Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

PEG: FORMALIZING LP-BASED MODELING OF PARALLEL DISCRETE-EVENT SIMULATION

Bing Wang, Bo Deng

Software Division,
Beijing Institute of Systems Engineering,
Beijing 100101, P.R.China

Fei Xing, Yiping Yao

School of Computer,
National University of Defense Technology,
Changsha 410073, P.R. China

ABSTRACT

Logical process (LPs) is a widely used modeling paradigm in parallel discrete-event simulation (PDES). However, there is a lack of effective methods for formalizing the LP-based modeling of PDES. This situation obstructs the unambiguous and platform-independent description of the LP-based model. To solve this problem, in this work, a formalism named Partitioned Event Graph (PEG) is presented. PEG is an extension to the classical Event Graph formalism, towards the formal specification of the LP-based models in PDES. We construct a mapping between PEG and LP-based models, and define the Structural Operational Semantics(SOS) of PEG on Timed-Transition System. Finally, the Wallclock Time based Execution is discussed. PEG formally specifies the event scheduling and the state partition properties of the LP paradigm, and thus provides modelers with a mean for cross platform model transformation and formal analysis to the model's behavior in PDES.

1 INTRODUCTION

Parallel discrete-event simulation (PDES) has been recognized as a challenging field of research bridging modeling and simulation, and high-performance computing(Liu, Cochran, Cox, Keskinocak, Kharoufeh, and Smith 2011). In PDES, performance gain against sequential Discrete Event System(DES) comes from explicitly decomposing the model into several sub models, spreading the computing load over distributed processors and (speculatively) pushing the simulation time forward. Logical Process (LP) is the most common used modeling paradigm in PDES, which term was introduced to PDES as a metaphor for the physical processes a system consists of. Because of its intuitive depiction of the time and space, as well as the commonality in implementation on parallel machines, LP paradigm has long been the dominating modeling method in PDES.

Over the last decades, many solutions for the simulation of LP-based models have been proposed (Fujimoto 2000). They usually aim at overcoming the performance bottleneck of a sequential DES (Misra 1986). Despite the fruitful work on efficient algorithms, customization to novel hardware platforms, and effective modeling methodology, however, there is a lack of effective methods for formalizing the LP-based modeling of PDES. This situation obstructs the unambiguous and platform-independent description of the LP-based models. Formal Semantics is the basis of a unambiguous specification of the model's behavior, and formal verification to the model's properties.

In this paper, we develop a formalism for PDES based on Event Graph(Schruben 1983), called PEG (Partitioned Event Graph). PEG not only presents a mapping to LP-based model, but also provides a Structured Operational Semantics on Timed-Transition System. PEG formally specifies the event scheduling and the state space partition of the LP paradigm, and these two properties respectively correspond to synchronization and partitioning, which are two important issues that strongly influence the correctness and performance

of PDES. Therefore PEG can be used for cross platform model transformation , automatical compilation into programming languages, and formal analysis to the models behavior in PDES.

The next section presents the background and related works with formalizing LP-based modeling of PDES. In section 3, we present Partitioned Event Graph in detail. We construct a mapping between PEG and LP-based model, and define the Structural Operational Semantics(SOS) of PEG on Timed-Transition System. Section 4 discusses the Wallclock Time Execution and equivalence relationship. Section 5 closes the paper with a discussion on the major findings and provides some direction for future work.

2 BACKGROUND AND RELATED WORKS

2.1 Logical Process and Event Graph

There are many definitions for the term Logical Process. Sometimes LP is referred to as a synonym for any sequential simulation or simulator (Fujimoto 2000, p. 40), and it has been introduced to sequential simulation branching (Curry et al. 2005, Peschlow et al. 2008). In time-parallel simulation, which decomposes the model temporally, every computation unit is also called LP (Fujimoto 2000, p. 177 et sqq.). We regard LPs as the core units of execution during the simulation (Chandy and Misra 1979, Misra 1986). They interact with each other by exchanging *events*. This definition of LPs as model entities is common in the field of PDES, e.g., in (Fujimoto 2000). We also would like to emphasize here that there are other levels of parallelism that can be exploited for parallel simulation, e.g., executing replications in parallel (e.g., (Leye et al. 2008)), or by conducting parallel computations within an LP.

The development of PEG formalism is based on the Event Graph formalism (Schruben 1983, Buss 2002), which has two other equivalent forms known as Simulation Graph (Yücesan and Schruben 1992) and event relationship graph(Schruben 1995). An EG model is being described by states which are changed according to incoming events and according to further conditions.

EG formalism explicitly expresses all the relationships between events and is the most direct way for the event scheduling world view. It has been shown that EG has the same expressiveness as Turing Machine(Savage, Schruben, and Yucesan 2005). Infinitesimal Perturbation Analysis(IPA) and Mathematical Programming are deployed to analysis the properties of EG models(Schruben, Roeder, Chan, Hyden, and Freimer 2003). OMIGA, SIMKIT(Buss 2002) and Viskit(Buss and Sanchez 2002)are typical toolkits for EG based modeling, but only for sequential execution. However, EG formalism can not yet describe the state space partition in LP paradigm.

2.2 Related Work

There are many classes of formalisms for discrete event system. They differ in the types of domains they are suitable for describing, in the types of properties,aspects, granularities they can be used to analysis, in their expressive power and underlying semantic model, in their usability, etc.

The common used formalisms for discrete event system modeling are DEVS (Zeigler, Praehofer, and Kim 2000),Timed Automata,Petri Net, Process Algebra GSMP (Cassandras and Lafortune 2008), Event Graph (Savage, Schruben, and Yucesan 2005), StateChart (Harel 1987), Control Flow Graphs (Cota, Fritz, and Sargent 1994) and activity cycle diagrams (Paul 1993), etc. The most common modeling method is to directly use the APIs provided by PDES platforms, such as TWOS(Rich and Michelsen 1991), GTW(Das, Fujimoto, Panesar, Allison, and Hybinette 1994), SPEEDES(Steinman 1991), POSE(Wilmarth 2005), ROSS(Carothers, Bauer, and Pearce 2000), μsik (Perumalla 2005).

API based modeling methods in PDES suffer from low usability because they are tightly coupled to the platforms. Domain specific languages hide the platform details, and thus makes the modeling more friendly to domain experts. For example, the Maisie(Bagrodia and Liao 1994) language is based on GTW, but the users do not have to know the details of the API. Since most domain specific modeling method in PDES are still tightly coupled with specific platforms and the execution semantics they provide, across platform reusability and formal analysis are hampered.

Visualization and component based modeling further reduce the difficulties in model construction and increase the reusability of the model constructed. In Control Flow Graph(Sargent 2004), both visualization and component based method are used to construct the LP-based models. In (Liu, Yao, and Liu 2010), a component based modeling environmentis built on top of YHSUPE, other works includes Jane(Perumalla and Fujimoto 2001) and Pave(Bagrodia 1998).

Mapping high level formalisms to LP paradigms is another choice to introduce formal method to PDES. Researchers in University of Carleton map cellular automata to LP paradigms(Wainer and Giambiasi 2002) based on DEVS, and implement a Cell-DEVS based PDES platform(Liu 2010). In SASSY multi-agent system is mapped to LP-based model(Hybinette, Kraemer, Xiong, Matthews, and Ahmed 2006). In Project MAS-PDES(Oguara, Chen, Theodoropoulos, Logan, and Lees 2005), researchers organize the LP-based models into reconfigurable tree structure based on the influence conical, and study the representation of transformation of information sharing among agents in LP paradigm.

Nutaro(Nutaro and Sarjoughian 2004) represents the simulation of LP-based model as a stack based computing process to facilitate formal analysis, but the special semantics it used makes it hard to incorporate common-used formal methods. Cristian Tapus discusses the formal model and operational semantics of speculative execution in distributed environment(Tapus 2006), which serves as a basis for the operational semantics of PDES but does not consider the equivalence to sequential execution as its purpose.

On one hand, although current formalisms have been proved useful for modeling general discrete event system, they do not yet well depict the soul properties of LP paradigm, which are state space partition, event-driven semantics, and partial order among events. On the other, current LP-based modeling of PDES depends on either high level programming language, or artificial structure to define its execution semantics. The lack of a formalism with formal semantics hampers further study on model transformation and formal verification of the LP-based model. To solve this problem, Partitioned Event Graph is put forward as a formalism for LP-based model in PDES. PEG is based on EG, and is attached with a structural operational semantics.

3 PARTITIONED EVENT GRAPH

3.1 Assumptions

Regulation of the system under investigation plays a central role in formal method. For the PEG formalism, the following assumptions are made:

- No simultaneous events. Since in simulation we are interested in the state trace generated by the system behavior in stead of high level interleaving concurrency, decoupling can be used to eliminate simultaneous events(Wieland 1997).
- The condition expression on edge should contains (finite) enumerable many relations joined by boolean operators, and thus the length of the edge condition is enumerable and finite(Yücesan and Schruben 1992).
- The amount of Logical Process Classes, LPs(partitions), Event Types, Event Types are respectively finite. The amount of Event Instance is enumerable, but not necessarily finite.
- The State space of the simulation model is enumerable but not necessarily finite, as for discrete event dynamic system model(Cassandras and Lafortune 2008).
- The model structure is static, no generation or reduction of LP instances during the simulation.

3.2 Logical Process based Model

General Discrete Event Model in event scheduling world view can be represented as:

Definition 1 $Model \triangleq \langle EventSet, StateVarSet, INIT, Time \rangle$

where

- *EventSet* holds all the Event types,
- *StateVarSet* stands for the set of state variables,
- *INIT* stands for the initial setup for *EventSet* and *StateVarSet*,
- *Time* stands for the time base.

A formalism for LP-based model has to allow at least the definition of events, states, and their dependencies. A discrete event system expressed in the logical process paradigm consists solely of "logical processes" instances as atomic model entities. We name logical process instance as logical process, and name the type which the logical process belongs to as logical process class.

Hence, a LP based model can be regarded as the partitions on *EventSet* and *StateVarSet* respectively. L is the set of indexed, and the L-indexed Set over Set_{LP} is a triple (L, A, μ) where $\mu : L \to Set_{LP}$ is the index function of Set_{LP}. since $\forall LP \in Set_{LP}, \exists i \in L, \mu(i) = LP$, we mark the LP as LP_i. Event Type in *EventSet* is composed from *EventBaseType* and the index of which LP it belongs to:

$$EventSet \triangleq EventBaseType \times L.$$

First, we define the set of Evaluation Functions, as shown in Definition 2.

Definition 2 (Set of Evaluation Functions) $EVAL(\{SV_1, \ldots, SV_M\})$ stands for the set of all evaluation functions to the state variable in *StateVarSet*, and is represented by $\prod_{0<j\leq M} domain_{SV_j}$. The evaluation function $\eta \in EVAL(\cdot)$ maps every $SV_j \in StateVarSet$ to a certain value v in $domain_{SV_j}$, where $0 < j \leq M$.

Definition 3 (Logical Process based Model) $Model_{LP} \triangleq \langle Set_{LP}, INIT, Time \rangle$ where

- $Set_{LP} \triangleq \{LP_i | i \in L\}$, where $\{LP_i \triangleq \langle EventSubset_i, StateVarSet_i, \delta_i^S, \delta_i^\eta \rangle, i \in L\}$
  - State variable subset $StateVarSet_i$ satisfies:

$$\forall i \in L. StateVarSet_i \subseteq StateVarSet$$
$$\cup_{i\in L} StateVarSet_i = StateVarSet$$
$$\forall i \neq j \in L. StateVarSet_i \cap StateVarSet_j = \phi$$

  - Event type subset $EventSubset_i$ satisfies:

$$\forall i \in L. EventSubset_i \subseteq EventSet$$
$$\cup_{i\in L} EventSubset_i = EventSet$$
$$\forall i \neq j \in L. EventSubset_i \cap EventSubset_j = \phi$$

  The State Space of LP_i is represented as $S_i \triangleq \Pi_{v\in StateVarSet_i} domain_v$,
  where $StateSpace \triangleq EVAL(StateVarSet) = \prod_{i\in L} S_i = \prod_{i\in L} EVAL(StateVarSet_i)$.
  Event Instance $event \triangleq \langle eT, fr, to, st, rt \rangle \in EventSet \times Set_{LP} \times Set_{LP} \times Time \times Time$,
  where $st, rt \in Time, fr \in Set_{LP}, eT \in EventSubset_i$.
  State Transition Function is represented as
  - $\delta_i^S : EventSubset_i \times S_i \to S_i, i \in L$.
  Event Scheduling Function is represented as
  - $\delta_i^\eta : EventSubset_i \times S_i \to \wp(EventSet \times Time), i \in L$.

  Types of Schedule Relationship are represented as $SchType \triangleq \{CancelSch, TimedSch\}$
  The schedule relationship for event types of LP_i is defined as $ScheduleRelationship_i$, which is a set of tuples consisted of the event types in LP_i and those event types scheduled by:

$$\forall (eT_{from}, eT_{to}) \in ScheduleRelationship_i$$
$$\Rightarrow (eT_{from} \in EventSubset_i) \wedge (\exists s \in S_i, \exists t \in T, s.t. (eT_{to}, t) \in \delta_i^\eta (eT_{from}, s))$$

The schedule relationship for $Model_{LP}$ is $ScheduleRelationship \triangleq \cup_{i \in L} ScheduleRelationship_i$.

- *Time* represents the time based, R^+ is used in this paper.
- *INIT* is the initial configuration for $EventSet_i$ $StateVarSet_i$,$(i \in L)$.

3.3 Partitioned Event Graph

The Event Graph formalism describes the elements of an event graph model, which is formally described in Definition 4.

Definition 4 (Event Graph) $M \triangleq < V, E, S, F, C, T, A, B >$, where

V, the set of vertices corresponding to the events of M,
E, the set of directed edges $e_{od} = < v_o, v_d >$ that describe the scheduling/canceling relationships between pairs of events in M,
S, the state space of M, it is depicted by a vector of state variable in DES,
$F = \{f_v : S \to S, \forall v \in V\}$ the set of state change functions each associated with each vertex v,
$C = \{c_e : S \to \{true, false\}, \forall e \in E\}$, the set of edge condition functions each associated with each edge e,
$T = \{t_e \in Time, \forall e \in E\}$, the time delay on each edge e(normally we use $\Re_0^+$ as Time Base),
$A = \{A_e, \forall e \in E\}$ the set of attribute lists, if any, associated with each edge e,
$B = \{B_v, \forall v \in V\}$,the set of parameter lists, if any, associated with each vertex v.

Definition 5 $Par \subseteq \wp(A)$ is a **partition** of A, if $(\forall B, B\prime \in Par : B \neq B\prime \Rightarrow B \cap B\prime = \phi) \wedge (\sqcup_{B \in Par} B = A)$.

Definition 6 (Partitioned Event Graph) $PEG_M \triangleq \langle M, P_V \rangle$
is the corresponded Partitioned Event Graph for $M \triangleq < V, E, S, F, C, T, A, B >$, in which P_V stands for partition on vertex set V.

For LP-based model, it is required that Condition 3.1 holds.

Condition 3.1 The class of State Variable set generated from $\{Project(StateVarSet, Set) | Set \in P_V\}$ is a partition of $StateVarSet$, where $Project(StateVarSet, Set) \triangleq \{Project(vertex) | vertex \in Set\}$ is the set of all state variables accessed by the event nodes in Set.

Since the introductions of attribute A and parameter B in Definition 4 are just a notational convenience, and do not bring change to the modeling power of Event Graph formalism(Schruben 1995) In (Savage, Schruben, and Yucesan 2005) the method to remove parameters in EG is described, and in (Ingalls, Morrice, Yucesan, and Whinston 2003), the EG model with canceling edges is transformed into equivalent EG model only with scheduling edges. Therefore, we consider only EG without attributes, parameters, and canceling edges in the discussion of transformation.

The proof of the equivalence between PEG and LP-based model by generating a bidirectional mapping between them is shown in Theorem 1 and Theorem 2.

Theorem 1 Any Logical Process based discrete event model can be described in Event Graph.

PROOF: We map from LP formalism to Event Graph formalism by constructing an EG-based model: $EG_{Set_{LP}} \triangleq \langle V, E, S, F, C, T \rangle$ in which

$$V = R_1(EventSet) \qquad E = R_2(ScheduleRelationship) \qquad S = R_3(StateSpace)$$
$$F \triangleq \cup_{i \in L} F_i \qquad C \triangleq \cup_{i \in L} C_i \qquad T \triangleq \cup_{i \in L} T_i$$

According to the definition of LP paradigm and Event Graph, the congruence relationships between Set_{LP} and $EG_{Set_{LP}}$ are constructed in the following way:
$R_1 : EventSet \Leftrightarrow V$, $R_2 : ScheduleRelationship \Leftrightarrow E$, $R_3 : StateSpace \Leftrightarrow S$.

Since event type belongs and only belongs to a single *LP*, which means that

$$\forall eT \in EventSet \Rightarrow (\exists i \in L \wedge eT \in EventSubset_i) \\ \wedge (\forall i,j \in L \Rightarrow (eT \in EventSubset_i \wedge eT \in EventSubset_j \Rightarrow i = j))$$

Thus there exists a function from *EventSet* to index set *L*.

$$\Gamma \triangleq \{EventSet \rightarrow L | \Gamma(eT) = EventSubset_i \Leftrightarrow eT \in EventSubset_i, eT \in EventSet, i \in L\}$$

By applying curry operation to δ_i^S, the state transformation function for LP_i is obtained:

$$F_i \triangleq \{f_{eT} : S_i \rightarrow S_i | \forall eT \in EventSet, f_{eT} = \delta_i^S(eT), i = \Gamma(eT)\}$$

The corresponding event scheduling function on the edge set of LP_i is defined as:

$$C_i \triangleq \{c_r : S_i \rightarrow \{True, False\} | \forall r \in ScheduleRelationship_i, \\ \forall s \in S_i, c_r(s) = IsTrue(r.to \in \pi_{EventSet}\delta_i^\eta(r.from, s))\}$$

, where $\pi_{EventSet}$ is the projection function from $EventSet \times T$ to $EventSet$. The time delay function on the edge set of LP_i is defined as:

$$T_i \triangleq \{t_r : S_i \rightarrow Time \cup \{\perp\} | \forall r \in ScheduleRelationship_i, \forall s \in S_i \\ t_r(s) = \begin{cases} \pi_{Time}\delta_i^\eta(r.from, s) & c_r(s) = true. \\ \perp & c_r(s) = false. \end{cases} \}$$

Through the above transformation, a Event Graph based model $EG_{Set_{LP}}$ is generated from Set_{LP}.

Define $P_E \triangleq \{EventSubset_i, \forall i \in L\}$, according to the definition of $EventSubset_i$ and Definition 5, P_E is a partition of $EventSet$. On one hand, since $StateVarSet_i$ and $EventSubset_i$ belongs and only belongs to a single LP_i, there is a bidirectional mapping from $P_{StateVarSet}$ to P_E,on the other,

$$\{Project(StateVarSet, x) | x \in P_E\} = \{StateVarSet_i, i \in L\}$$

is a partition of the state variable set *StateVarSet*, the constraint in Definition 6 is satisfied.

The PEG based model constructed from Set_{LP} is $PEG_{Set_{LP}} \triangleq \langle EG_{Set_{LP}}, P_E \rangle$, and we have proved that $PEG_{Set_{LP}}$ is a Partitioned Event Graph based representation of Set_{LP}. Done.

Theorem 2 Any discrete event model in Partitioned Event Graph formalism can be described with Logical Process Paradigm.

PROOF: Noticing that Constraint 3.1 is held by the PEG based model, the construction of the mapping is similar to theorem1. The *LP*s are constructed according to P_V and $\{Project(StateVarSet, x) | x \in P_V\}$. Particularly, $M = \langle V, E, S, F, C, T, A, B \rangle$ corresponds to a discrete event model in event scheduling world view, which is $Model_{LP}$ with $|L| = 1$.

Done.

Theorem 1 and Theorem 2 together prove the existence of bidirectional mapping between LP-based model and PEG based model.

3.4 The Structural Operational Semantics of PEG

The Structural Operational Semantics of PEG are defined on an abstract machine with Timed-Transition System, which is an extended Labelled-Transition Systems.

Definition 7 (Labelled-Transition Systems (LTS)) Labelled-Transition Systems is a tuple $\langle S, s_0, L, \rightarrow \rangle$, in which S stands for the state space, L stands for the Set of all Labels, $\rightarrow \in S \times L \times S$ is the state transition relationship, for example, $s_a \xrightarrow{action} s_b$ means that$(s_a, action, s_b) \in \rightarrow$, $s_0 \in S$ is the initial state.

Definition 8 (Timed-Transition System (TTS)) $TTS = \langle S, s_0, L, D, T \rangle$, where

- S is state space.
- s_0 is initial state.
- L is set of labels,stands for the set of event types.
- D is the set of discrete transitions, e.g. $D \subseteq S \times L \times S$, $s_a \xrightarrow{action} s_b$ stands for $(s_a, action, s_b) \in D$.
- T is the set of time-passage transitions, e.g. $T \in S \times R^+ \times S$, $s_a \xrightarrow{delay} s_b$ stands for $(s_a, delay, s_b) \in T$, in which $s_a, s_b \in S$, $delay \in T$, $action \in D$.

Timed-Transition System contains two kinds of transition: those correspond to R^+ are time-passage transitions, and those else are discrete transitions.

Based on the definitions above, we define the Structural Operational Semantics of EG and PEG. The execution of an EG model is on an abstract machine whose configuration (global state) keeps track of the state of the component and the time of the last transition. The configuration of a Partitioned Event Graph based model is composed from the configuration of all partitions. The time used in the following definition refers to Simulation Time(Fujimoto 2000).

3.4.1 Operational Semantics of EG

The SOS of the EG model is represented by $TTS(EG) = (Config, config_0, L, D, T)$.

- State Space is marked as

$$Config \triangleq STATE \times Time \times \wp(EventSet \times Time) \times (EventSet \times Time) \times PHASE$$

  $state \triangleq \langle s, \tau, \Lambda, Active, Phase \rangle$, $\forall state \in Config$, in which
  - $s \in STATE$, $STATE = EVAL(StateVarSet)$. $EVAL(StateVarSet)$ corresponds to the model's *StateSpace* in LP paradigm. *StateVarSet* stands for the set of state variables in EG model. The type of the state variable v is defined as $dom(v)$. $\eta \in EVAL(StateVarSet)$ is an evaluation function for the *StateVarSet*. $Cond(StateVarSet)$ is the set of bool expressions on *StateVarSet*.
  - $\tau \in Time$ stands for the simulation time, $delay$ stands for the simulation time delay.
  - $\Lambda \in \wp(EventSet \times Time)$ is the set of pending events. Define $\Lambda.head$ as the reference to the event in Λ with the smallest timestamp if the set is empty.
  - $Active \in EventSet \times Time$ is the cursor to the currently executing event if any. $Active.Type \in V \cup \{NULL\}$, $Type$ stands for the Event Type of the cursor, $Type = NULL$ means that there's no event executing currently.
  - $Phase \in PHASE$. in which the event retrieval operation from the pending event set and the execution operation are treated as a single transaction, and thus $PHASE \triangleq \{Elapse, ExeSch\}$
  - Time represents the time based, R^+ is used in this paper.
- $config_0$ is the initial state.
- $L \triangleq V \cup E$, is the union of the edge set and the vertex set of the PEG.
- Transition relationships D and T are defined in the following rules:
  a Event Execution

$$\frac{Phase = Elapse \wedge \Lambda.head.TYPE = EGvertex \wedge \tau = \Lambda.head.timestamp}{\langle s, \tau, \Lambda, Active, Phase \rangle \xrightarrow{EGvertex} \langle s', \tau, \Lambda', Active', Phase' \rangle}$$

where ($s' = f_{EGvertex}(s)$, $Active' = \Lambda.head$, $\Lambda' = \Lambda - \{Active'\}$, $Phase' = ExeSch$, $EGvertex \in V$);

b Event Scheduling

$$\frac{Phase = Scheduling \wedge Active.TYPE = EGvertex_i \wedge \eta(StateVarSet) \Vdash EGedge_{ij}.c}{\langle s, \tau, \Lambda, Active, Phase\rangle \xrightarrow{EGedge_{ij}(c,t)} \langle s, \tau, \Lambda', Active', Phase'\rangle}$$

where ($\Lambda' = \Lambda \cup \{\langle EGvertex_j, \tau + t\rangle\}$, $Active'.TYPE = NULL$, $Phase' = Elapse$, $EGvertex_i$ and $EGvertex_j \in V$, $EGedge_{ij} \in E$);

c Time Elapsing

$$\frac{Phase = Elapse \wedge \tau + delay < \Lambda.head.timestamp}{\langle s, \tau, \Lambda, Active, Phase\rangle \xrightarrow{delay} \langle s, \tau + delay, \Lambda, Active, Phase\rangle}$$

where ($delay \in Time$).

3.4.2 Operational Semantics of PEG

In order to define the SOS of *PEG* based Model, first, we transform the $PEG_M \triangleq \langle M, P_{StateVarSet}, P_E\rangle$ into the equivalent style $PEG \triangleq \{LP_0, \ldots, LP_{n-1}\}$, where

$$LP_i \triangleq \{< StateVarSet_i, EventSubset_i > | i \in L\}, (0 \le i < n).$$

The SOS of the PEG based model is represented as a TTS: $TTS(PEG) =< Config, config_0, L, D, T >$.

Particularly, $\forall state \triangleq \langle s, \tau, \Lambda, Active, Phase\rangle \in Config$, $\Lambda = [\Lambda_1, \ldots, \Lambda_{|L|}]^T$, $s_i \in S_i, 1 \le i \le |L|$, which means that Λ is the composition of pending event sets of all *LP*s, and $\tau \triangleq \tau_{GVT} = Min_{i \in L}(lvt_i)$ is the global simulation time, in which lvt_i is the local simulation time of LP_i.

Transition relationships D and T are defined in the following rules:

a. Event Execution

$$\frac{EGvertex \in EventSubset_i \wedge Phase = Elapse \wedge \Lambda.head.TYPE = EGvertex \wedge \tau = \Lambda.head.timestamp}{\langle \begin{bmatrix}\cdots\\ s_i\\ \cdots\end{bmatrix}, \begin{bmatrix}\cdots\\ lvt_i\\ \cdots\end{bmatrix}, \begin{bmatrix}\cdots\\ \Lambda_i\\ \cdots\end{bmatrix}, Active, Phase\rangle \xrightarrow{EGvertex} \langle \begin{bmatrix}\cdots\\ s_i'\\ \cdots\end{bmatrix}, \begin{bmatrix}\cdots\\ lvt_i\\ \cdots\end{bmatrix}, \begin{bmatrix}\cdots\\ \Lambda_i'\\ \cdots\end{bmatrix}, Active', Phase'\rangle}$$

where ($s' = f_{EGvertex}(s)$, $Active' = \Lambda.head$, $\Lambda' = \Lambda - \{Active'\}$, $Phase' = Scheduling$, $EGvertex \in V$).

b. Time Elapsing

$$\frac{Phase = Elapse \wedge \tau_{GVT} + delay < \Lambda.head.timestamp}{\langle \begin{bmatrix}s_1\\ \cdots\\ s_i\\ \cdots\\ s_{|L|}\end{bmatrix}, \begin{bmatrix}lvt_1\\ \cdots\\ lvt_i\\ \cdots\\ lvt_{|L|}\end{bmatrix}, \begin{bmatrix}\Lambda_1\\ \cdots\\ \Lambda_i\\ \cdots\\ \Lambda_{|L|}\end{bmatrix}, Active, Phase\rangle \xrightarrow{delay} \langle \begin{bmatrix}s_1\\ \cdots\\ s_i\\ \cdots\\ s_{|L|}\end{bmatrix}, \begin{bmatrix}lvt_1 + delay\\ \cdots\\ lvt_i + delay\\ \cdots\\ lvt_{|L|} + delay\end{bmatrix}, \begin{bmatrix}\Lambda_1\\ \cdots\\ \Lambda_i\\ \cdots\\ \Lambda_{|L|}\end{bmatrix}, Active, Phase\rangle}$$

where ($delay \in Time$).

c. Event Scheduling

Depending on whether the vertex of the scheduling edge cross the partition boundary, the external and internal event scheduling are distinguished.

External event scheduling: If $(EGvertex_i \in EventSubset_m \wedge EGvertex_j \in EventSubset_k \wedge k \ne m)$

$$\frac{(Phase = Scheduling) \wedge (Active.TYPE = EGvertex_i) \wedge (\eta(StateVarSet) \Vdash EGedge_{ij}.c)}{\langle \begin{bmatrix}\cdots\\ s_m\\ \cdots\\ s_k\\ \cdots\end{bmatrix}, \begin{bmatrix}\cdots\\ lvt_m\\ \cdots\\ lvt_k\\ \cdots\end{bmatrix}, \begin{bmatrix}\cdots\\ \Lambda_m\\ \cdots\\ \Lambda_k\\ \cdots\end{bmatrix}, Active, Phase\rangle \xrightarrow{EGedge_{ij}(c,t)} \langle \begin{bmatrix}\cdots\\ s_m\\ \cdots\\ s_k\\ \cdots\end{bmatrix}, \begin{bmatrix}\cdots\\ lvt_m\\ \cdots\\ lvt_k\\ \cdots\end{bmatrix}, \begin{bmatrix}\cdots\\ \Lambda_m\\ \cdots\\ \Lambda_k'\\ \cdots\end{bmatrix}, Active', Phase'\rangle}$$

where $(\Lambda'_k = \Lambda_k \cup \{\langle EGvertex_j, lvt_m + t\rangle\}, Active'.TYPE = NULL, Phase' = Elapse, EGvertex_i, EGvertex_j \in V,\ EGedge_{ij} \in E\ )$.

Internal event scheduling: If $EGvertex_i, EGvertex_j \in EventSubset_m$

$$\frac{Phase = Scheduling \wedge Active.TYPE = EGvertex_i \wedge \eta(StateVarSet) \Vdash EGedge_{ij}.c}{\langle \begin{bmatrix} \cdots \\ s_m \\ \cdots \end{bmatrix}, \begin{bmatrix} \cdots \\ lvt_m \\ \cdots \end{bmatrix}, \begin{bmatrix} \cdots \\ \Lambda_m \\ \cdots \end{bmatrix}, Active, Phase\rangle \xrightarrow{EGedge_{ij}(c,t)} \langle \begin{bmatrix} \cdots \\ s_m \\ \cdots \end{bmatrix}, \begin{bmatrix} \cdots \\ lvt_m \\ \cdots \end{bmatrix}, \begin{bmatrix} \cdots \\ \Lambda'_m \\ \cdots \end{bmatrix}, Active', Phase'\rangle}$$

where $(\Lambda'_m = \Lambda_m \cup \{\langle EGvertex_j, lvt_m + t\rangle\}, Active'.TYPE = NULL, Phase' = Elapse, EGvertex_i, EGvertex_j \in V,\ EGedge_{ij} \in E\ )$.

4 WALLCLOCK TIME BASED EXECUTION

The SOS of PEG in Section 3 are strictly sequential, and hence the event execution order in wallclock time strictly follows the nondecreasing order of the simulation time stamp.

In order to study the concurrent execution of LP-based model, the Wallclock Time Execution of TTS is introduced as well as the equivalence relationship based on Local Causality Constraint (LCC)(Fujimoto 2000). First, several definitions are presented.

Definition 9 ($I-$ Trace) $\omega : I \to S$, I represent the closed interval start from time 0. S represents the State Space of the system. in which $\forall t, t'$, if $t \le t'$ then $\omega(t) \xrightarrow{t'-t} \omega(t')$, $\omega.lt$ represents the supremum of I, and $\omega.ft$ represents the infimum of I. $\omega.ls$ represents the final state , and $\omega.fs$ the initial state $\omega(0)$.

For example, $config \xrightarrow{d} config'$ corresponds to the$[0,d]$-Trace, in which$\omega.fs = config$, $\omega.ls = config'$.

Definition 10 (Timed Execution of TTS) The Timed Execution of TTS is an alternately sequence $\gamma = \omega_0 a_1 \omega_1 a_2 \ldots \omega_{n-1} a_n \ldots$. in which ω_i represents I-trace, $a_i$represent event type, and $\omega_i.ls \xrightarrow{a_{i+1}} \omega_{i+1}.fs$. The duration and initial state of γ is marked as $\gamma.lt$ and $\gamma.fs$, respectively. $\gamma.fs = \omega_0.fs, \gamma.ft = \omega_0.ft$. For limited Timed Execution, $\gamma.lt = \omega_n.lt$, $\gamma.ls = \omega_n.ls$. Note the set of all possible Timed Execution of TTS as $execs(TTS)$.

Definition 11 (Wallclock Time Execution of TTS) Wallclock Time Execution of TTS is formed by attaching the Wallclock timestamp $a_i.wct$ to each discrete event type a_i in $\gamma = \omega_0 a_1 \omega_1 a_2 \ldots \omega_{n-1} a_n \ldots$. The actual execution order of TTS in wallclock time is $\gamma_{WC} = \omega_{WC_0} a_{WC_1} \omega_{WC_1} a_{WC_2} \ldots \omega_{WC_{n-1}} a_{WC_n} \ldots$. Define the set of all possible wallclock time execution of TTS as $execs_{WC}(TTS)$.

If each LP adheres to the LCC, then the parallel execution will yield exactly the same results as a sequential execution of the same model, provided that simultaneous events are processed in the same order in both cases(Fujimoto 2000, p. 53).

And hence LCC can be regarded as a selection of all executions without casual error from $execs_{WC}(TTS)$. The set of all wallclock time executions of TTS qualifying LCC is defined as $execs^{LCC}_{WC}(TTS) \subseteq execs_{WC}(TTS)$, and event is defined as $e_i \triangleq \langle a_i, \omega_i.ft\rangle$, with event type $e_i.type = a_i$, and execution time $e_i.st = \omega_i.ft$.

If there 's no casuality between e_i and e_j, which is noted as $e_i \| e_j$, then exchanging places of e_i and e_j in the wallclock time execution, does not alter the result. That is to say, for executions

$\gamma_{WC} = \omega_{WC_0} a_{WC_1} \omega_{WC_1} \ldots a_{WC_i} \omega_{WC_i} \ldots a_{WC_j} \omega_{WC_j} \ldots$

$\gamma'_{WC} = \omega_{WC_0} a_{WC_1} \omega_{WC_1} \ldots a_{WC_j} \omega'_{WC_j} \ldots a_{WC_i} \omega'_{WC_i} \ldots$

The following solutions are held to be true:

$\omega_{WC_j}.fs = \omega'_{WC_i}.fs$, $\gamma_{WC}, \gamma'_{WC} \in execs^{LCC}_{WC}(TTS)$, and $\gamma_{WC} \neq \gamma'_{WC}$.

In summary, even though parallel execution under different parallel time management algorithms can generate different wallclock time executions, given that their time management algorithms adhere to LCC, they result in the same final system state as in the case SOS does in section 3.

5 CONCLUSION AND OUTLOOK

By formalizing the LP-based modeling of PDES, the PEG formalism is a well-defined interpretation to the elements of LP paradigm, and thus serve as an unambiguous and platform-independent description of the LP-based model. The rigorous structural operational semantics shown in this paper, though being sequential, is well adapted to represent timed behavior of LP-based model. PEG provides modelers with a mean for cross platform model transformation and formal analysis to the model's behavior. However, PEG is not designed for high level specifications of discrete event systems, this is because that model for high level specifications can be indeterministic and the occurrence times of events are not defined precisely. The focus of future research will be to develop a more specialized language which only allows to code valid models for the PEG formalism, and a flexible framework for domain specific modeling for PDES, which takes PEG as the semantic specification and incorporates automatic model partitioning.

ACKNOWLEDGMENTS

This work is partly supported by the *China Scholarship Council* (Grant No. 2007U39612), the *National Science Foundation of China (NSF)* (Grant No. 61003075, 61170048, 61170047). The authors would like to show their gratitude to Prof. Adelinde M. Uhrmacher, Dr. Jan Himmelspach, Dr. Roland Ewald from University of Rostock for their continuous help throughout the development of this work.

REFERENCES

Bagrodia, R. L. 1998, April. "Parallel languages for discrete-event simulation models". *IEEE Computational Science and Engineering* 5 (2): 27–38.

Bagrodia, R. L., and W. T. Liao. 1994, April. "Maisie: A Language for the Design of Efficient Discrete-Event Simulations". *IEEE Transactions on Software Engineering* 20 (4): 225–238.

Buss, A. 2002. "Component based simulation modeling with Simkit". Volume 1, 243–249. Los Alamitos, CA, USA: IEEE Computer Society.

Buss, A. H., and P. J. Sanchez. 2002, December. "Building complex models with LEGOs (Listener Event Graph Objects)". *Winter Simulation Conference* 1:732–737.

Carothers, C. D., D. Bauer, and S. Pearce. 2000. "ROSS: A High-Performance, Low Memory, Modular Time Warp System". *Parallel and Distributed Simulation, Workshop on* 0:53+.

Cassandras, C. G., and S. Lafortune. 2008, October. *Introduction to Discrete Event Systems*. 2nd ed. ed. Springer.

Chandy, K. M., and J. Misra. 1979. "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs". *Software Engineering, IEEE Transactions on* SE-5 (5): 440–452.

Cota, B. A., D. G. Fritz, and R. G. Sargent. 1994. "Control flow graphs as a representation language". In *Proceedings of the 26th conference on Winter simulation*, WSC '94, 555–559. San Diego, CA, USA: Society for Computer Simulation International.

Curry, R., C. Kiddle, R. Simmonds, and B. Unger. 2005. "Sequential Performance of Asynchronous Conservative PDES Algorithms". In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, 217–226. Washington, DC, USA: IEEE Computer Society.

Das, S., R. Fujimoto, K. Panesar, D. Allison, and M. Hybinette. 1994. "GTW: a time warp system for shared memory multiprocessors". In *Simulation Conference Proceedings, 1994. Winter*, 1332–1339.

Fujimoto, R. M. 2000. *Parallel and Distributed Simulation Systems*. John Wiley and Sons.

Harel, D. 1987, June. "Statecharts: a visual formalism for complex systems". *Science of Computer Programming* 8 (3): 231–274.

Hybinette, M., E. Kraemer, Y. Xiong, G. Matthews, and J. Ahmed. 2006. "SASSY: a design for a scalable agent-based simulation system using a distributed discrete event infrastructure". In *Proceedings of the 38th conference on Winter simulation*, WSC '06, 926–933: Winter Simulation Conference.

Ingalls, R. G., D. J. Morrice, E. Yucesan, and A. B. Whinston. 2003, January. "Execution Conditions: A Formalization of Event Cancellation in Simulation Graphs". *INFORMS JOURNAL ON COMPUTING* 15 (4): 397–411.

Leye, S., J. Himmelspach, M. Jeschke, R. Ewald, and A. M. Uhrmacher. 2008. "A Grid-Inspired Mechanism for Coarse-Grained Experiment Execution". In *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, DS-RT '08, 7–16. Washington, DC, USA: IEEE Computer Society.

Liu, G., Y. Yao, and B. Liu. 2010, August. "VISICOM: A Component-Based Parallel Discrete Event Modeling Framework". *Advances in System Simulation, International Conference on* 0:158–163.

Liu, J., J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith. 2011. "Parallel Discrete-Event Simulation".

Liu, Q. 2010, Sept. *Algorithms for Parallel Simulation of Large-Scale DEVS and Cell-DEVS Models*. Ph. D. thesis, Carleton University.

Misra, J. 1986, March. "Distributed discrete-event simulation". *ACM Comput. Surv.* 18 (1): 39–65.

Nutaro, J., and H. Sarjoughian. 2004, November. "Design of Distributed Simulation Environments: A Unified System-Theoretic and Logical Processes Approach". *SIMULATION* 80 (11): 577–589.

Oguara, T., D. Chen, G. Theodoropoulos, B. Logan, and M. Lees. 2005. "An Adaptive Load Management Mechanism for Distributed Simulation of Multi-agent Systems". *Distributed Simulation and Real Time Applications, IEEE/ACM International Symposium on* 0:179–186.

Paul, R. J. 1993. "Activity cycle diagrams and the three-phase method". In *Proceedings of the 25th conference on Winter simulation*, WSC '93, 123–131. New York, NY, USA: ACM.

Perumalla, K. 2005, June. "μsik: A Micro-kernel for Parallel/Distributed Simulation Systems". In *Workshop on Parallel and Distributed Simulation*, 59–68. Monterey, CA: IEEE.

Perumalla, K. S., and R. M. Fujimoto. 2001, March. "Interactive parallel simulations with the Jane framework". *Future Gener. Comput. Syst.* 17:525–537.

Peschlow, P., M. Geuer, and P. Martini. 2008, April. "Logical Process Based Sequential Simulation Cloning". *Simulation Symposium, 2008. ANSS 2008. 41st Annual*:237–244.

Rich, D. O., and R. E. Michelsen. 1991. "An assessment of the ModSim/TWOS parallel simulation environment". In *WSC '91: Proceedings of the 23rd conference on Winter simulation*, 509–518. Washington, DC, USA: IEEE Computer Society.

Sargent, R. G. 2004. "Some recent advances in the process world view". In *Proceedings of the 36th conference on Winter simulation*, WSC '04, 293–299: Winter Simulation Conference.

Savage, E. L., L. W. Schruben, and E. Yucesan. 2005, January. "On the Generality of Event-Graph Models". *INFORMS JOURNAL ON COMPUTING* 17 (1): 3–9.

Schruben, L. 1983, November. "Simulation modeling with event graphs". *Commun. ACM* 26 (11): 957–963.

Schruben, L. W. 1995. *Graphical Simulation Modeling and Analysis: Using SIGMA for Windows*. 1st ed. Boston, MA, United States: Course Technology Press.

Schruben, L. W., T. M. Roeder, W. K. Chan, P. Hyden, and M. Freimer. 2003. "Advanced event scheduling methodology:". In *Proceedings of the 35th conference on Winter simulation: driving innovation*, WSC '03, 159–165: Winter Simulation Conference.

Steinman, J. S. 1991, January. "SPEEDES: Synchronous Parallel Environment for Emulation and Discrete Event Simulation". In *Proceedings of the SCS Multiconference on Advances in Parallel and Distributed Simulation*, 95–101.

Tapus, C. 2006. *Distributed speculations: providing fault-tolerance and improving performance*. Ph. D. thesis, California Institute of Technology.

Wainer, G. A., and N. Giambiasi. 2002, April. "N-dimensional Cell-DEVS Models". *Discrete Event Dynamic Systems* 12 (2): 135–157.

Wieland, F. 1997. "The threshold of event simultaneity". In *Proceedings of the eleventh workshop on Parallel and distributed simulation*, PADS '97, 56–59. Washington, DC, USA: IEEE Computer Society.

Wilmarth, T. L. 2005. *POSE: Scalable General-purpose Parallel Discrete Event Simulation*. Ph. D. thesis, University of Illinois at Urbana-Champaign.

Yücesan, E., and L. Schruben. 1992, January. "Structural and behavioral equivalence of simulation models". *ACM Trans. Model. Comput. Simul.* 2 (1): 82–103.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000, January. *Theory of Modeling and Simulation, Second Edition*. 2 ed. Academic Press.

AUTHOR BIOGRAPHIES

BING WANG is a researcher in Software Division, Beijing Institute of Systems Engineering. He received the B.S., M.S., and PHD. degrees in 2003, 2006, 2011 respectively, at School of Computer, National University of Defense Technology, P.R. China. His current research interests are parallel discrete-event simulation, modeling methodology for software systems, and experimental algorithm evaluation. He can be contacted by email at wangbing.nudt@gmail.com

BO DENG is a professor and director of the Software Division, Beijing Institute of Systems Engineering. He received his B.S. and Ph.D. degrees in 1995 and 2000 respectively at National University of Defense Technology, P.R. China. His research interests include distributed computing, software architecture, and software assurance. His e-mail address is bodeng@vip.tom.com.

FEI XING is a Ph.D student in Institute of Mathematics and Computer Sciences, University of Heidelberg. He received the B.S. degree in 2008 at School of Mathematical Sciences, Anhui University, P.R. China. Then he received his M.S. degree at School of Computer, National University of Defense Technology, P.R. China. His current research interests are Monte Carlo Simulation method, modeling and simulation for tumor drug therapy. He can be contacted by email at xingfei.nudt@gmail.com

YIPING YAO is a professor in School of Computer, National University of Defense Technology, R.R. China. In this school, he received his M.S. and Ph.D. degrees in 1987 and 2004 respectively. He received his B.S. degree in computer science from Huazhong University of Science and Technology in 1985. His research interests include modeling methodology for complex systems, high performance simulation, distributed simulation, and virtual reality. His e-mail address is ypyao@nudt.edu.cn and his web page is http://yhsim.nudt.edu.cn/yaoyiping.htm.