in the following paragraphs.

A new definition of SANs is based on a unified view of the system in three settings: *nondeterministic*, *probabilistic*, and *stochastic*. In a nondeterministic setting, nondeterminacy and parallelism are represented in a nondeterministic manner. In a probabilistic setting, nondeterminacy is specified probabilistically but parallelism is treated nondeterministically. In a stochastic setting, both nondeterminacy and parallelism are modelled probabilistically.

The nondeterministic setting of SANs is referred to as *activity networks*, which are nondeterministic models for representing concurrent and reactive systems. Application of this setting is on the analysis of logical aspects or *verification* of concurrent and reactive systems. Disregarding the timing related information of the model and viewing it in a nondeterministic setting accomplish this. The activity network model is then translated into a transition system and verification is done.

For evaluating the operational aspects of systems such as *performance*, *dependability* and *performability*, the stochastic setting of SANs is used. In this setting, both nondeterminacy and parallelism are represented probabilistically [Movaghar, 2001]. Based on the probability distribution of timed activities, Markovian or non-Markovian SAN models will be resulted. Application of the another setting of SANs, namely, probabitistic setting, is on *probabilistic verification*.

As we mentioned before, the nondeterministic setting of SANs is called activity networks. Activity networks have been developed for representing concurrent systems. The *transition* of Petri nets is replaced by a primitive called *activity* in activity networks. There are two types of activities: *instantaneous activities* and *timed activities*. The former describe events, which occur instantaneously, the latter represent processes, which usually take some time to complete. Instantaneous activities represent system activities, which, relative to the performance variable in question, are completed in a negligible amount of time. Instantaneous activities model nondeterminacy while timed activities represent parallelism. Other primitives, which distinguish activity networks from Petri nets, are *gates*. Gates model complex interactions among activities and, thus, increase modelling flexibility.

The original definition of SANs, as appeared in 1984, includes extra primitives, called "cases," for modelling nondeterminacy, which, with the new definition, can equivalently be replaced by some instantaneous activities. A model based on the

original definition of SANs, must be checked for well-behavedness. The well-behavedness check is in general undecidable and is computationally complex for most models. Some solutions have been proposed in the literature to alleviate this problem. However, in a new definition of SANs, no such check will ever be necessary. Because the syntax of the model has been defined free from well-behavedness checking.

For more information about the differences between these two definitions of SANs, please see [Movaghar, 1984 and Movaghar, 2001].

Graphically, activity networks consist of the following elements:

1. *Activities*, which are of two kinds: timed and instantaneous. In a graphical representation, a timed activity is depicted as ▮ and an instantaneous activity is shown as ▏.
2. *Places*, depicted as ◯.
3. *Input gates*, which have a finite set of inputs and one output. An input gate with n inputs is depicted as in Fig. 1(a). To each such input gate is associated a n-ary computable predicate, called the *enabling predicate*, such that e: $N^n \rightarrow \{true, false\}$ and a computable partial function f, called the *input function*, such that f : $N^n \rightarrow N^n$, where f is defined for all values for which the enabling predicate is true and N is the set of natural numbers.
4. *Output gates*, which have a finite set of outputs and one input. Gates are introduced to permit greater flexibility in defining enabling and completion rules. An output gate with n output is depicted as in Fig. 1(b). To each such output gate is associated a computable function g, called *output function*, such that g: $N^n \rightarrow N^n$, where N is the set of natural numbers.

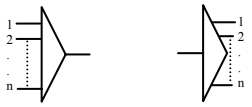


Fig. 1. Graphical representation of gates: (a) Input gate, and (b) Output gate

Structurally, an activity network is an interconnection of a finite number of primitives, subject to the following connection rules [Movaghar, 1985]:

1. Each input of an input gate is connected to a unique place and the output of an input gate is connected to a single activity.
2. Different input gates of an activity are connected to different places.
3. Each output of an output gate is connected to a unique place and the input of an output gate in connected to a single activity.
4. Different output gates of an activity for a case are connected to different places.
5. Each place and activity is connected to some input or output gates.

In order to facilitate the use and to increase the understandability of activity networks, the following conventions are used in their graphical representation:

a) Input gates with similar enabling predicates and input functions are named similarly.

b) Output gates with similar output functions are also named similarly.

c) An input gate with one input, enabling predicate $e(x)$: $x \geq 1$, and input function f such that $f(x) = x-1$, is shown as a directed line from its input to its output.

d) An output gate with one output and output function f such that $g(x) = x+1$, is shown as a directed line from its input to its output.

e) An input gate with one input, enabling predicate $e(x)$: $x = 0$, and identity input function is shown as a directed line from its input to its output crossed by two short parallel lines.

The stochastic setting of SANs is called *stochastic activity networks*. A SAN is formed by adjoining functions C, F, and G, where C specifies the *case probability* assigned to instantaneous activities, F represents the *probability distribution functions* of *activity times* and G describes the sets of *reactivation markings* of timed activities.

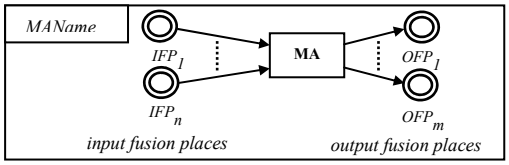
SANs are a generalization of *stochastic Petri nets* (SPNs) [Molloy, 1982] and closely resemble the class of *generalized stochastic Petri nets* (GSPNs) [Ajmone; Balbo and Conte, 1985]. The development of SANs was motivated by the need for a class of network (graphical) models suited to modelling the performability of distributed real-time systems.

3.2 HSAN Models

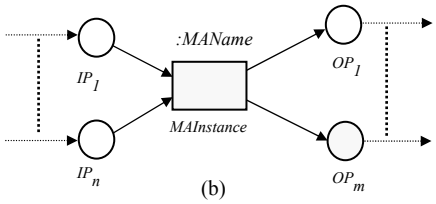
Now, we introduce hierarchical stochastic activity networks (HSANs). HSAN models provide hierarchies for a new definition of SANs. These models have a new element called *macro activity*, in addition to the five primitives of SANs. A macro activity is an HSAN submodel, which is composed of SAN elements or other macro activities.

The syntax of the usage of macro activity is similar to the usage of timed or instantaneous activities. *Place fusion* is used as a mechanism for interfacing macro activities to other parts of an HSAN model.

A macro activity may have zero or more *input* and *output fusion places*. Fusion places are a subset of normal places. A macro activity has a well-defined interface that is similar to parameter passing of procedure and functions in high-level programming languages. The places surrounding a macro activity are *formal fusion places*. When a macro activity is used in an HSAN model, these formal places will be bound by *actual places,* which are normal places of SANs. In a graphical representation, a fusion place is depicted as . A graphical representation of a macro activity is shown in Fig. 2(a) and its usage in Fig. 2(b).



(a)



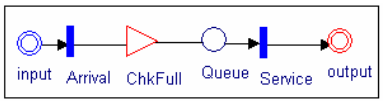
(b)

Fig. 2. Graphical representation of a macro activity: (a) Definition, and (b) Usage

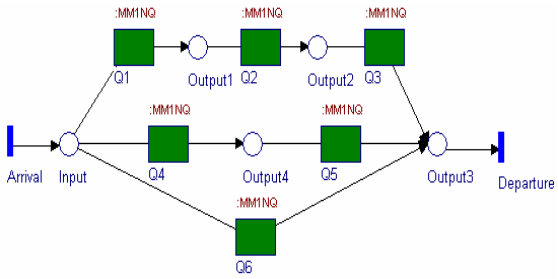
As an example of HSAN models, please see the model depicted in Fig. 3. Fig. 3(a) displays a macro activity, called *MM1NQ*, for the M/M/1/N queue. The macro activity has the following elements: *input* (an input fusion place), *Arrival* (a timed activity that models the arrival process), *ChkFull* (an output gate that checks for the capacity of the queue, *N*), *Queue* (a place that models the queue line), *Service* (a timed activity that models the service time) and *output* (an output fusion place).

The gate table for the macro activity *MM1NQ* is shown in Tab. 1. In this table, the function of output gate *ChkFull* is defined. This function checks whether the queue is full or not. *Arrivals*, *Rejected* and *N* are three global variables, which denote the number of arrivals to the system, the number of rejected customers and the capacity of the queue, respectively.

Fig. 3(b) displays the HSAN model for a queueing network, composed of six tandem/parallel queues. *Q1* through *Q6*, are some instantiations of the *MM1NQ*. *Arrival* and *Departure* timed activities models the input and output process of the network. *Input*, *Output1* through *Output4* are some places, which are bound to the input/output fusion places of *Q1* through *Q6*.



(a)



(b)

Fig. 3. HSAN representation of a queueing network: (a) *MM1NQ* macro activity, (b) The HSAN model

Tab. 1. Gate table for the macro activity *MM1NQ* of Fig. 3

Gate Name	Function

Output	Arrivals++;
	if (MARK(Queue) < N)
	MARK(Queue)++;
	else
	Rejected++;

4. FORMAL DEFINITIONS AND BEHAVIOUR OF HSANS

Formal definition of HSANs is the basis for analytic solution and simulation of these models. Based on these definitions and a well-defined syntax and semantics, HSAN tools will facilitate the process of modelling and analysis with these models.

The following formal definitions are based on a new definition of SANs [Movaghar, 2001]. In the following definitions, *N* denotes the set of natural numbers and R^+ represents the set of non-negative real numbers.

4.1 Formal Definitions of HSANs

In this subsection, we formally define hierarchical stochastic activity networks, macro activity and some other concepts of HSANs.

Definition 4.1. *Stochastic activity network (SAN)* is defined as a 11-tuple *HSAN* = (*P*, *IA, TA, IG*, *OG*, *IR*, *OR*, *C, F*,*Π, ρ*) where:

- *P* is a finite set of *places*,
- *IA* is a finite set of *instantaneous activities*,
- *TA* is a finite set of *timed activities*,

- *IG* is a finite set of *input gates*. Each input gate has a finite number of inputs. To each $G \in IG$, with *m* inputs, is associated a *function* $f_G : N^m \rightarrow N^m$, called the function of *G*, and a predicate g_G: $N^m \rightarrow \{true, false\}$, called the *enabling predicate* of *G*,
- *OG* is a finite set of *output gates*. Each output gate has a finite number of outputs. To each $G \in OG$, with *m* outputs, is associated a *function* $f_G : N^m \rightarrow N^m$, called the function of *G*,
- $IR \subseteq P \times \{1, ..., |P|\} \times IG \times (IA \cup TA)$ is the *input relation*. *IR* satisfies the following conditions:
  - For any $(P_1, i, G, a) \in IR$ such that *G* has *m* inputs, $i \leq m$,
  - For any $G \in IG$ with *m* inputs and $i \in N$, i ≤ m, there exist $a \in (IA \cup TA)$ and $P_1 \in P$ such that $(P_1, i, G, a) \in IR$,
  - For any $(P_1, i, G_1, a), (P_1, j, G_2, a) \in IR$, $i = j$ and $G_1 = G_2$.

  In a graphical representation, $(P_k, k, G, a) \in IR$ means that place P_k is linked to *k*-th input of an input gate *G* whose output is connected to activity *a*. P_k is said to be an input place of *a* and *G* is referred to as an input gate of *a*.
- $OR \subseteq (IA \cup TA) \times OG \times \{1, ..., |P|\} \times P$ is the *output relation*. *OR* satisfies the following conditions:
  - For any $(a, i, G, P_1) \in OR$ such that *G* has *m* outputs, $i \leq m$,
  - For any $G \in OG$ with *m* outputs and $i \in N$, $i \leq m$, there exist $a \in (IA \cup TA)$ and $P_1 \in P$ such that $(a, G, i, P_1) \in OR$,
  - For any $(a, G_1, i, P_1), (a, G_2, j, P_1) \in OR$, $i = j$ and $G_1 = G_2$.

  In a graphical representation, $(a, G, k, P_k) \in OR$ means that activity *a* is linked to the input of an output gate *G* whose *k*-th output is connected to place P_k. *G* is said to be an output gate of *a* and P_k is referred to as an output place of *a*.
- *C*: $N^n \times IA \rightarrow [0, 1]$ is the *case probability function*, where $n = |P|$.
- $F = \{F(.|\mu, a); \mu \in N^n, a \in TA\}$ is the set of *activity time distribution functions*, where $n = |P|$ and, for any $\mu \in N^n$, and $a \in TA$, $F(.|\mu, a)$ is a probability distribution function,
- *Π*: $N^n \times TA \rightarrow \{true, false\}$ is the *reactivation predicate*, where *n* is defined as before,
- ρ: $N^n \times TA \rightarrow R^+$ is the *enabling rate function*, where *n* is defined as before.

HSAN macro activity class is formally defined as follows:

Definition 4.2. An HSAN *macro activity class* (MAC) is defined as a 3-tuple *MAC* = (*SAN*, *IFP*, *OFP*) where:

- *SAN* is defined as in Definition 4.1,
- *IFP* is the list of *input fusion places*, such that:
  - $OFP \subseteq P$,
- *OFP* is the list of *output fusion places*, such that:
  - $OFP \subseteq P$,
  - $IFP \cap OFP = \varnothing$,
  - $IFP \cup OFP \neq \varnothing$.

Instances of a macro activity class can be used to compose an HSAN model. The HSAN model is defined as follows:

Definition 4.3. *Hierarchical stochastic activity network (HSAN)* is defined as a is defined as a 4-tuple *HSAN* = (*SAN, δ, MA, FF*) where:

- *SAN* is defined as in Definition 4.1,
- *δ* is a finite set of *macro activity classes* as in Definition 4.2,
- *MA* is a finite set of *macro activities*. To each *ma* ∈ *MA* is associated a macro activity class *mac* ∈ *δ*.
- *FF* is a *fusion function*, which is defined as,

  $$FF: MA \times FP \rightarrow P.$$

  The fusion function *FF*, maps each *fusion place fp* of *ma* to a place *p*, where *ma* ∈ *MA*, *fp* ∈ *ma.FP*, *ma.FP* = *ma.IFP* ∪ *ma.OFP* and *p* ∈ *P*.

Definition 4.4. Consider an HSAN as in Definition 4.3. A *marking* is a function μ: $P \rightarrow N$, where for each $P_i \in P$, $\mu(P_i) \in N$. It is often convenient to characterize a marking μ as a vector, that is, $\mu = (\mu_1, \ldots, \mu_n)$, where $\mu_i = \mu(P_i)$, i = 1, ..., *n*, and $P_i \in P$. An activity is *enabled* in a marking if the enabling predicates of its input gates are *true* in that marking. More formally, we have:

Definition 4.5. Consider an HSAN as in Definition 4.3. $a \in (IA \cup TA)$ is *enabled* in a marking μ if for any input gate *G* of *a* with *n* inputs and an enabling predicate g_G,

$$g_G(\mu_1, \ldots, \mu_n) = true,$$

where $\mu_k = \mu(P_k)$, for some $P_k \in P$ such that $(P_k, k, G, a) \in IR$, $k = 1, \ldots, n$. An activity is *disabled* in a marking if it is not enabled in that marking. A marking is *stable* if no instantaneous activity is enabled in that marking. A marking is *unstable* if it is not stable.

4.2 Well-Defined HSAN Models

HSANs are hierarchical models composed of places, timed, instantaneous or macro activities. A macro activity itself is composed of some places and timed, instantaneous or other macro activities. Therefore, it is needed to check for a *cycle* in the nested usage of macro activities. This check is done on the *dependency graph* (or *composition graph*) of the model. In this graph, each *node* is a macro activity and a composition relation between two macro activities determines an *arc*. The *root* node of the dependency graph is the HSAN model. If there is no cycle in the dependency graph of an HSAN model, it is well-defined and can be analysed by a state space analysis or simulation method. Therefore, we have the following definitions:

Definition 4.6. *Dependency graph* (DG) of an HSAN model, as in Definition 4.3, is defined as a triplet $DG = (N, A, R)$, where:

- $N = \{H\} \cup MA$ is the set of nodes of the graph, such that:
  - H is the *root* node of the model, and
  - MA is the set of macro activities of the model,
- A is set of arcs of the graph, and
- R is the *composition relation* between two nodes such that: $R: A \rightarrow N \times N$.

Definition 4.7 An HSAN model is *well-defined*, if its dependency graph is *finite* and *acyclic*.

4.3 Behaviour of HSANs

A hierarchical stochastic activity network (HSAN) with a marking is a dynamic system. The behaviour of HSANs can be described as follows:

In an HSAN model, a marking will change only if an activity either in the root level of the model or in one of its macro activities *completes*. In a stable marking, only one of the enabled timed activities is allowed to complete. When there is more than one enabled timed activity, the choice of which activity to complete first is done stochastically. Enabled timed activities require some time to complete. A timed activity becomes *active* as soon as it is enabled and remains so until it completes; otherwise, it is *inactive*. Consider a hierarchical stochastic activity network H as in Definition 4.3. Suppose, at time t, a timed activity completes, and μ is the stable marking of H immediately after t. A timed activity a is activated at t, if a is enabled in μ and one of the following occurs:

- a is inactive immediately before t,
- a completes at t,
- $\Pi(\mu, a) = true$.

Whenever the above happens, a is assigned an activity time τ, where τ is a random variable with probability distribution function $F(.|\mu, a)$. When a timed activity a is enabled in a stable marking μ, it is processed with a rate $\rho(\mu, a)$. A timed activity completes whenever it is processed for its activity time. Upon completion of an activity, the next marking occurs immediately.

In an unstable marking, only one of the enabled instantaneous activities may complete (i.e., enabled instantaneous activities have priority over enabled timed activities for completion). When there is more than one enabled instantaneous activity, the choice of which activity to complete first is made probabilistically. More specifically, let H be a hierarchical stochastic activity network as in Definition 4.3. Suppose, H is in an unstable marking μ. Let A' be the set of enabled instantaneous activities of H in μ. Then, $a \in A'$ completes with probability α, where

$$\alpha = \frac{C(\mu, a)}{\sum_{a' \in A'} C(\mu, a')} \quad (4.1).$$

When an activity completes, it may change the marking of its input and output places. This change is governed by the functions of its input gates and output gates, and is done in two steps as follows. First, the marking of its input places may change due to the functions of its input gates, resulting into an intermediary marking. Next, in this latter marking, the marking of its output places may also change due to the functions of its output gates, resulting into a final marking after the completion of that activity. More specifically, let us consider a hierarchical stochastic activity network as in Definition 4.3. Suppose an activity, a, completes in a marking μ. The next marking μ' is determined in two steps as follows. First, an intermediary marking μ'' is obtained from μ by the function of input gates of a. μ' is then determined from μ'' by the function of output gates of a. More formally, μ' and μ'' are defined as follows:

- For any $P_1 \in P$, which is not an input or output place of a, $\mu'(P_1) = \mu(P_1)$,
- For any input gate G of a with m inputs and a function f_G, $f_G(\mu_1, \ldots, \mu_m) = (\mu''_1, \ldots, \mu''_m)$, where $\mu_k = \mu(P_k)$ and $\mu''_k = \mu''(P_k)$ such that $(P_k, k, G, a) \in IR$, $k = 1, \ldots, m$.

- For any output gate G of a with m outputs and a function f_G, $f_G(\mu''_1, ..., \mu''_m) = (\mu'_1, ..., \mu'_m)$, where $\mu''_k = \mu''(P_k)$ and $\mu'_k = \mu'(P_k)$ such that $(a, G, k, P_k) \in OR$, $k = 1, ..., m$.

The above summarizes the behaviour of an HSAN. This behaviour may be studied more formally using the concept of the state process of HSANs as in the following subsection.

3.2 State Process of HSANs

In order to study the state process of HSANs, we need to define the notion of a *hierarchical probabilistic activity networks* and *hierarchical probabilistic activity system*:

Definition 4.8. A *hierarchical probabilistic activity network* is derived from HSAN, where the stochastic properties of timed activities (F, ρ and Π) are eliminated. In the resulting model, nondeterminacy is specified probabilistically by case probability functions (C) of instantaneous activities.

Definition 4.9. A *hierarchical probabilistic activity system* is a 4-tuple (Q, A, h, p_0) where:

- Q is a set of *states*,
- A is the *activity* alphabet,
- $h = \{h(.|q, a);\ q \in Q,\ a \in A\}$ is the set of *transition distributions* such that, for any $q \in Q$ and $a \in A$, $h(.|q, a) = 0$ or $h(.|q, a)$ is a probability distribution over Q,
- p_0 is the *initial state distribution,* which is a probability distribution over Q.

For $a \in A$ and $q, q' \in Q$, q' is said to be *immediately reachable* from q under a with probability α, if $h(q'|q, a) = \alpha$.

We now present a notion of *equivalence* for hierarchical probabilistic activity systems.

Definition 4.10. Let $U = (Q, A, h, p_0)$ and $U' = (Q', A', h', p'_0)$ be two hierarchical probabilistic activity systems with the same activity alphabet (i.e., $A = A'$). U and U' are said to be equivalent if there exists a symmetric binary relation γ on $Q \cup Q'$ such that:

- $Q = \gamma(Q')$ and $Q' = \gamma(Q)$,
- For any $q_0 \in Q$ and $q'_0 \in Q'$ such that $(q_0, q'_0) \in \gamma$,

$$\sum_{q \in \gamma(\{q'_0\})} p_0(q) = \sum_{q' \in \gamma(\{q_0\})} p'_0(q'),$$

- For any $a \in A$, $q_1, q_2 \in Q$, and $q'_1, q'_2 \in Q'$ such that $(q_1, q'_1) \in \gamma$ and $(q_2, q'_2) \in \gamma$,

$$\sum_{q \in \gamma(\{q'_2\})} h(q \mid q_1, a) = \sum_{q' \in \gamma(\{q_2\})} h'(q' \mid q'_1, a).$$

γ above is said to be a *bisimulation* between U and U'. U and U' are *isomorphic* if γ is a *bijection*.

Definition 4.11. Let H be an HSAN as in Definition 4.3. The state process of H is a random process $\{X(t);\ t \in R^+\}$ where $X(t)$ denotes the stable marking of H at time t.

Definition 4.12. Let $X = \{X(t);\ t \in R^+\}$ and $X' = \{X'(t);\ t \in R^+\}$ be two *random processes* with the set of states Q and Q', respectively. X and X' are said to be *stochastically equivalent* if there exists a symmetric binary relation γ on $Q \cup Q'$ such that:

- $\gamma(Q) = Q'$ and $\gamma(Q') = Q$,
- For any $t_i \in [0, \infty)$, $Q_i \subseteq Q$, and $Q'_i \subseteq Q'$, such that $Q_i = \gamma(Q'_i)$ and $Q' = (Q_i)$, $i = 0, \ldots, n \in N$,

$$p[X(t_i) \in Q_i;\ i = 0, \ldots, n] = p[X'(t_i) \in Q';\ i = 0, \ldots, n].$$

X and X' are stochastically *isomorphic* (equal) if γ is a *bijection* (an equality).

We have:

Proposition 4.1. Let $H = (L, F, \Pi, \rho)$ and $H' = (L', F', \Pi', \rho')$ be two HSANs where L and L' are some equivalent hierarchical probabilistic activity networks. Suppose, L and L' realize hierarchical probabilistic activity systems $U = (Q, A, h, p_0)$ and $U' = (Q', A', h', p'_0)$, respectively. ($A = A'$.) The state processes of H and H' will be *stochastically equivalent* if there exists a symmetric binary relation γ on $Q \cup Q'$ such that:

- γ is a *bisimulation* between U and U',
- For any $a \in A$, $q \in Q$, and $q' \in Q'$ such that $(q, q') \in \gamma$ and a is enabled in both q and q', $F(.|q, a) = F'(.|q', a)$, $G(q, a) = G(q', a)$, and $\rho(q, a) = \rho'(q', a)$.

The state behaviour of an HSAN is closely related to the notion of a *generalized semi-Markov process* as defined in [Schassberger, 1978]. We get:

Proposition 4.2. The following statements are true:

- Any generalized semi-Markov process with a finite set of events is stochastically isomorphic to the state process of an HSAN,
- There exists an HSAN whose state process is not a generalized semi-Markov process,
- The state process of any HSAN with state-independent activity time distribution functions and a false *reactivation predicate* is a generalized semi-Markov process.

We now consider Markovian models. We have:

Theorem 4.1. Let H be an HSAN as in Definition 4.3. The state process of H is a Markov process *iff* for any timed activity a which is enabled in a stable marking μ, and any stable marking μ_{ac} in which a is last activated prior to being enabled in μ,

$$F(\tau|\mu_{ac}, a) = 1 - e^{-\alpha(\mu, a)\tau}$$

where $\alpha(\mu, a)$ is a positive real number which only depends on μ and a.

Proof. Let $H = (L, F, \Pi, \rho)$ with a corresponding hierarchical probabilistic activity network L which realizes a *hierarchical probabilistic activity system* $U = (Q, A, h, p_0)$. Denote $X = \{X(t), t \in R^+\}$ the state process of H.

if: We note that for any t, $\delta t \in R^+$, where δt is sufficiently small, and any stable markings $\mu, \mu' \in Q$, we have

$$\begin{aligned} P[X(t+\delta t) = \mu' | X(t) = \mu, X(t'), 0 \le t' < t] &= P[X(t+\delta t) = \mu' | X(t) = \mu] \\ &\approx \sum_{a \in A} \alpha(\mu, a)\rho(\mu, a)h(\mu, \mu')\delta t \end{aligned}$$

Using the memoryless property of exponentially distributed random variables and the dynamic behaviour of the model, we can conclude that X is a Markov process.

only if: Note that exponentially distributed random variables are the only random variables with memoryless property and that X is assumed to be a Markov process. The proof then follows from the definition of the dynamic behaviour of the model. □

An HSAN is said to be Markovian if its state process is a Markov process. We find:

Corollary 4.1. Let H be an HSAN with a set of exponential activity time distribution functions such that any activity with a state-dependent activity time distribution function has also a true *reactivation predicate*. Then, H is Markovian.

Corollary 4.2. Any discrete-space, continuous-time, and time-homogeneous Markov process is stochastically isomorphic to the state process of a Markovian HSAN.

5. SOLUTION OF HSAN MODELS

If an HSAN model is well-defined (Definition 4.6 and Definition 4.7), it can be solved by analytic solution or simulation methods; otherwise, it cannot be solved. It is possible to transform a well-defined HSAN model into an equivalent flat SAN model. Then, it is possible to employ the existing analytic solution or simulation techniques. It is also possible to transform an HSAN model to appropriate composition formalism. These methods are described in the following subsection.

5.1 State Spaces of HSANs

State space analysis is the standard method for the analysis of Petri net models. The basic idea behind this method is to construct a *directed graph*, which has a *node* for each reachable system state and an *arc* for each possible state change.

Definition 5.1. *State space* (SS) of an HSAN model is defined as a 4-tuple SS = (M, E, A, M_0), where:

- M is the set of nodes (reachable markings of the graph) such that $\forall \mu \in M$: $\mu = (\mu_1, ..., \mu_n)$, where $\mu_i = \mu(P_i)$, $i = 1, ..., n$, $n = |P|$ and $P_i \in P$.
- E is set of arcs (edges) of the graph, and
- A is a node function such that: $A: E \rightarrow M \times M$,
- M_0 is the *initial marking* of the HSAN model such that: $M_0 = (\mu_1, ..., \mu_n)$, where $\mu_i = \mu(P_i)$, $i = 1, ..., n$, $n = |P|$ and $P_i \in P$.

5.2 Transformation of HSANs into a Flat HSAN Model

If an HSAN model is well-defined, it has an equivalent flat SAN model. For the analysis of an HSAN model, it is possible to transform it into an equivalent flat SAN model. A substitution algorithm can be employed to flatten the HSAN model. In each step of the algorithm, all macro activities on the leaves of the graph will be substitute by their definitions. This will be repeated until the only node on the graph is the root node. The resulting model is a flat SAN model. Since, naming in HSAN models is local, names of two elements in two different levels of the hierarchy of an HSAN model may be identical. To resolve the problem of duplicate names in the substitution of macro activity with its definition, the name of each element of an macro activity will be preceded by the name of its parent macro activity. For example, if we have an macro activity named *ma1* in an HSAN model and there is a place named *p1* in the definition of *ma1*, a place named *ma1.p1* will be added to the flattened model.

5.3 Transformation of HSANs into Composition Formalisms

The main disadvantage of the above flattening method is the explosion of nodes in the resulting model that may lead to the explosion of the state space of the model. There are a few techniques for constructing SAN models in a way, which avoid state-space explosion problem. A key benefit of HSAN models is the possibility of automatic employment of such techniques by their modelling tools. A modelling tool for HSANs can transform a model into the following composition formalisms:

1. *Replicate/Join construct:* A composition technique for SANs is the *Replicate/Join construct* [Sanders and Meyer, 1991]. A possible

way of the solution of HSAN models is to transform them into this construct. A modelling tool for HSANs can check for a tree-like structure in the dependency graph of the model. It can automatically find the *shared states* based on fusion places of macro activities. Then, it can obtain the hierarchical and composed model using Replicate and Join operations. For the solution of the resulting model, the technique proposed in [Sanders and Meyer, 1991 or Stillman, 1999] can be employed.

2. *Graph Composition formalism:* Another composition formalism, which has been proposed in the Möbius modelling framework for SANs and other models, is the *Graph Composition formalism* [Stillman, 1999]. This formalism does not limit the model hierarchy to a tree-like structure and any arbitrary structure (such as ring or mesh) is possible. A modelling tool for HSANs can check for the possibility of the usage of this formalism. Then, it can transform the corresponding HSAN model and uses the method proposed in [Stillman, 1999] to solve the model.

5.4 Discrete-Event Simulation of HSAN Models

If an HSAN model or one of its macro activities is composed of one or more non-exponential timed activities or its state-space is infinite, it may not be solved analytically. In such cases, discrete-event simulation may be employed to solve the model. A discrete-event simulation algorithm for an HSAN model may have the following steps:

1. *Determine the set of enabled activities in the current marking of the model.*
2. *Reactivate those disabled activities whose reactivation predicates are true in the current marking.*
3. *Generate the activity execution time for newly enabled or reactivated activities.*
4. *Apply the corresponding enabling rates on the remaining time of the enabled timed activities.*
5. *If more than one instantaneous activity is enabled in an unstable marking, select one of them probabilistically. The activity selection probability (α) is computed by formula (4.1) of Sec. 4.3.*
6. *In a stable marking (i.e. no instantaneous activity is enabled), considering the remaining time of all enabled timed activities and their respective enabling rates, select the next timed activity for completion. In the case of two equal completion times, select one of the corresponding timed activities, probabilistically.*
7. *Fire the selected instantaneous or timed activity:*
   - *Remove a token from all input places,*
   - *Execute the function of all input gates,*
   - *Add a token to all output places, and*
   - *Execute the function of all output gates.*
8. *Disable those enabled activities whose enabling predicate are not true in the current marking.*
9. *Set the simulation clock to the time of the most eminent event.*
10. *Collect the aggregated statistics and update user-defined queries.*
11. *If the specified confidence level (or fixed replications/time interval) has not been achieved, go to step (1).*

It is needed to introduce fast and efficient techniques for the simulation of HSAN models. These methods may be based on a method proposed for fast simulation of SAN composed models [Sanders and Freire, 1993] or techniques for discrete-event simulation in the Möbius framework [Williamson, 1998].

5.5 Tool Support for HSANs

For modelling and analysis with SAN-based models, including *a new definition of SANs* [Movaghar, 2001] and HSANs we have developed a modelling tool called *SANBuilder* [Abdollahi and Movaghar, 2004]. SANBuilder runs under Windows 2000/XP and enables the modeller to construct SAN and HSAN models in a graphical editor, define and save macro activities and load them to reuse in a new model.

SANBuilder has an *integrated development environment* (IDE) for construction, compilation, animation, discrete-event simulation, and analytic solution of SAN-based models. We have used this tool for the construction of an HSAN model of a queueing network that is appeared in Fig. 3.

A view of the user interface of SANBuilder is displayed in Fig. 4.

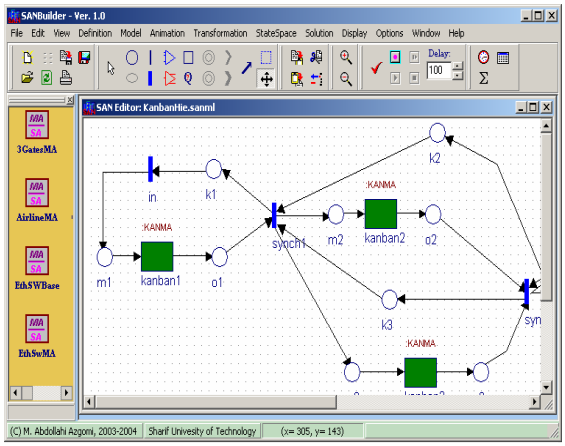


Fig. 4. A view of the user interface of SANBuilder

For more information about SANBuilder, including its features, structure and solution techniques, please see [Abdollahi and Movaghar, 2004]

6. CONCLUSIONS

In this paper, we have presented the informal and formal definitions and behaviour of *hierarchical stochastic activity networks* (HSANs). HSAN models provide facilities for composing a hierarchy of submodels that are called *macro activity*. HSAN models encapsulate hierarchies in away similar to the programming languages.

Similar to the ordinary SANs, HSANs can be used for modelling and analysis of various kinds and different aspects of computer and communication systems. HSANs can be used to build both Markovian and non-Markovian models.

A summary of the advantages of HSANs is as follows:

1. *HSANs are hierarchical models.* Macro activities encapsulate hierarchies and provide compositionality in a natural way and similar to object-oriented programming languages. HSAN models represent the hierarchy of hierarchical systems in a natural and understandable manner. Especially for software systems, HSANs are more appropriate than SANs plus composition techniques.
2. *HSANs provide top-down modelling paradigm.* HSANs prepare a top-down model construction paradigm. While the approaches of SANs plus composition techniques are bottom-up.
3. *HSANs allow incremental modelling.* It is possible to incrementally construct large HSAN models for complex systems by starting with simple macro activities and easily replacing them with enhanced ones.
4. *HSANs encourage reusability of submodels.* The existence of macro activity with well-defined interface makes it possible to construct and share a repository of reusable HSAN submodels.
5. *Automatic code generation from HSAN models is possible.* This can be achieved by modelling tools for HSANs and is a key requirement to use these models for software systems.
6. *Reduced state space generation for HSAN models is possible.* With automatic transformation into an appropriate composition technique, the advantages of them for reduced state space generation can still be taken.

We have also introduced basic methods for state space analysis and simulation of HSAN models and have shown how the existing methods can be used for this purpose. For modelling with HSANs, we have developed a modelling tool called SANBuilder. To make this tool more useful for the application on large-scale systems, we are working to implement efficient methods for the solution and simulation of HSAN models.

REFERENCES

Abdollahi Azgomi M. and Movaghar A. 2003, "Hierarchical Stochastic Activity Networks," Proc. of 10th Int. Conf. on Analytical and Stochastic Modelling Tech. and App. (ASMTA03), Nottingham, UK Pp169-174.

Abdollahi Azgomi M. and Movaghar A. 2004, "A Modelling Tool for Hierarchical Stochastic Activity Networks," Proc. of 11th Int. Conf. on Analytical and Stochastic Modelling Tech. and App. (ASMTA04), Magdeburg, Germany (2004) Pp141-146.

Ajmone Marsan M.; Balbo, G. and Conte, G.: A Class of Generalized Stochastic Petri Nets for Performance Evaluation of Multiprocessors Systems, ACM Trans. Comp. Sys., 2(2) (1984) 93-122

Deavours D.D. 2002, *Formal Specification of The Möbius Modelling Framework*, Ph.D. Dissertation, University of Illinois at Urbana-Champaign.

Deavours D.D. et al 2002, "The Möbius Framework and Its Implementation," *IEEE Trans. on Soft. Eng.*, Vol. 28, No. 10, Pp956-969.

Molloy M.K. 1982, "Performance Analysis Using Stochastic Petri Nets," *IEEE Trans. on Computers*, C-31, Pp913-917.

Movaghar A. and Meyer J.F. 1984, "Performability Modelling with Stochastic Activity Networks," Proc. of the 1984 Real-Time Systems Symp., Austin, TX, USA, Pp215-224.

Movaghar A. 1985, *Performability Modelling with Stochastic Activity Networks*, Ph.D. Dissertation, University of Michigan.

Movaghar A. 2001, "Stochastic Activity Networks: A New Definition and Some Properties," *Scientia Iranica,* Vol. 8, No. 4, Pp303-311.

Peterson J.L. 1981, *Petri Net Theory and the Modelling of Systems*, Prentice-Hall.

Sanders W.H. and Freire R.S. 1993, "Efficient Simulation of Hierarchical Stochastic Activity Network Models," Discrete Event Dynamic Systems: Theory and App., Vol. 3, no. 2/3, Pp271-300.

Sanders W.H. and Meyer J.F. 1991, "Reduced Base Model Construction Methods for Stochastic Activity Networks," *IEEE J. on Selected Areas in Comm.*, Vol. 9, No. 1, Pp25–36.

Sanders W.H. et al 1995, "The UltraSAN Modelling Environment," *Performance Evaluation*, Vol. 24, Pp1-33.

Schassberger R. 1978, "Insensitivity of Steady-State Distributions of Generalized Semi-Markov Processes with Speeds," *Adv. Appl. Prob.*, Vol. 10, Pp836-851

Stillman A.J. 1999, *Model Composition in the Möbius Modelling Framework*, M.S. Thesis, University of Illinois at Urbana-Champaign.

Williamson A.L. 1998, *Discrete Event Simulation in the Möbius Modelling Framework*, M.S. Thesis, University of Illinois at Urbana-Champaign.

BIOGRAPHIES

Mohammad Abdollahi Azgomi received the B.S. and M.S. degrees in computer engineering (1991 and 1996) from Sharif University of Technology, Tehran, Iran. He is currently a Ph.D. candidate in computer engineering at the department of computer engineering, Sharif University of Technology, under the supervision of professor Ali Movaghar. His research interests include modelling and evaluation with Petri nets and stochastic activity networks, object-oriented modelling and network security.

Ali Movaghar received the B.S. degree in electrical engineering (1977) from Tehran University and the M.S. and Ph.D. degrees in computer, information and control engineering (1979 and 1985) from the University of Michigan, Ann Arbor, USA. His research interests include performance and dependability modelling, verification and validation, computer networks and distributed real-time systems. He is currently a faculty member at the department of computer engineering, Sharif University of Technology, Tehran, Iran.