University of Udine

Lumpabilities in PEPA

Riccardo Romanello[1]

[1] UNIVERSITY OF UDINE, ITALY

June 6, 2023

Stochastic Process

A *stochastic process* is a random process evolving with time.

Definition 1 (Stochastic Process)

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $(\mathcal{S}, \Sigma)$ be a measurable space and T be a totally ordered set. A *stochastic process* $\mathcal{X}$ is defined as:

$$\mathcal{X} = \{X_t \mid t \in T\}$$

$\mathcal{S}$ is the *state space* of the process and we say that $\mathcal{X}$ is in state $s \in \mathcal{S}$ at time $t \in T$ if $\mathcal{X}(t) = X_t = s$.

Discrete Time vs Continuous Time

Stochastic processes can be categorized into two different classes:

- *Discrete Time* if T is discrete. State changes with fixed frequency and transitions all take the same time
- *Continuous Time* if T is continuous. In this case transitions can take different amounts of times.

Markov Property

Definition 2 (Markov Property)

Let $\mathcal{X}$ be a stochastic process and let t be the current (discrete) time.
We say that $\mathcal{X}$ has the *Markov property*, if for each future time $y > t$, the distribution of X_y does not depend on the past history $P = \{X_u \mid u < t\}$.

Roughly speaking, the future depends only on the current state and not on the past.
We will say that $\mathcal{X}$ is a *Markov Process* if and only if $\mathcal{X}$ enjoys the *Markov property*.

Markov Chain

Definition 3 (Markov chain)

A *Markov Chain* is a Markov process $\mathcal{X}$ indexed by time T such that the state space S is discrete.

As for stochastic processes, also for Markov chain we can split between *Continuous Time* (CTMC) and *Discrete Time* (DTMC) according to the properties of T.

DTMC and CTMC for everyone

- We can thinkg of a DTMC as a process being in a certain state s_t at time t
- At time $t+1$, the process goes from s_t to some s' with probability $p_{s_t,s'}$
- In CTMC the transition between state is not *instantaneous*
- It starts in state s_0 and then moves towards state s_1 with probability p_{s_0,s_1}
- Such movement has a delay that is exponentially distributed with parameter $q_{s_0,s_1} = r_{s_0} \cdot p_{s_0,s_1}$ — also known as *transition rate*.

Markov Properties—reformulated

For CTMC the *Markov property* can be restated as:

$$\mathbb{P}(X_{t+y} = s' \mid X_u = s_u : u \leq t) = \mathbb{P}(X_{t+y} = s' \mid X_t = s_t)$$

On the other hand, for DTMC it becomes:

$$\mathbb{P}(X_t = s_t \mid \bigwedge_{i=1}^{t-1} X_{t-1} = s_{t-1}) = \mathbb{P}(X_t = s_t \mid X_{t-1} = s_{t-1})$$

Time-homogeneous Markov Chains

We are interested in Markov Chains for which probabilities do not change over time.

Definition 4 (Time-homogeneous DTMC)

A DTMC $\mathcal{X}$ is *time-homogeneous* if for all states $s, s' \in \mathcal{S}$ and all times $t, t' \in T$ it holds that

$$p_t(s, s') = p_{t'}(s, s')$$

Definition 5 (Time-homogeneous CTMC)

Similarly, a CTMC $\mathcal{X}$ is *time-homogeneous* if for all states $s, s' \in \mathcal{S}$ and all times $t, t' \in T$ the following holds

$$\mathbb{P}(X_{t+y} = s' \mid X_t = s) = \mathbb{P}(X_{t'+y} = s' \mid X_{t'} = s) = p_y(s, s')$$

Representing Markov Chains

- A DTMC is described using a *transition probability matrix* P where $P_{i,j} = p_{i,j}$ for all i, j
- A CTMC is described using a *infinitesimal generator matrix* Q where $Q_{i,j} = q_{i,j}$ for all $i \neq j$ and $Q_{i,i} = r_i$ for all i

Steady State

- With Markov Chains, we usually describe reactive systems
- We are often interested in properties for the long run
- One property could be the probability of the system to be in a certain state after some amount of time
- Probability of being in state s' at time t depends on the initial state s
- This dependency disappears as time goes on, for the markov property

Given a time-homogeneous Markov chain, we can express this fact by taking the limit of such probabilities:

$$\lim_{t \to \infty} \mathbb{P}(X_t = s' \mid X_0 = s) = \pi(s')$$

Encoding stochastic processes

- The description of stochastic processes via Markov chains is a cumbersome and often unfeasible activity
- Moreover, it is hard to check if the stochastic process is correct or not
- Adding an abstraction layer by using a compact representation that is easy to produce and verify, can be a way to address both the issues
- Stochastic process algebras fit this role of *additional layer*
- We adopt PEPA as process algebra to model stochastic processes.

Performance Evaluation Process Algebra

- A PEPA model consists of a set of components that engage either individually or cooperatively in activities
- Components represent identifiable units, and can be either atomic or composed
- Activities capture actions performed by the components. Every activity is associated with an action type
- Activities are not instantaneous. The probability that an activity a with rate r happens within a period of time of length t is given by $F_a(t) = 1 - e^{-rt}$
- Hence, an activity with action type α and rate r is completely defined by the pair (α, r).

PEPA syntax

The PEPA language provides very simple combinators that can be used to compose the components and describe their activities.
The grammar is presented with a distinction between parallel and sequential components:

$$P ::= P \underset{L}{\bowtie} P \mid P/L \mid S$$
$$S ::= (\alpha, r).S \mid S + S \mid A$$

PEPA race condition

- An activity is enabled for a component P if P can immediately start that activity
- A PEPA components may have more than one activity enabled
- We assume that when a component reaches a certain state all the enabled activities start to take place
- Despite that, only the first activity that is completed takes effect — the other ones are aborted
- The set of enabled activities for a PEPA component C is denoted by $\mathcal{A}ct(C)$.

PEPA operational semantics

- PEPA semantics rules induce a *labelled transition system* $(S, T, \{\xrightarrow{t} \mid t \in T\})$
- S is the set of possible configurations of the components
- T is the set of the labels
- The transition relation represents the completed activities.

PEPA operational semantics

Cooperation

$$\frac{P \xrightarrow{(\alpha,r)} P'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} P' \underset{L}{\bowtie} Q}(\alpha \notin L) \qquad \frac{Q \xrightarrow{(\alpha,r)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,r)} Q' \underset{L}{\bowtie} Q}(\alpha \notin L)$$

$$\frac{P \xrightarrow{(\alpha,r_1)} P', Q \xrightarrow{(\alpha,r_2)} Q'}{P \underset{L}{\bowtie} Q \xrightarrow{(\alpha,R)} P' \underset{L}{\bowtie} Q'}(\alpha \in L) \quad R = \frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(Q)} \min(r_\alpha(P), r_\alpha(Q))$$

Labelled Transition System - LTS

Definition 6 (Derivative set)

Let C be a PEPA component. The *derivative set* of C, denoted by ds(C) is the smallest set of PEPA components such that:

- $C \in \mathrm{ds}(C)$
- if $C_i \in \mathrm{ds}(C)$ and there exists an activity $a \in Act(C)$ such that $C_i \xrightarrow{a} C_j$, then $C_j \in \mathrm{ds}(C)$.

Labelled Transition System - LTS - II

Definition 7 (Derivation graph)

Let C be a PEPA component. We denote with $\mathcal{D}(C)$ the *derivation graph* of C with nodes $\mathrm{ds}(C)$.
Edge (C_i, C_j, a) exists if and only there exists an inference tree leading to $C_i \xrightarrow{a} C_j$.
Let $C_i, C_j \in \mathrm{ds}(C)$ be two nodes, then:

$$q(C_i, C_j, \alpha) = \sum_{C_i \xrightarrow{(\alpha,r)} C_j} r$$

Moreover, we define $q(C_i, C_j) = \sum_{\alpha \in A} q(C_i, C_j, \alpha)$ as the *total transition rate*.

From LTS to CTMC

Definition 8

Let C be a finite PEPA model, its *underlying Markov chain* is the stochastic process $\mathcal{X}(t)$ whose states are elements of $\mathrm{ds}(C)$, and whose transitions are given by the edges of $\mathcal{D}(C)$ with rates given by the total transition rates.

Lumping of Stochastic processes

Before introducing equivalences over PEPA components, we must define what makes two states of a stochastic process equivalent.

Definition 9 (strong lumpability)

Let $\mathcal{X}(t)$ be a CTMC with state space $\mathcal{S}$ and let $\sim$ be an equivalence relation over $\mathcal{S}$. We say that $\mathcal{X}(t)$ is *strongly lumpable* with respect to $\sim$, if it induces a partition on the state space $\mathcal{S}$ such that for any equivalence classes $S_i, S_j \in \mathcal{S}/\sim$ with $i \neq j$ and for $s, s' \in S_i$:

$$\sum_{s'' \in S_j} q(s, s'') = \sum_{s'' \in S_j} q(s', s'')$$

Strongly Lumped Markov chain

Definition 10 (strongly lumped CTMC)

Let $\mathcal{X}(t)$ be a CTMC with a state space $\mathcal{S}$ and let $\sim$ be a strong lumping. Then the lumped CTMC $\tilde{\mathcal{X}}(t)$ has $\mathcal{S}/\sim$ as set of states, and the transition rates are given by:

$$q(S, S') = \sum_{s' \in S'} q(s, s')$$

Exact Lumped Markov chain

- Strong lumpability is not the only notion of lumpability defined over stochastic processes
- Exact lumpability is another example
- It differs from strong lumpability because it defines two states as exactly lumpable if the rates *into* them are the same from any other class
- The exactly lumped CTMC is defined in the same way as the strongly lumped one
- If $\sim$ is both an *exact* and a *strong* lumpability, then it is called a *strict* lumpability.

Lumping of PEPA components

- We can now move to the definition of equivalences between PEPA components
- It will induce a partitioning of the underlying CTMC
- Equivalences oves PEPA components must take into account also action types
- Equivalences over PEPA components will be stricter than equivalences over stochastic processes.

Lumpable relation

Definition 11 (Lumpable relation)

Let $\mathcal{R}$ be a relation over PEPA components. We say that $\mathcal{R}$ is a *lumpable relation* if, for any PEPA component P, the quotient $\mathrm{ds}(P)/\mathcal{R}$ induces an equivalence relation over the state space of the underlying CTMC of P that is a strong lumping.

Definition 12 (Unioun closure)

Let I be a set of indices and $\mathcal{R}_i$ be a lumpable relation for all $i \in I$. Then the union:

$$\mathcal{R} = \cup_{i \in I} \mathcal{R}_i$$

is also a lumpable relation.

Lumpable bisimulation

Definition 13 (Lumpable bisimulation)

Let $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ be an equivalence relation over PEPA components. We say that $\mathcal{R}$ is a *lumpable bisimulation* if for every components P, Q such that $P\mathcal{R}Q$, then for all $\alpha \in \mathcal{A}$ and for all $S \in \mathcal{C}/\mathcal{R}$ such that:

- either $\alpha \neq \tau$
- or $\alpha = \tau$ and $P, Q \notin S$

it holds that $q(P, S, \alpha) = q(Q, S, \alpha)$.

Proportional Lumpability

Definition 14 (Proportional Lumpability)

Let $\mathcal{X}(t)$ be a CTMC with state space $\mathcal{S}$ and $\sim$ be an equivalence relation over $\mathcal{S}$. We say that $\mathcal{X}(t)$ is *proportionally lumpable* with respect to $\sim$ if there exists a function κ from $\mathcal{S}$ to $\mathbb{R}^+$ such that $\sim$ induces a partition on the state space of $\mathcal{X}(t)$ satisfying the property that for any equivalence classes $S_i, S_j \in \mathcal{S}/\sim$ with $S_i \neq S_j$ and $s, s' \in S_i$

$$\frac{\sum_{s'' \in S_j} q(s, s'')}{\kappa(s)} = \frac{\sum_{s'' \in S_j} q(s', s'')}{\kappa(s)}$$

Eclipse plugin

- PEPA provides an Eclipse plug-in freely downloadable
- The plug-in allows the user to model concurrent processes using the PEPA stochastic process algebra inside the Eclipse Editor
- After that, the state space can be derived and various performances can be obtained.

High level design

The general steps that the plug-in performs on an input file are:

1. Parsing the PEPA file
2. Exploration of the state space according to PEPA's semantics
3. Derivation of the CTMC model
4. Compute performance measures over the model.

On the CTMC derivation

We now explain the actual steps performed by PEPA plug-in to derive the CTMC model:

1. Derivation of the LTS model
2. Aggregation of the LTS according to the chosen aggregation algorithm
3. Derivation of the aggregated CTMC model

Partition Refinement Algorithms

The aggregation algorithms are applied directly to the LTS model.
The classes used to handle the generated partition are the following:

- An interface `PartitionBlock` is provided to handle a single block of a partition
- The `Partition` class is used to handle a partition refinement algorithm in its completeness
- `PartitionBlock` has been implemented within two classes: `LinkedPartitionBlock` and `ArrayPartitionBlock`
- `LinkedPartitionBlock` allows to store a block with a linked list
- `ArrayPartitionBlock` uses an array instead.

The Labelled Transition System

Aggregation algorithms are applied to LTS.

- An interface `LTS` has been provided
- It has been subsequently implemented in the `LTSModel` class
- The state space exploration is done using the `TextSpaceExplorer` class. The output is a multi-LTS semantics of the input PEPA model
- Such structure is then used to initialize an `LTSModel` which is then aggregated.

Aggregation algorithms

- An interface `AggregationAlgorithm` is provided
- Every time a new aggregation algorithm has to be implemented, it must implement such interface
- Each aggregation algorithm is agnostic of the rest of the plug-in. It refers only to the partition refinement data structure and the LTS

Aggregated state space

Aggregation algorithms compute a partition of the state space, that has to be aggregated.

`AggregationStateSpaceBuilder` class performs the following steps to fulfill such goal:

- A state space exploration is performed
- The LTS is build from such visit
- Aggregated partition is obtained via the selected aggregation algorithm
- The state space is aggregated.

Aggregating a single block

- Each block of a partition is aggregated into a single state
- Let B, B' be two blocks and let $s \in B, s' \in B'$ be two states in such blocks
- If there is a transition labelled α between s and s', then there is a transition labelled α between B and B'.
- The rate of the α transition between B and B' is:

$$\sum_{s \in B} \sum_{s' \in B'} q(s, s', \alpha)$$