An Aggregation Technique For Large-Scale PEPA Models With Non-Uniform Populations

Alireza Pourranjbar, Jane Hillston
School of Informatics, University of Edinburgh

10th December 2013

Overview

In this paper we present an aggregation technique suitable for handling compositional models where we have different populations of entities interacting.

Overview

In this paper we present an aggregation technique suitable for handling compositional models where we have different populations of entities interacting.

In particular we consider the case where some populations are large but crucially some entities are only present in low copy numbers.

Overview

In this paper we present an aggregation technique suitable for handling compositional models where we have different populations of entities interacting.

In particular we consider the case where some populations are large but crucially some entities are only present in low copy numbers.

Such models are difficult to represent with explicit representation of the state space due to the large populations, but are not suitable for fluid approximation due to the small populations.

Overview

In this paper we present an aggregation technique suitable for handling compositional models where we have different populations of entities interacting.

In particular we consider the case where some populations are large but crucially some entities are only present in low copy numbers.

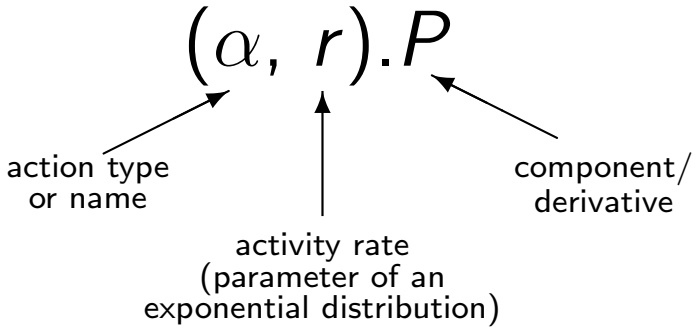
Such models are difficult to represent with explicit representation of the state space due to the large populations, but are not suitable for fluid approximation due to the small populations.

We assume that the models are constructed using the stochastic process algebra, PEPA.

Overview

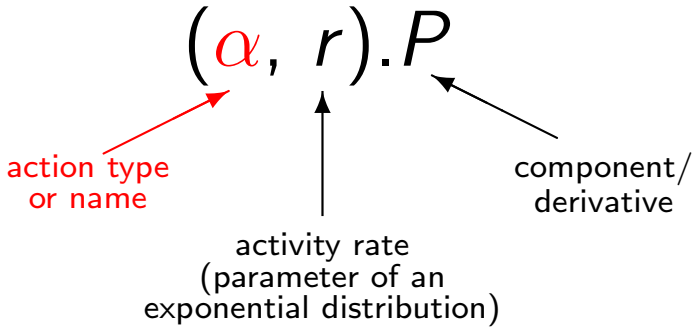
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



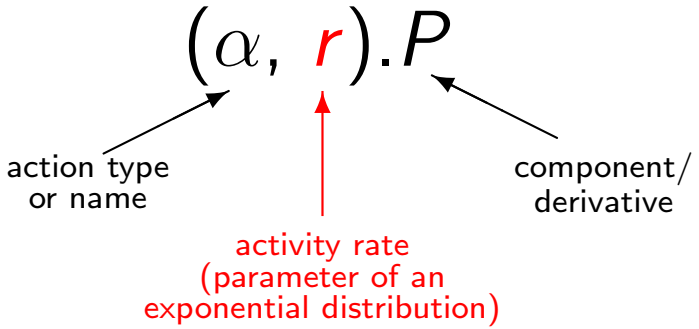
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



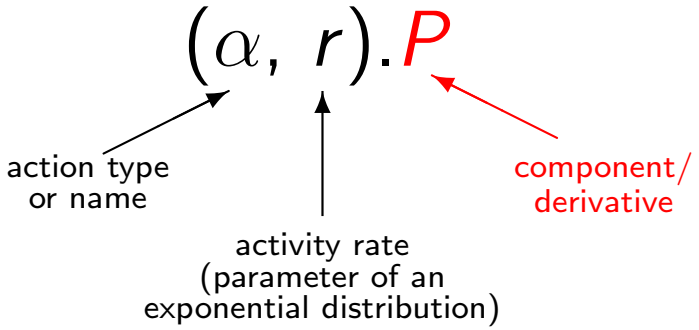
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



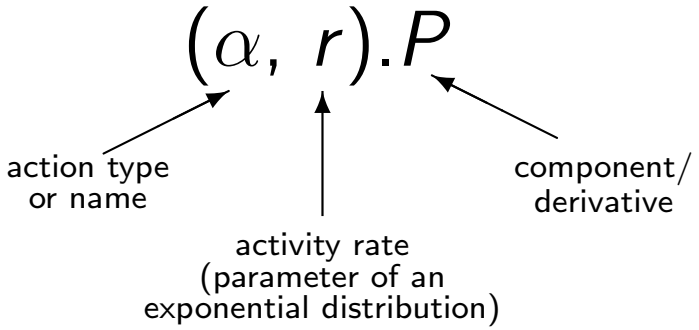
Stochastic Process Algebra

- Models are constructed from components which engage in activities.



Stochastic Process Algebra

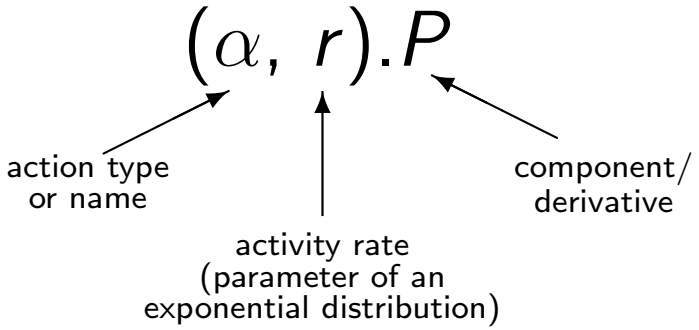
- Models are constructed from components which engage in activities.



- The language is used to generate a CTMC for performance modelling.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

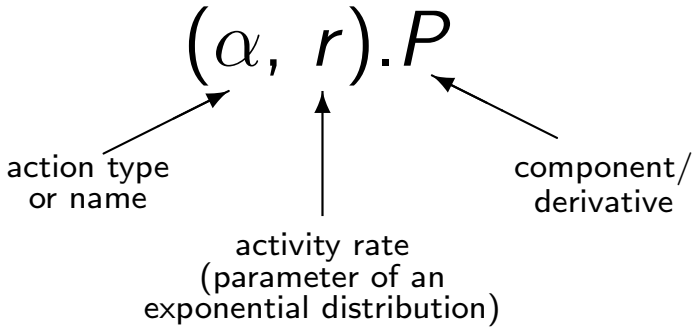


- The language is used to generate a CTMC for performance modelling.

SPA
MODEL

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

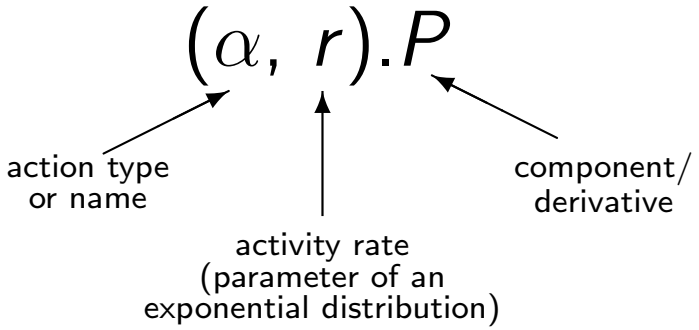


- The language is used to generate a CTMC for performance modelling.

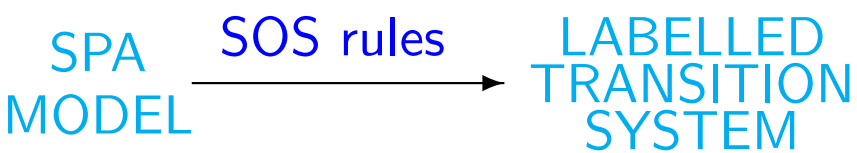
SPA MODEL $\xrightarrow{\text{SOS rules}}$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

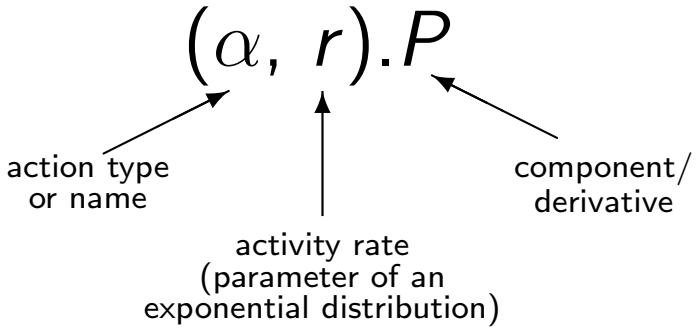


- The language is used to generate a CTMC for performance modelling.



Stochastic Process Algebra

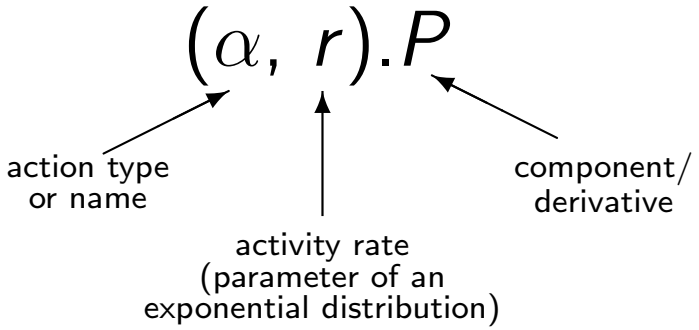
- Models are constructed from components which engage in activities.



- The language is used to generate a CTMC for performance modelling.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.



- The language is used to generate a CTMC for performance modelling.

PEPA

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

PEPA

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

PREFIX: $(\alpha, r).S$ designated first action

PEPA

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

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components

PEPA

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

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components
CONSTANT:	$A \stackrel{def}{=} S$	assigning names

PEPA

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

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components
CONSTANT:	$A \stackrel{def}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ individual actions
		$\alpha \in L$ shared actions

PEPA

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

PREFIX:	$(\alpha, r).S$	designated first action
CHOICE:	$S + S$	competing components
CONSTANT:	$A \stackrel{def}{=} S$	assigning names
COOPERATION:	$P \underset{L}{\bowtie} P$	$\alpha \notin L$ individual actions
		$\alpha \in L$ shared actions
HIDING:	P/L	abstraction $\alpha \in L \Rightarrow \alpha \rightarrow \tau$

Building Models of Large-Scale Systems With PEPA

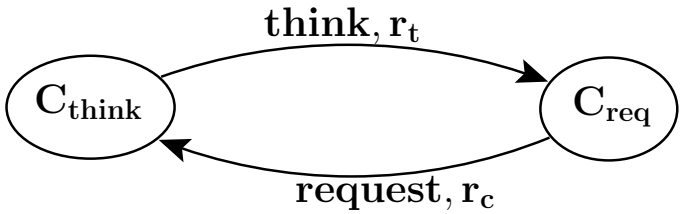
We use the grouped PEPA syntax[1] to express our large-scale PEPA models:

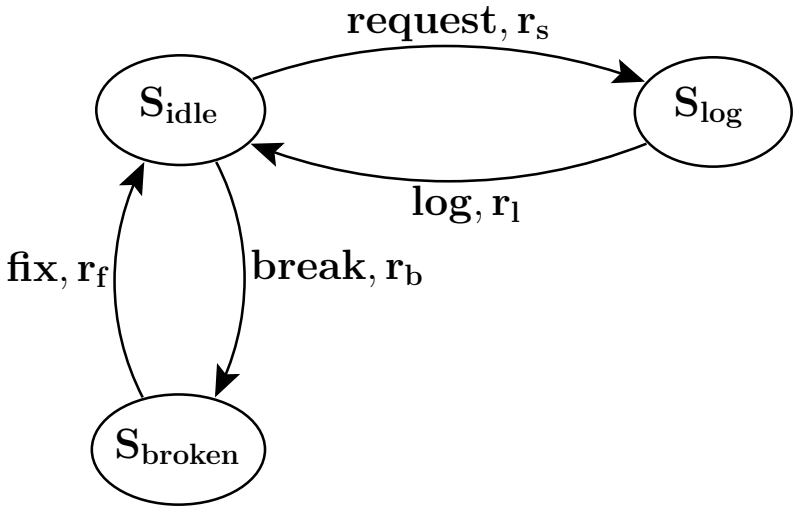
$$\begin{aligned} S &= (\alpha, r).S \mid S + S \mid C_s \qquad & P &= P \underset{L}{\bowtie} P \mid S \\ D &= D \parallel D \mid P & M &= M \underset{L}{\bowtie} M \mid Y\{D\} \end{aligned}$$

1. S : *sequential* component (essentially, a state machine)
2. C_s : constant used to name a component
3. P : the composition of sequential components
4. D : a group of sequential components
   - We only consider simple groups
5. M : a grouped PEPA model

[1] R.A.Hayden and J.T.Bradley, *A fluid analysis framework for a Markovian process algebra*, in TCS. 411(22-24), 2010

Building Models With PEPA - A Client-Server System





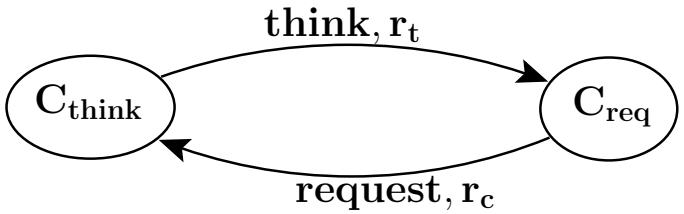
$$C_{think} \stackrel{def}{=} (think, r_t).C_{req}$$
$$C_{req} \stackrel{def}{=} (req, r_c).C_{think}$$

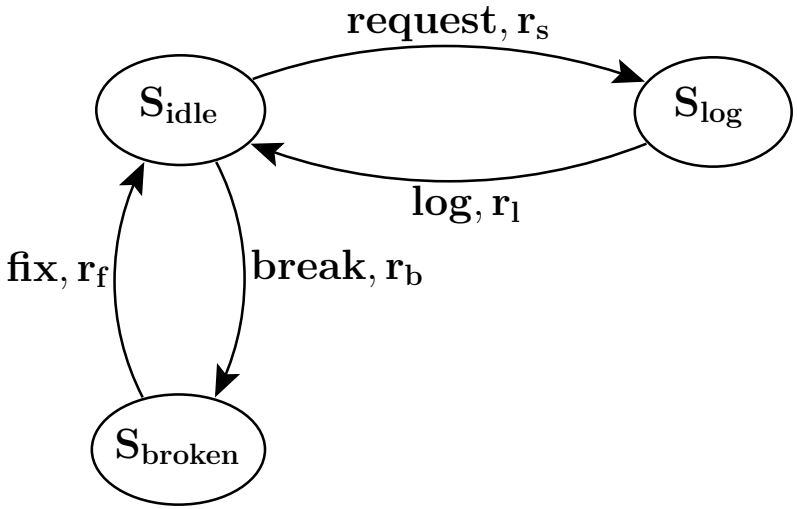
$$S_{idle} \stackrel{def}{=} (req, r_s).S_{log} + (brk, r_b).S_{broken}$$
$$S_{log} \stackrel{def}{=} (log, r_l).S_{idle}$$
$$S_{broken} \stackrel{def}{=} (fix, r_f).S_{idle}$$

$$CS \stackrel{def}{=} Servers\ \{\ S_{idle}[20]\ \} \underset{\{req\}}{\bowtie} Clients\ \{\ C_{think}[1000]\ \}$$

(In general, $P[n] = P \parallel P \parallel \ldots \parallel P$ (n times).)

Building Models With PEPA - A Client-Server System





$$C_{think} \stackrel{def}{=} (think, r_t).C_{req}$$
$$C_{req} \stackrel{def}{=} (req, r_c).C_{think}$$

$$S_{idle} \stackrel{def}{=} (req, r_s).S_{log} + (brk, r_b).S_{broken}$$
$$S_{log} \stackrel{def}{=} (log, r_l).S_{idle}$$
$$S_{broken} \stackrel{def}{=} (fix, r_f).S_{idle}$$

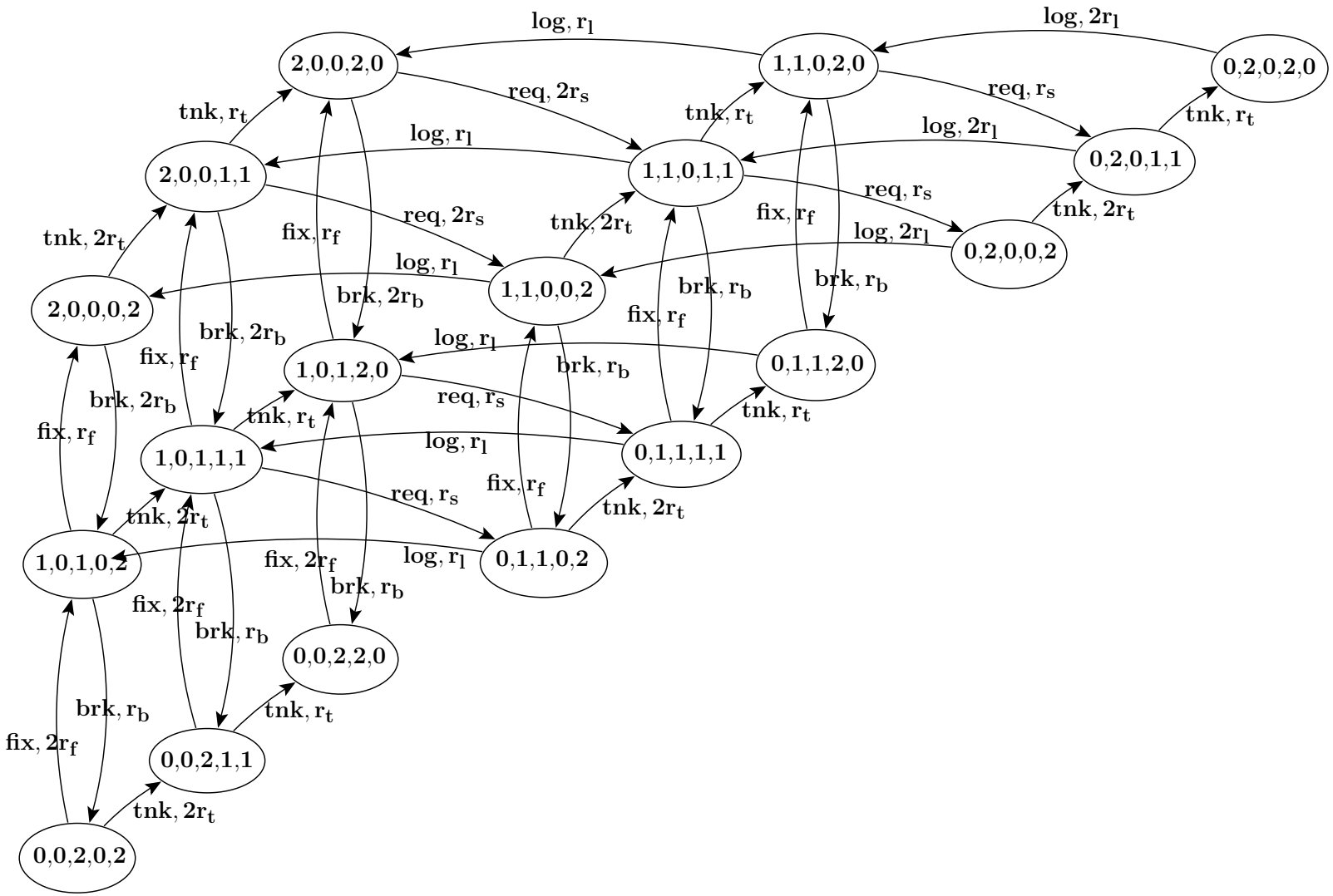
$$CS \stackrel{def}{=} Servers \{ S_{idle}[20] \} \underset{\{req\}}{\bowtie} Clients \{ C_{think}[1000] \}$$

(In general, $P[n] = P \parallel P \parallel \ldots \parallel P$ (n times).)

State Space Representation of Large Scale PEPA Models

- In order to take advantage of the symmetry within a large scale model and lumpability we use a state representation based on a counting abstraction: the state of each group is a count of how many instances within the group are within each local state.
- The rates of the individual and shared activities are defined as functions of the counters.
- Example: at any given time t, the state of the client-server system is captured by a vector $\xi_{CS} = \langle S_i, S_l, S_b, C_r, C_t \rangle$.

State Space Representation of Client-Server system (2 servers & 2 clients)



Markovian Analysis of PEPA Models

- For PEPA, a set of structural operational semantics supports the derivation of the model's underlying population CTMC.
- The CTMC may be analysed for finding the model's transient/equilibrium behaviour.
- Markovian analysis leads to detailed information about the model's behaviour:
  - The complete set of states
  - The complete joint probability distribution of the CTMC
  - The complete probability distribution of the reward random variables
- The problem of state space explosion hampers this process even after using the counting abstraction.

Alternative solution techniques for large scale models

For large scale models, even with counting abstraction aggregation, Markovian analysis becomes infeasible due to state space explosion.

Alternative solution techniques for large scale models

For large scale models, even with counting abstraction aggregation, Markovian analysis becomes infeasible due to state space explosion.

- Stochastic simulation:

Alternative solution techniques for large scale models

For large scale models, even with counting abstraction aggregation, Markovian analysis becomes infeasible due to state space explosion.

- Stochastic simulation:
  - Computationally expensive

Alternative solution techniques for large scale models

For large scale models, even with counting abstraction aggregation, Markovian analysis becomes infeasible due to state space explosion.

- Stochastic simulation:
  - Computationally expensive
  - Intrinsic error

Alternative solution techniques for large scale models

For large scale models, even with counting abstraction aggregation, Markovian analysis becomes infeasible due to state space explosion.

- Stochastic simulation:
  - Computationally expensive
  - Intrinsic error
- Fluid flow approximation (mean-field method): treating the counters as continuous variables governed by a set of ordinary differential equations.

Alternative solution techniques for large scale models

For large scale models, even with counting abstraction aggregation, Markovian analysis becomes infeasible due to state space explosion.

- Stochastic simulation:
  - Computationally expensive
  - Intrinsic error
- Fluid flow approximation (mean-field method): treating the counters as continuous variables governed by a set of ordinary differential equations.
  - Convergence ensured as populations uniformly scale to infinity.

Alternative solution techniques for large scale models

For large scale models, even with counting abstraction aggregation, Markovian analysis becomes infeasible due to state space explosion.

- Stochastic simulation:
  - Computationally expensive
  - Intrinsic error
- Fluid flow approximation (mean-field method): treating the counters as continuous variables governed by a set of ordinary differential equations.
  - Convergence ensured as populations uniformly scale to infinity.
  - Empirically, shown to be sufficient for models with uniformly large populations (without getting to infinity)

Alternative solution techniques for large scale models

For large scale models, even with counting abstraction aggregation, Markovian analysis becomes infeasible due to state space explosion.

- Stochastic simulation:
  - Computationally expensive
  - Intrinsic error
- Fluid flow approximation (mean-field method): treating the counters as continuous variables governed by a set of ordinary differential equations.
  - Convergence ensured as populations uniformly scale to infinity.
  - Empirically, shown to be sufficient for models with uniformly large populations (without getting to infinity)
  - Efficient

Large-Scale Models with Non-Uniform Populations (LSNP)

- Unfortunately, convergence of fluid approximation is based on assumption that all populations scale consistently and not all models fit this pattern.
- For example in resource-bound computer and communication networks often involve interaction of small population of resources and relatively larger populations of resource users.
- In these cases the stochastic dynamics of the small populations can significantly affect the model's global behaviour preventing it from being close to a deterministic limit.

LSNP Models, Small and Large Groups

We consider PEPA models in which some groups have a large number of instances — large groups — but some also only have a few instances — small group.

LSNP Models, Small and Large Groups

We consider PEPA models in which some groups have a large number of instances — large groups — but some also only have a few instances — small group.

We assume that the modeller has specified a partition on set $\mathcal{G}(\mathbb{M})$ specifying $\mathcal{G}_s(\mathbb{M})$ and $\mathcal{G}_l(\mathbb{M})$:

$$\mathcal{G}(\mathbb{M}) = \mathcal{G}_l(\mathbb{M}) \cup \mathcal{G}_s(\mathbb{M}).$$

LSNP Models, Small and Large Groups

We consider PEPA models in which some groups have a large number of instances — large groups — but some also only have a few instances — small group.

We assume that the modeller has specified a partition on set $\mathcal{G}(\mathbb{M})$ specifying $\mathcal{G}_s(\mathbb{M})$ and $\mathcal{G}_l(\mathbb{M})$:

$$\mathcal{G}(\mathbb{M}) = \mathcal{G}_l(\mathbb{M}) \cup \mathcal{G}_s(\mathbb{M}).$$

Therefore, the state vector can be partitioned:

$$\boldsymbol{\xi} = \langle \boldsymbol{\xi}_s, \boldsymbol{\xi}_l \rangle$$

Aggregation Condition

The proposed approach depends on non-uniform population models which satisfy the following aggregation condition:

> For any shared activity, synchronised between one or more large groups and one or more small groups, the rate of the shared activity, should be completely decided by the small groups.

- In any synchronisation on a shared activity of type α, if both small and large groups are involved, then all instances in those large groups need to undertake α passively.
- The condition can be checked automatically by syntactic analysis of the process terms.

Aggregation Condition - Example

Assuming that $\mathcal{G}_s(CS) = Servers$ and $\mathcal{G}_l(CS) = Clients$:

$$
\begin{array}{rcl}
C_{think} & \stackrel{def}{=} & (think, r_t).C_{req} \\
C_{req} & \stackrel{def}{=} & (req, \top).C_{think} \\
\\
S_{idle} & \stackrel{def}{=} & (req, r_s).S_{log} + (brk, r_b).S_{broken} \\
S_{log} & \stackrel{def}{=} & (log, r_l).S_{idle} \qquad\qquad S_{broken} \stackrel{def}{=} (fix, r_f).S_{idle} \\
\\
CS & \stackrel{def}{=} & Servers\ \{\ S_{idle}[20]\ \} \underset{\{req\}}{\bowtie} Clients\ \{\ C_{think}[1000]\ \}
\end{array}
$$

In any state of the system, if *req* is enabled, the rate of the shared activity depends only on the state of the servers.

Structural Properties of the State Space: $\overrightarrow{\mathcal{A}}_s^*(\mathbb{M})$

A model which satisfies the syntactic aggregation condition exhibits important structural properties in its state space.

- There are some action types, $\overrightarrow{\mathcal{A}}_s^*(\mathbb{M})$, which only involve the small populations.

Structural Properties of the State Space: $\overrightarrow{\mathcal{A}}_s^*(\mathbb{M})$

A model which satisfies the syntactic aggregation condition exhibits important structural properties in its state space.

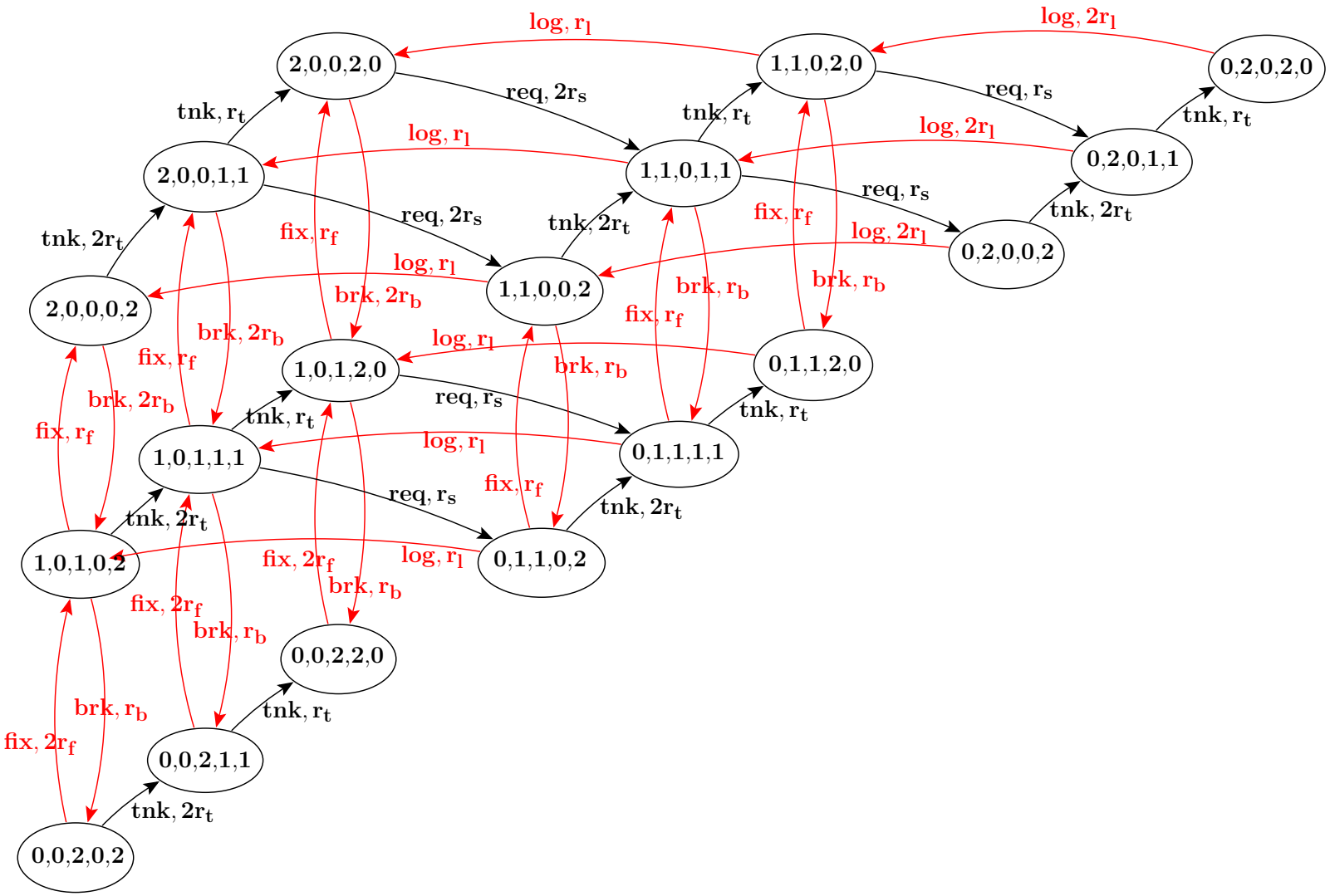
- There are some action types, $\overrightarrow{\mathcal{A}}_s^*(\mathbb{M})$, which only involve the small populations.
- Transitions of these types only change $\boldsymbol{\xi}_s$ and have no effect on $\boldsymbol{\xi}_l$.

Structural Properties of the State Space: $\overrightarrow{\mathcal{A}}_s^*(\mathbb{M})$

A model which satisfies the syntactic aggregation condition exhibits important structural properties in its state space.

- There are some action types, $\overrightarrow{\mathcal{A}}_s^*(\mathbb{M})$, which only involve the small populations.
- Transitions of these types only change $\boldsymbol{\xi}_s$ and have no effect on $\boldsymbol{\xi}_I$.
- Moreover the rate of these transitions only depend on the configuration of the small populations.

$\overrightarrow{\mathcal{A}}_s^*(CS) = \{log, brk, fix\}$



Structural Properties of the State Space: $\overrightarrow{\mathcal{A}}^*_{sl}(\mathbb{M})$

- The remainder of action types for small groups, $\overrightarrow{\mathcal{A}}^*_{sl}(\mathbb{M})$, involve cooperation with instances in one or more large groups.

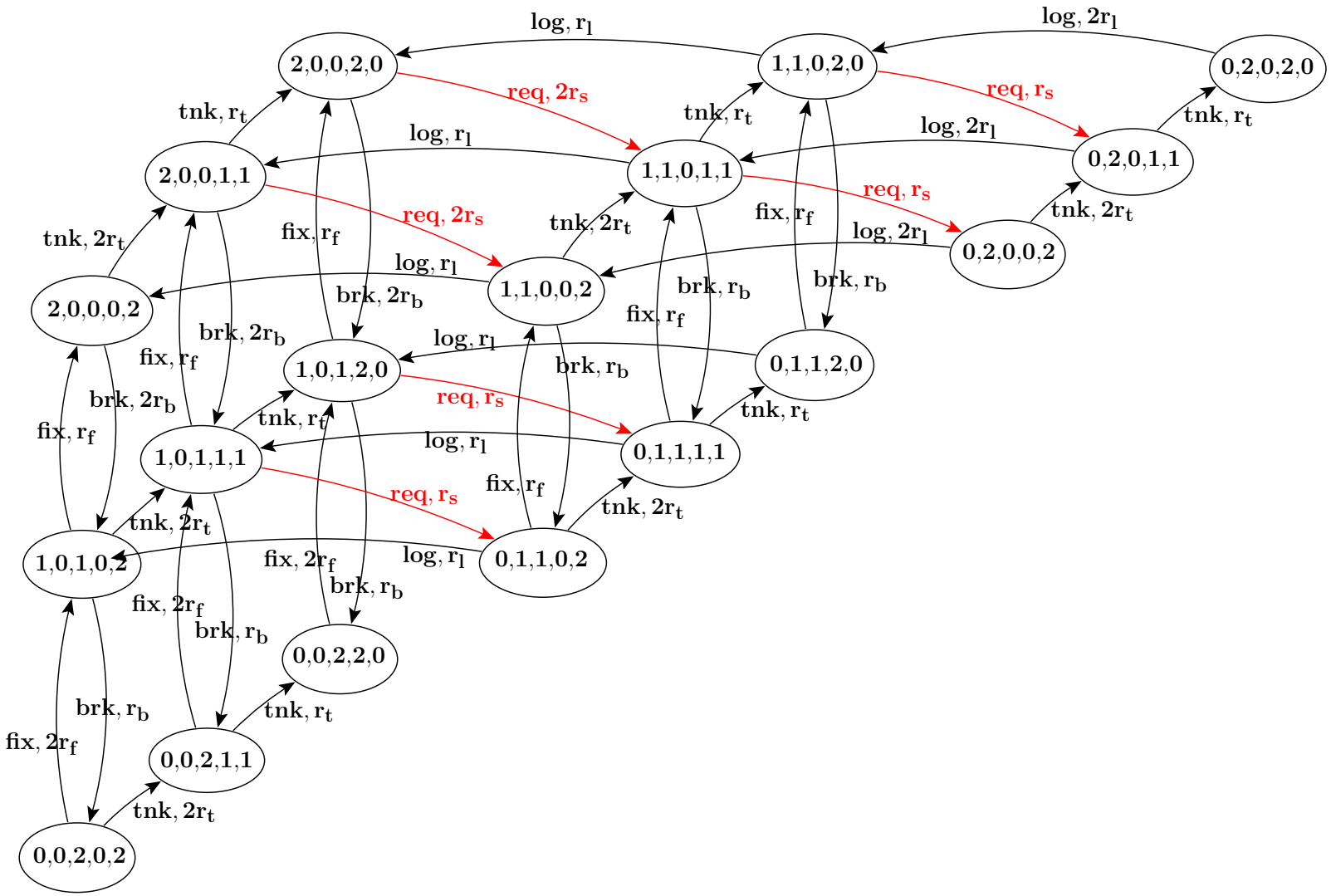
Structural Properties of the State Space: $\overrightarrow{\mathcal{A}}^*_{sl}(\mathbb{M})$

- The remainder of action types for small groups, $\overrightarrow{\mathcal{A}}^*_{sl}(\mathbb{M})$, involve cooperation with instances in one or more large groups.
- Transitions of these types change both ξ_s and ξ_l.

Structural Properties of the State Space: $\overrightarrow{\mathcal{A}^*_{sl}}(\mathbb{M})$

- The remainder of action types for small groups, $\overrightarrow{\mathcal{A}^*_{sl}}(\mathbb{M})$, involve cooperation with instances in one or more large groups.
- Transitions of these types change both ξ_s and ξ_l.
- However, due to the aggregation condition, the rates such transitions still depend only on the configuration of small populations and are independent of ξ_s.

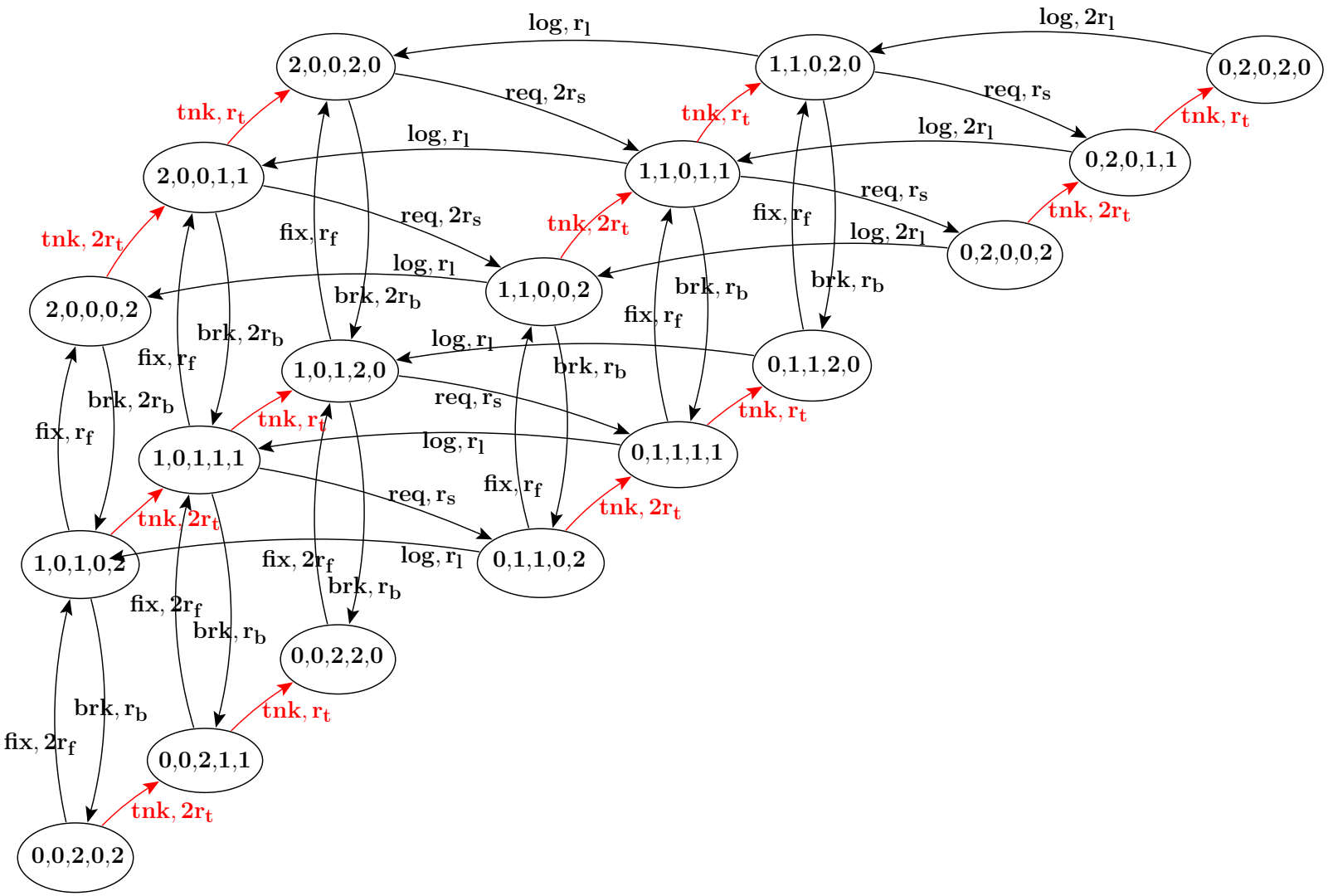
$\overrightarrow{\mathcal{A}^*_{sl}}(CS) = \{req\}$



Structural Properties of the State Space: $\overrightarrow{\mathcal{A}}_l^*(\mathbb{M})$

- The remaining action types, $\overrightarrow{\mathcal{A}}_l^*(\mathbb{M})$, represent the action types in which the large populations alone are involved.
- Transitions of these types change $\boldsymbol{\xi}_l$ only and have no effect on $\boldsymbol{\xi}_s$
- When these actions occur the state of the small groups do not change.
- The rate of such transitions depends on the configuration of large populations only.
- These transitions capture the internal dynamics of the large populations.

$\overrightarrow{\mathcal{A}_l^*}(CS) = \{tnk\}$



Identifying Sub-chains

- The underlying CTMC can be divided into a number of sub-chains of states that are connected only by $\overrightarrow{\mathcal{A}_l^*}(\mathbb{M})$ transitions and for which the configuration of instances in $\mathcal{G}_s(\mathbb{M})$ remains constant.

Identifying Sub-chains

- The underlying CTMC can be divided into a number of sub-chains of states that are connected only by $\overrightarrow{\mathcal{A}_l^*}(\mathbb{M})$ transitions and for which the configuration of instances in $\mathcal{G}_s(\mathbb{M})$ remains constant.
- From a state S, a sub-chain Y_i can be derived using these rules:

$$S \in Y_i \ \wedge \ S \xrightarrow{(\alpha,\cdot)} S' \ \wedge \ \alpha \in \overrightarrow{\mathcal{A}_l^*}(\mathbb{M}) \implies S' \in Y_i$$

$$S \in Y_i \ \wedge \ S'' \xrightarrow{(\alpha,\cdot)} S \ \wedge \ \alpha \in \overrightarrow{\mathcal{A}_l^*}(\mathbb{M}) \implies S'' \in Y_i$$

Identifying Sub-chains

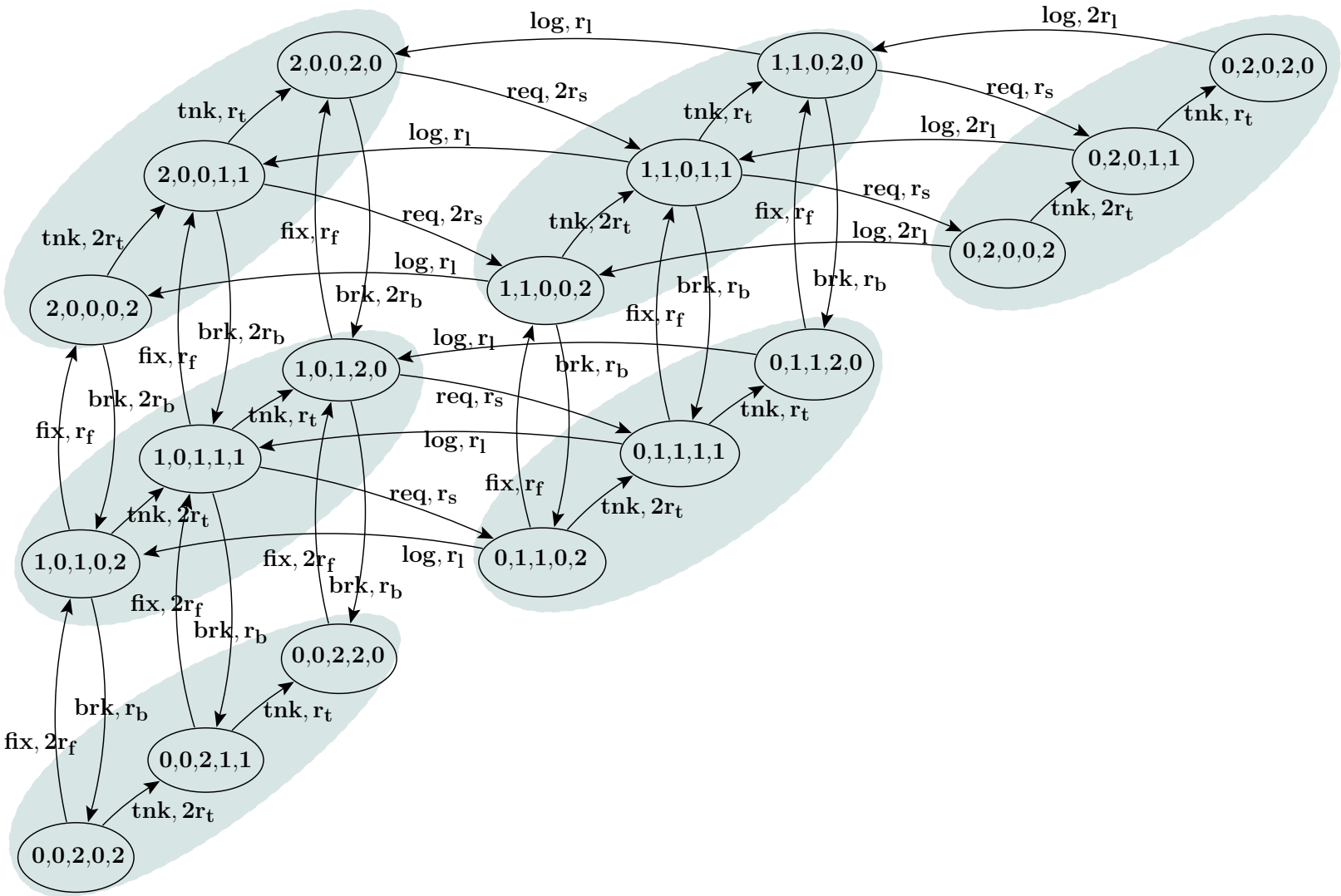
- The underlying CTMC can be divided into a number of sub-chains of states that are connected only by $\overrightarrow{\mathcal{A}_l^*}(\mathbb{M})$ transitions and for which the configuration of instances in $\mathcal{G}_s(\mathbb{M})$ remains constant.
- From a state S, a sub-chain Y_i can be derived using these rules:

$$S \in Y_i \ \wedge \ S \xrightarrow{(\alpha,\cdot)} S' \ \wedge \ \alpha \in \overrightarrow{\mathcal{A}_l^*}(\mathbb{M}) \implies S' \in Y_i$$

$$S \in Y_i \ \wedge \ S'' \xrightarrow{(\alpha,\cdot)} S \ \wedge \ \alpha \in \overrightarrow{\mathcal{A}_l^*}(\mathbb{M}) \implies S'' \in Y_i$$

- A sub-chain can be characterised by the configuration it captures for the small groups.

Identifying Sub-chains

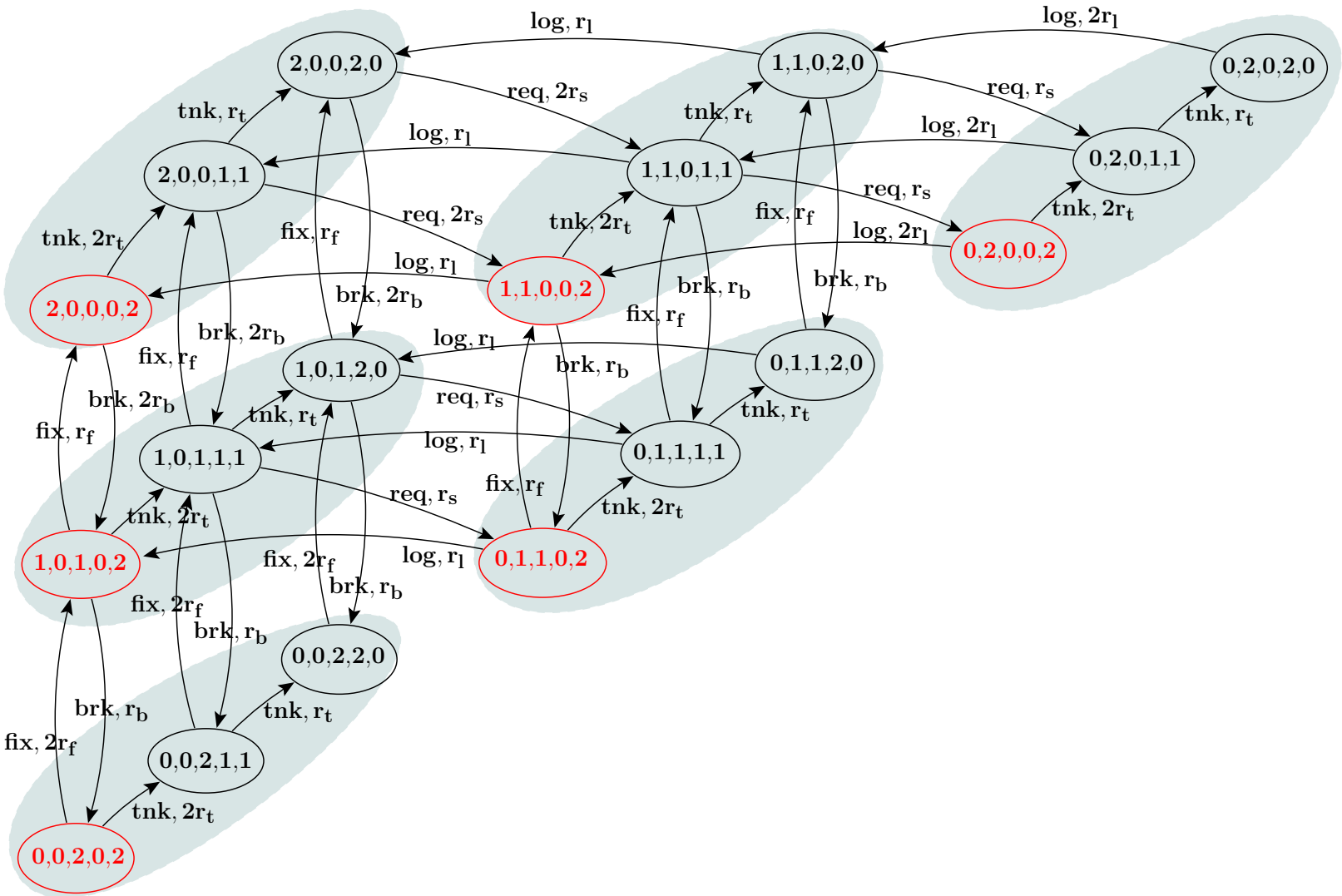


Each sub-chain identifies a unique configuration for the servers.

Low Probability Boundary States

- Consider an actions type $\alpha \in \vec{\mathcal{A}}^{*}_{sl}(\mathbb{M})$ and a sub-chain Y_i
- In any state $S \in Y_i$, whether α is enabled or not depends on the configurations of both small and large groups
- There are some states where the small groups are ready for cooperation, but the large groups are not.
  We call these the boundary states.
- In a heavily loaded system where the resources are under contention, the probability of these states is very low.

Low Probability Boundary States



The servers are ready to handle requests, but no clients are requesting.

Rate Regularity for Non-Boundary States

- All non-boundary states in sub-chain $Y_i \in Y_{\mathbb{M}}$ enable the same set of activities of type $\overrightarrow{\mathcal{A}^*_{sl}}(\mathbb{M}) \cup \overrightarrow{\mathcal{A}^*_{s}}(\mathbb{M})$.
- Moreover the rates of these activities in non-boundary states are same within each Y_i.

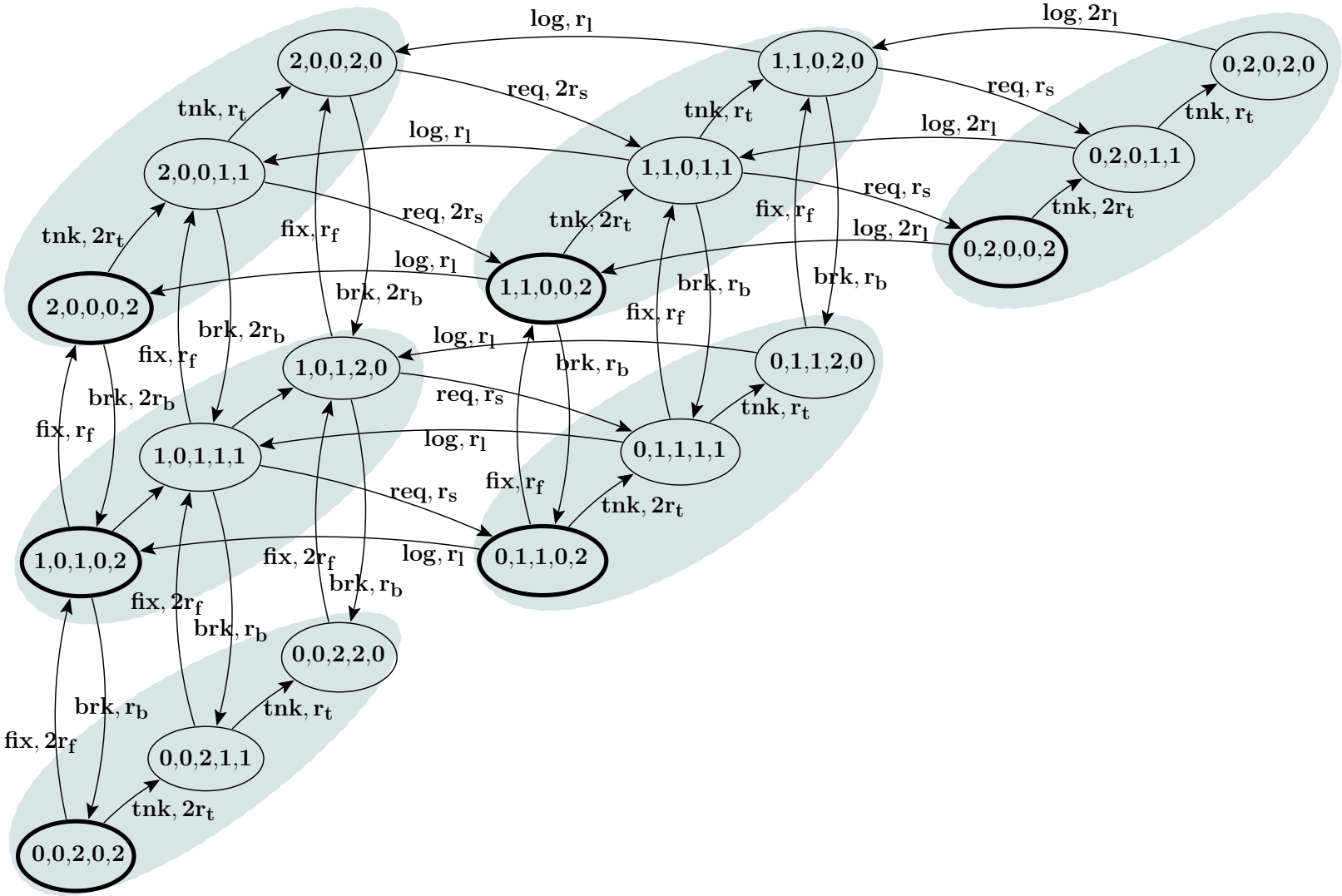
The Aggregated State Space

- Assuming that the probability of being in boundary states is close to zero, we construct an aggregated state space which captures the evolution of the process at the level of the sub-chains,
  i.e. each subchain becomes one macro-state in the aggregation.

The Aggregated State Space

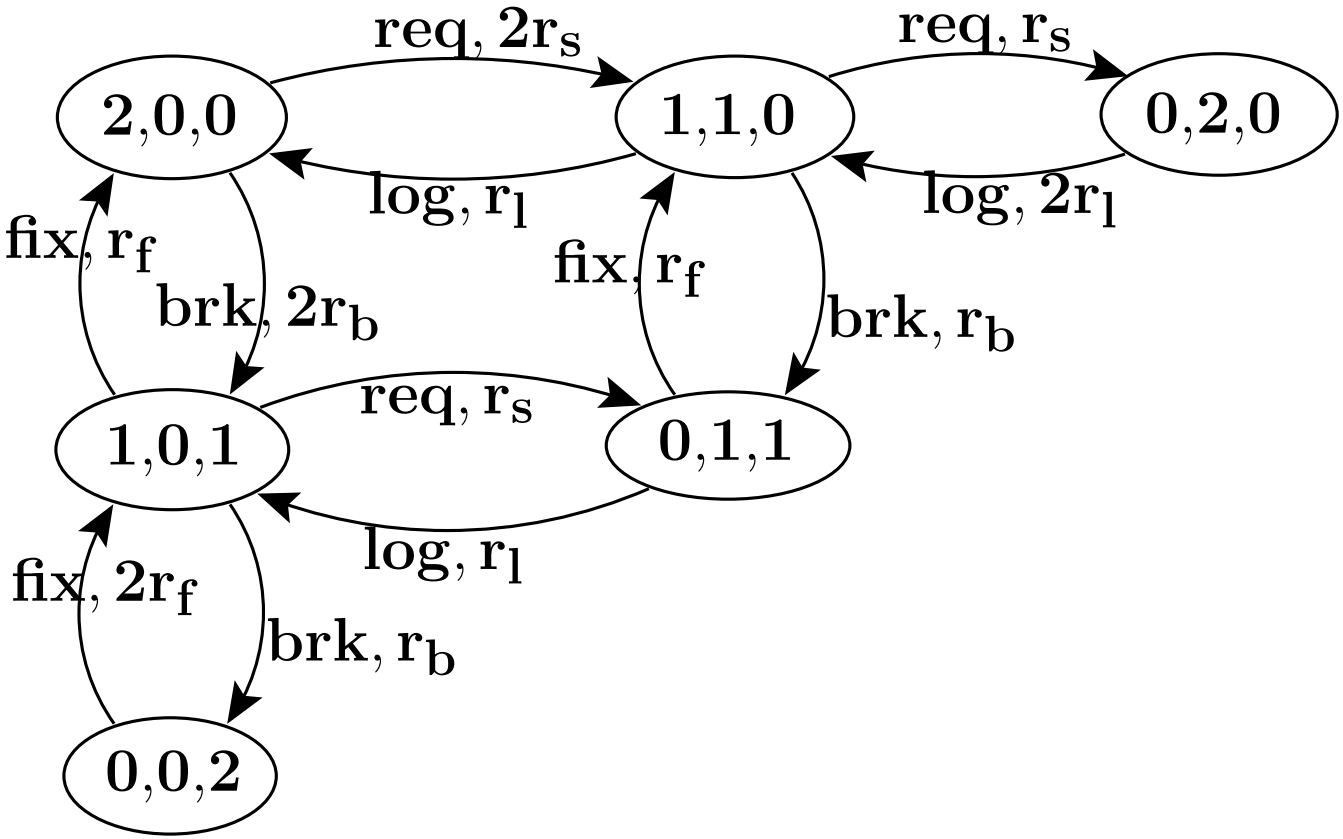
- Assuming that the probability of being in boundary states is close to zero, we construct an aggregated state space which captures the evolution of the process at the level of the sub-chains, i.e. each subchain becomes one macro-state in the aggregation.
- In aggregated state space the dynamics of the large groups are abstracted but the stochastic behaviour of the small groups can be studied explicitly.

The Complete State Space



The complete state space of the client-server system.

The Aggregated State Space



The aggregated state space of the client-server model with two servers.

Aggregation Algorithm

- For a model which satisfies the syntactic condition, we assume that the probability of being in boundary states is close to zero.
- The algorithm takes the model and directly generates its underlying aggregated state space in numerical vector form in two steps:

1. Reduction: working at the syntactic level build a reduced version of the model which abstracts from the behaviours specified for the large groups.
2. Count-oriented Structured Semantics are applied on the reduced form to generate its aggregated CTMC directly in the numerical vector form.
3. From the aggregated CTMC the marginal probability distribution with respect to the small groups can be derived.

Reduction

Consider model $\mathbb{M}$ which satisfies the aggregation condition. Reduction rules are applied on $\mathbb{M}$'s system equation to produce reduced model $\mathbb{M}_R$.

$$red(G) = \begin{cases} red(G_1) \underset{L}{\bowtie} red(G_2), & \text{if } G = G_1 \underset{L}{\bowtie} G_2 \\ H\{\cdot\}, & \text{if } G = H\{\cdot\}\ ,\ H \in \mathcal{G}_s(\mathbb{M}) \\ Nil, & \text{if } G = H\{\cdot\}\ ,\ H \in \mathcal{G}_l(\mathbb{M}) \\ red(X), & \text{if } G \stackrel{def}{=} X \end{cases}$$

The process *Nil* represents a sequential process which does not undertake any activity.

Reduction

Consider model $\mathbb{M}$ which satisfies the aggregation condition. Reduction rules are applied on $\mathbb{M}$'s system equation to produce reduced model $\mathbb{M}_R$.

$$red(G) = \begin{cases} red(G_1) \underset{L}{\bowtie} red(G_2), & \text{if } G = G_1 \underset{L}{\bowtie} G_2 \\ H\{\cdot\}, & \text{if } G = H\{\cdot\}\ ,\ H \in \mathcal{G}_s(\mathbb{M}) \\ Nil, & \text{if } G = H\{\cdot\}\ ,\ H \in \mathcal{G}_l(\mathbb{M}) \\ red(X), & \text{if } G \stackrel{def}{=} X \end{cases}$$

The process *Nil* represents a sequential process which does not undertake any activity.

The following rules removes *Nil* processes to find the minimal reduced system equation:

$$Nil \underset{\cdot}{\bowtie} Nil = Nil \qquad Nil \underset{\cdot}{\bowtie} P = P \qquad P \underset{\cdot}{\bowtie} Nil = P$$

Count-Oriented Semantics

Working on the reduced model $\mathbb{M}_R$, the count-oriented semantics works from the transitions associated with individual components in the small group, to derive the aggregated behaviour of the group as a whole.

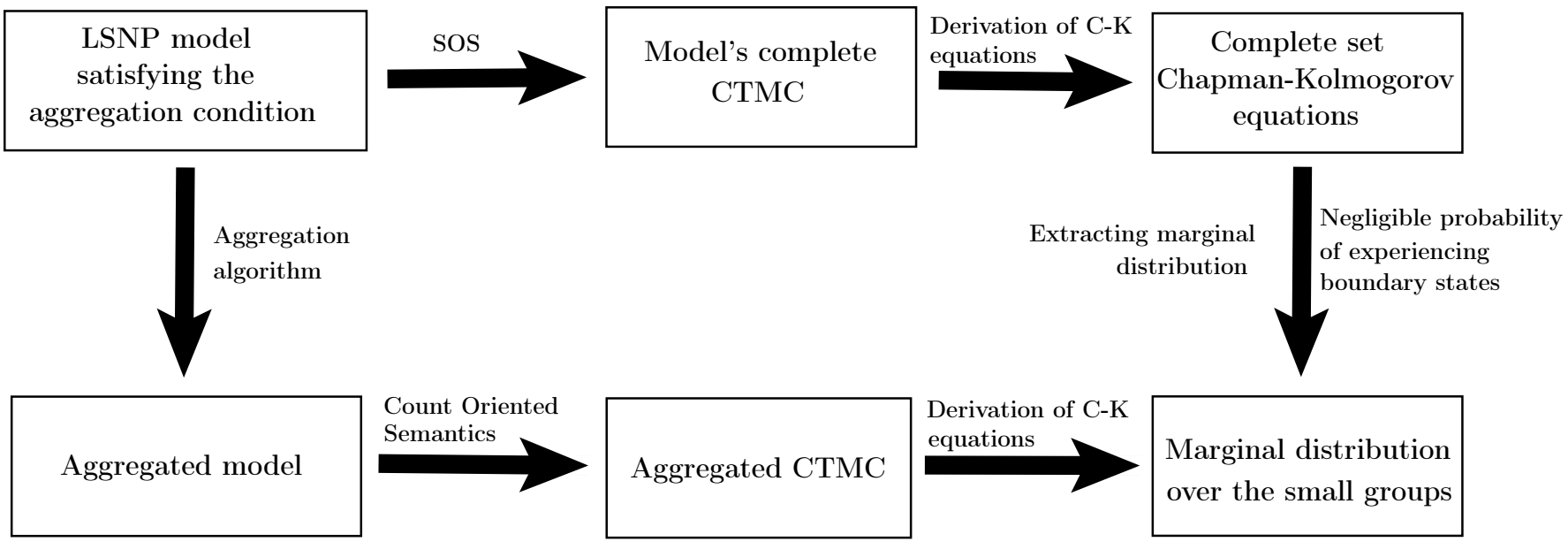
Count-Oriented Semantics

Working on the reduced model $\mathbb{M}_R$, the count-oriented semantics works from the transitions associated with individual components in the small group, to derive the aggregated behaviour of the group as a whole.

In this way the state space of the aggregated model is generated based on the numerical count vector representation of the state of each small group.

Deriving Marginal Distribution Over the Small Groups

- There are two routes from the original model to the marginal distribution: using the model's complete system of Chapman-Kolmogorov equations and using the aggregated CTMC.
- The latter is faster as it takes advantage of the aggregation.



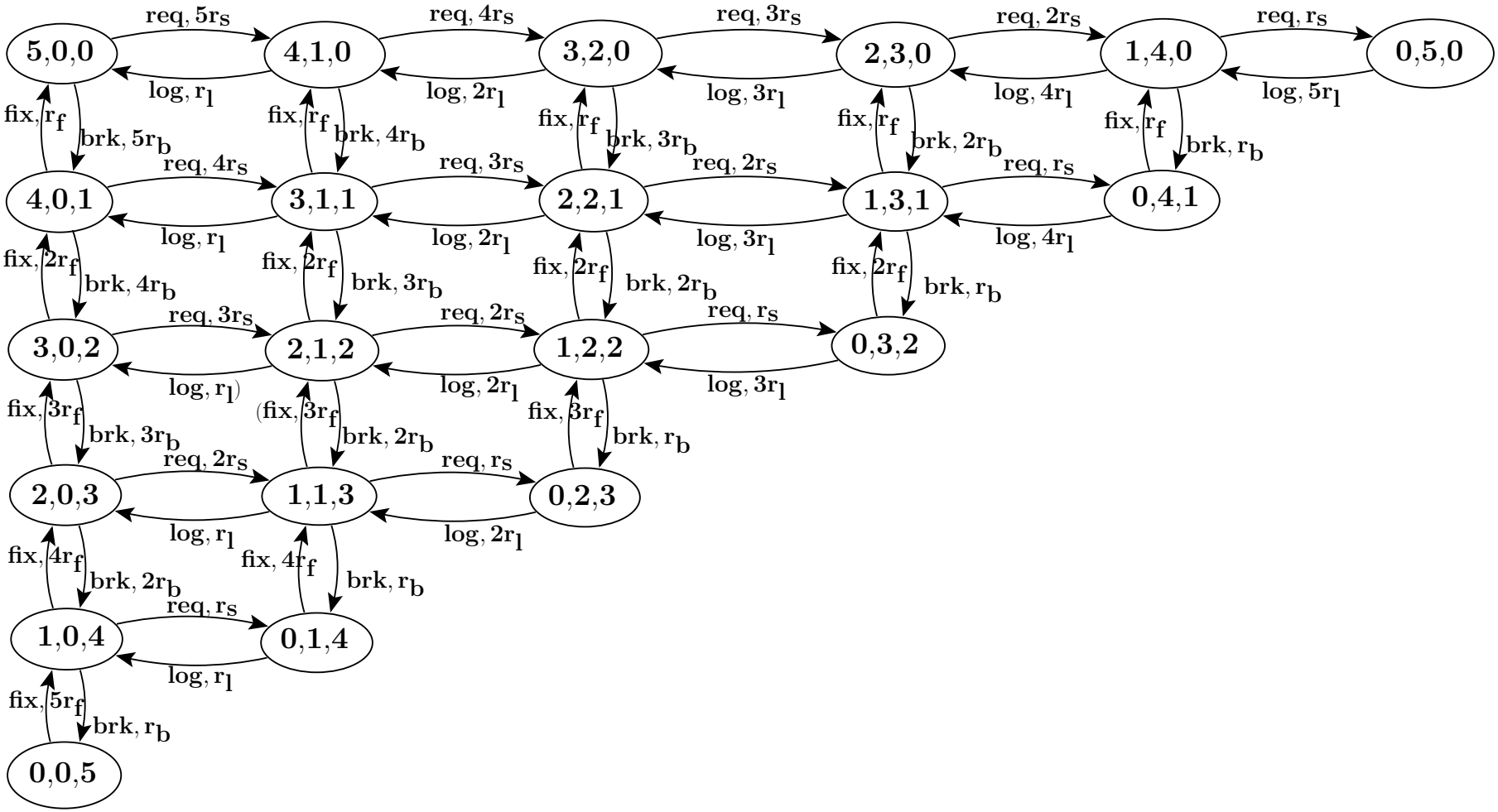
Numerical Example: Case 1

- The aggregated model is suitable to answer queries about the performance of the small groups.
- Consider the client-server model with the following parameters:

r_s	r_l	r_b	r_c	r_t	n_s	n_c
10	50	0.005	$\top$	15	5	100

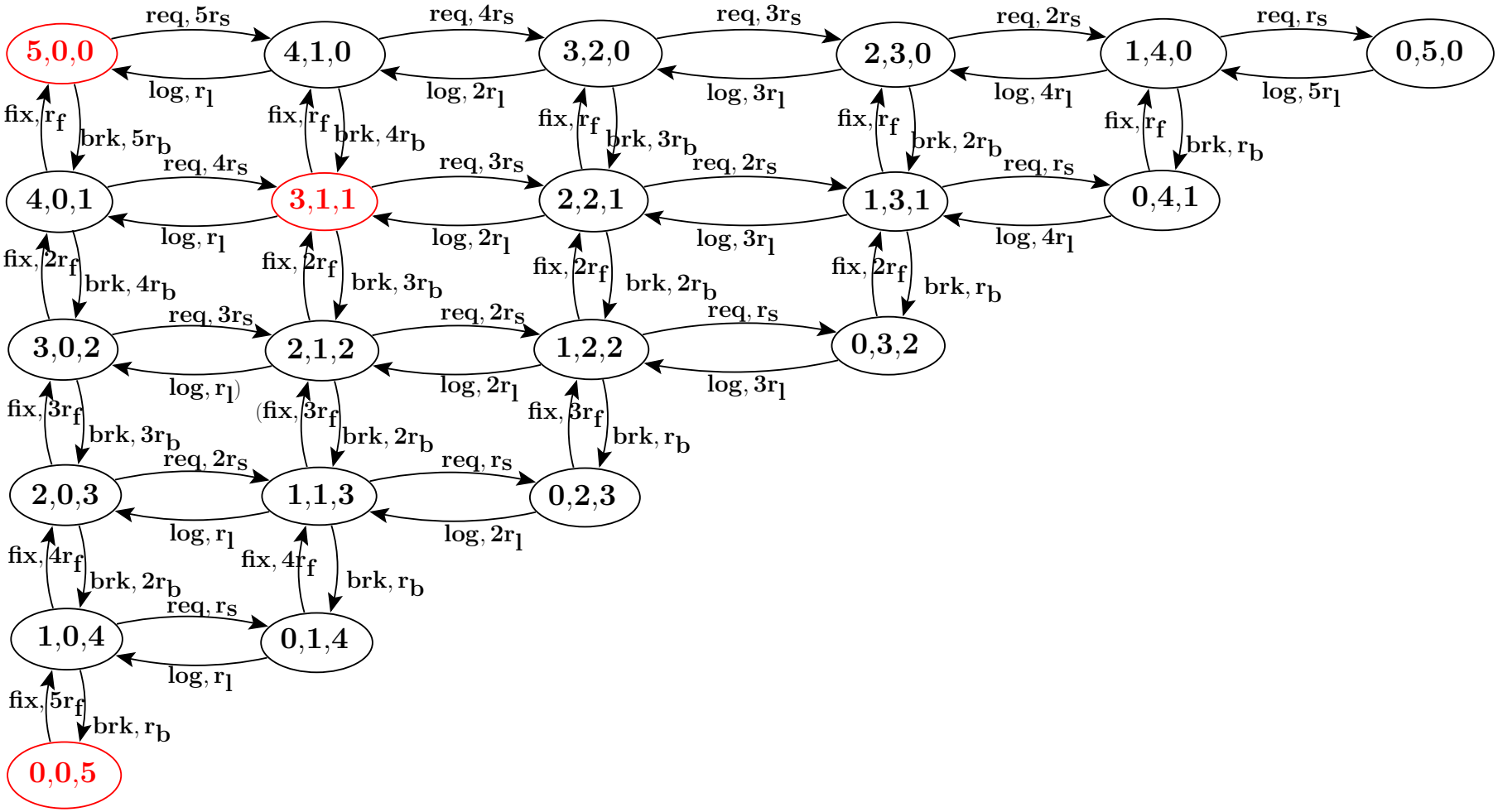
- This set of parameters guarantee that the probability of $C_r = 0 \approx 0$ (the event of being in boundary states).
- Let Z represent the model's original CTMC and Z_a, the aggregated CTMC.
- We compare the distribution associated with Z and Z_a across three representative states.

The Aggregated State Space



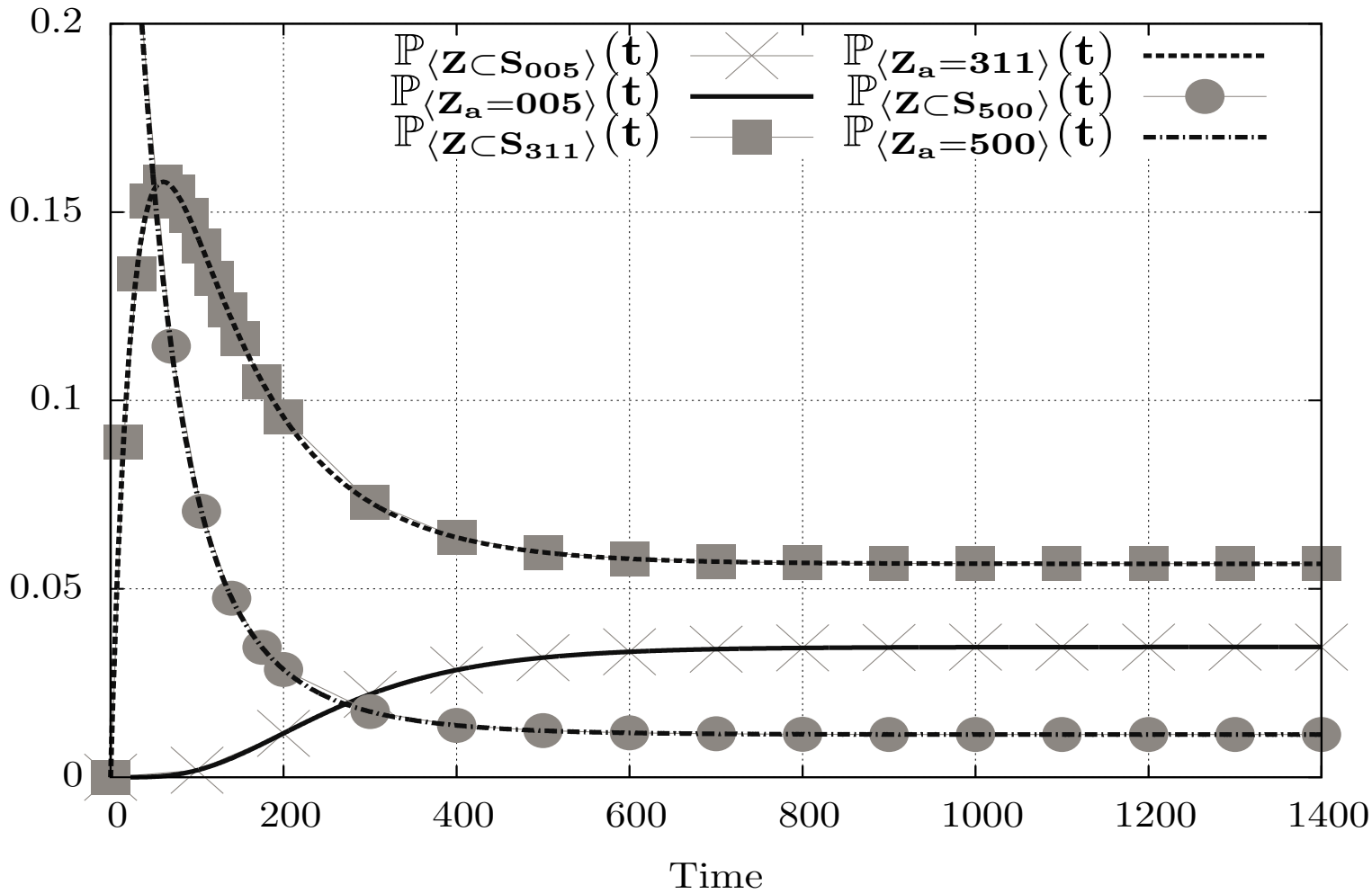
The aggregated state space of the client-server model with five servers.

The Aggregated State Space



The aggregated state space of the client-server model with five servers.

Comparison of the approximate and exact distribution across three different states.

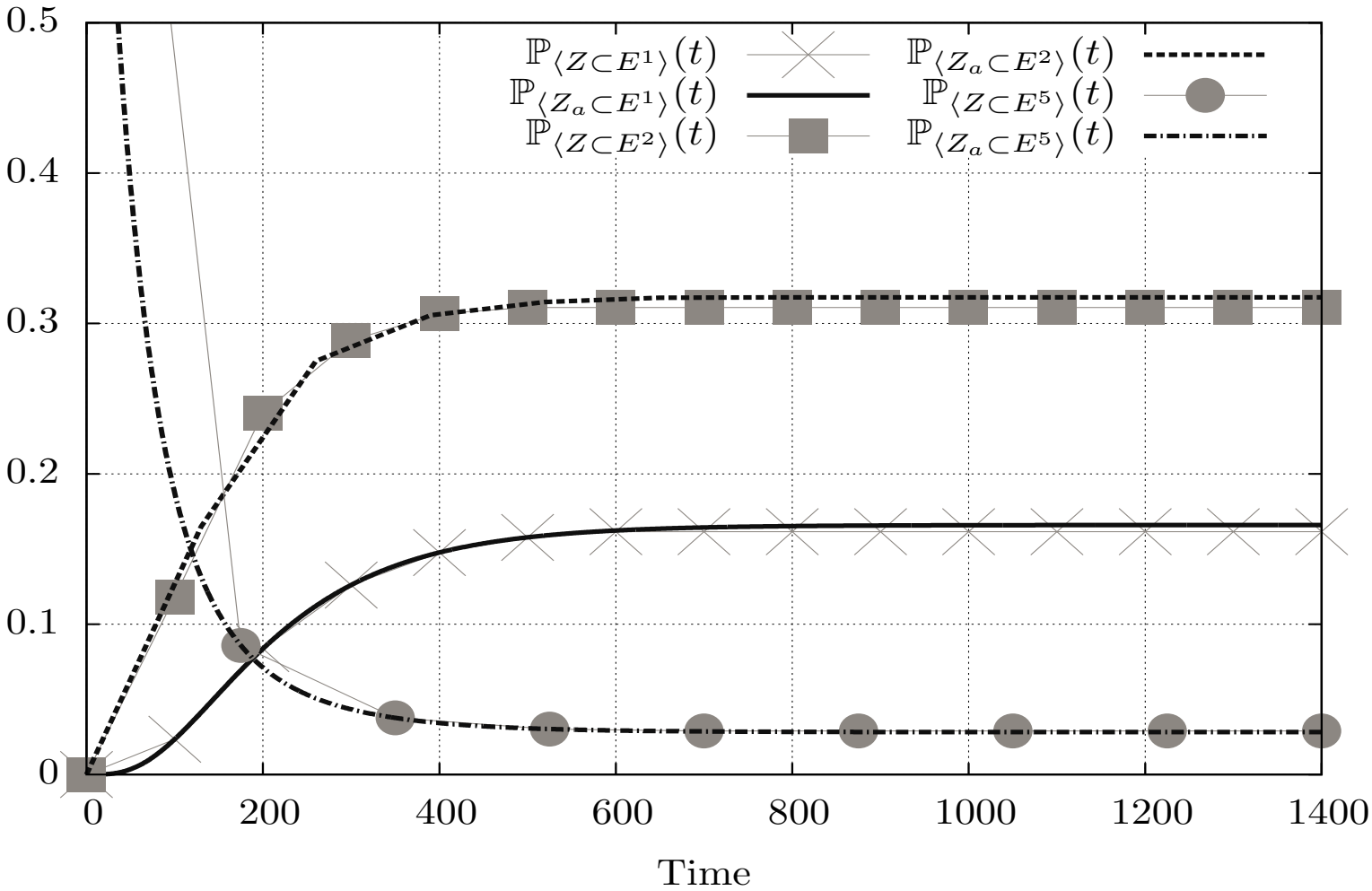


Using the Marginal Distribution for Dependability

- Assume that the measure of interest is the number of working servers, i.e. those which are not broken
- Let us define the event $E^K, k = 1, \ldots, 5$.
- E^i represents the probability of having i servers working at time t.

$$\mathbb{P}_{\langle Z \subset E^5 \rangle} = \mathbb{P}_{\langle Z \subset (5,0,0) \rangle} + \mathbb{P}_{\langle Z \subset (4,1,0) \rangle} + \cdots + \mathbb{P}_{\langle Z \subset (0,5,0) \rangle} \approx$$
$$\mathbb{P}_{\langle Z_a \subset E^5 \rangle} = \mathbb{P}_{\langle Z_a = (5,0,0) \rangle} + \mathbb{P}_{\langle Z_a = (4,1,0) \rangle} + \cdots + \mathbb{P}_{\langle Z_a = (0,5,0) \rangle}$$

Comparison of the approximate and exact dependability measure.



Exact and Approximate Marginal Distribution

	First Case		
	exact	approximate	error (%)
$\mathbb{P}_{\langle Z_a=500\rangle}$	0.011	0.011	0
$\mathbb{P}_{\langle Z_a=311\rangle}$	0.056	0.056	0
$\mathbb{P}_{\langle Z_a=005\rangle}$	0.034	0.034	0
$\mathbb{P}_{\langle Z_a\subset E^5\rangle}$	0.028	0.028	0
$\mathbb{P}_{\langle Z_a\subset E^2\rangle}$	0.310	0.317	2
$\mathbb{P}_{\langle Z_a\subset E^1\rangle}$	0.161	0.165	2.48

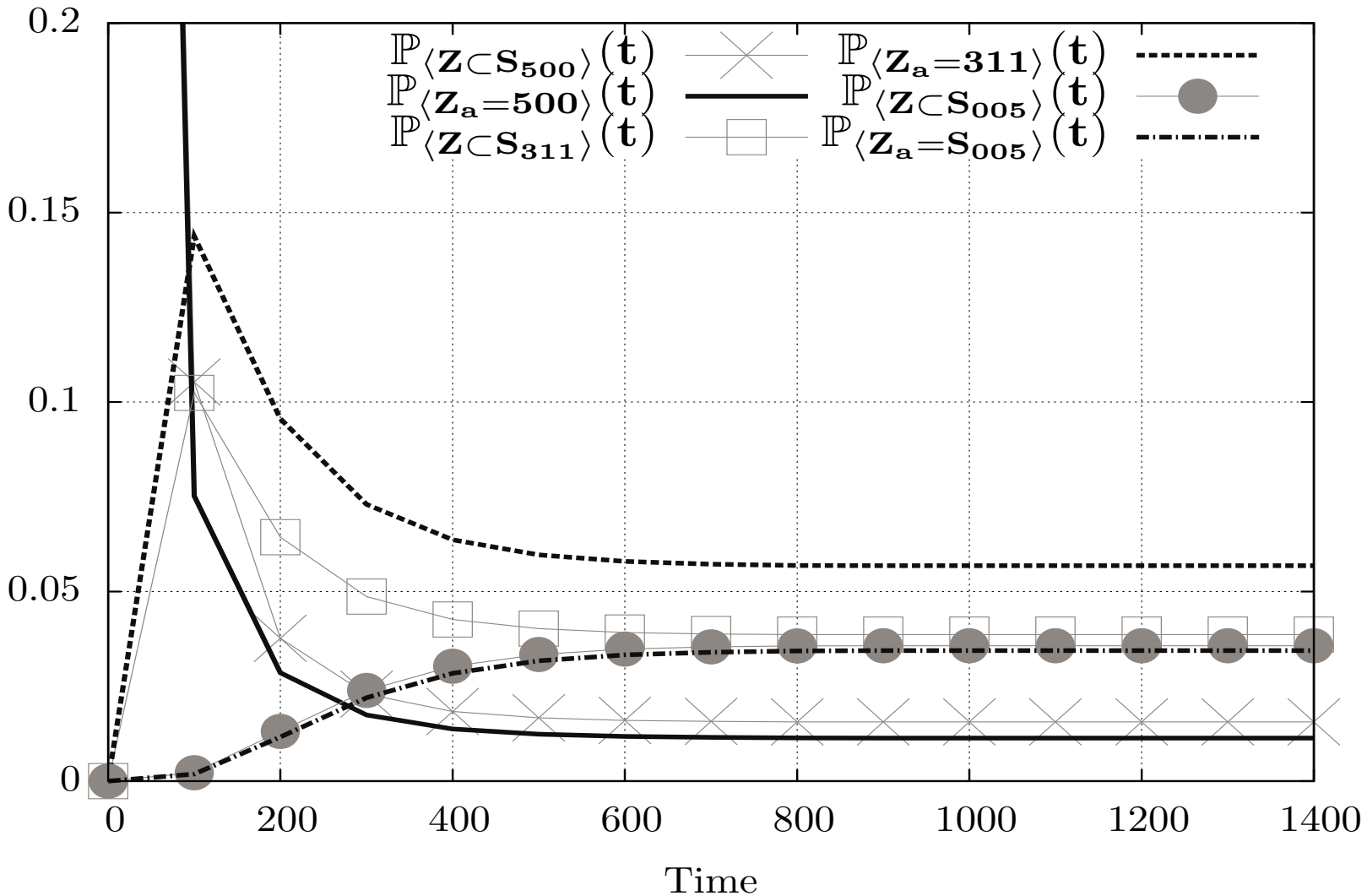
Accuracy of the Aggregation: Case 2

- We consider a client-server model with the following parameters:

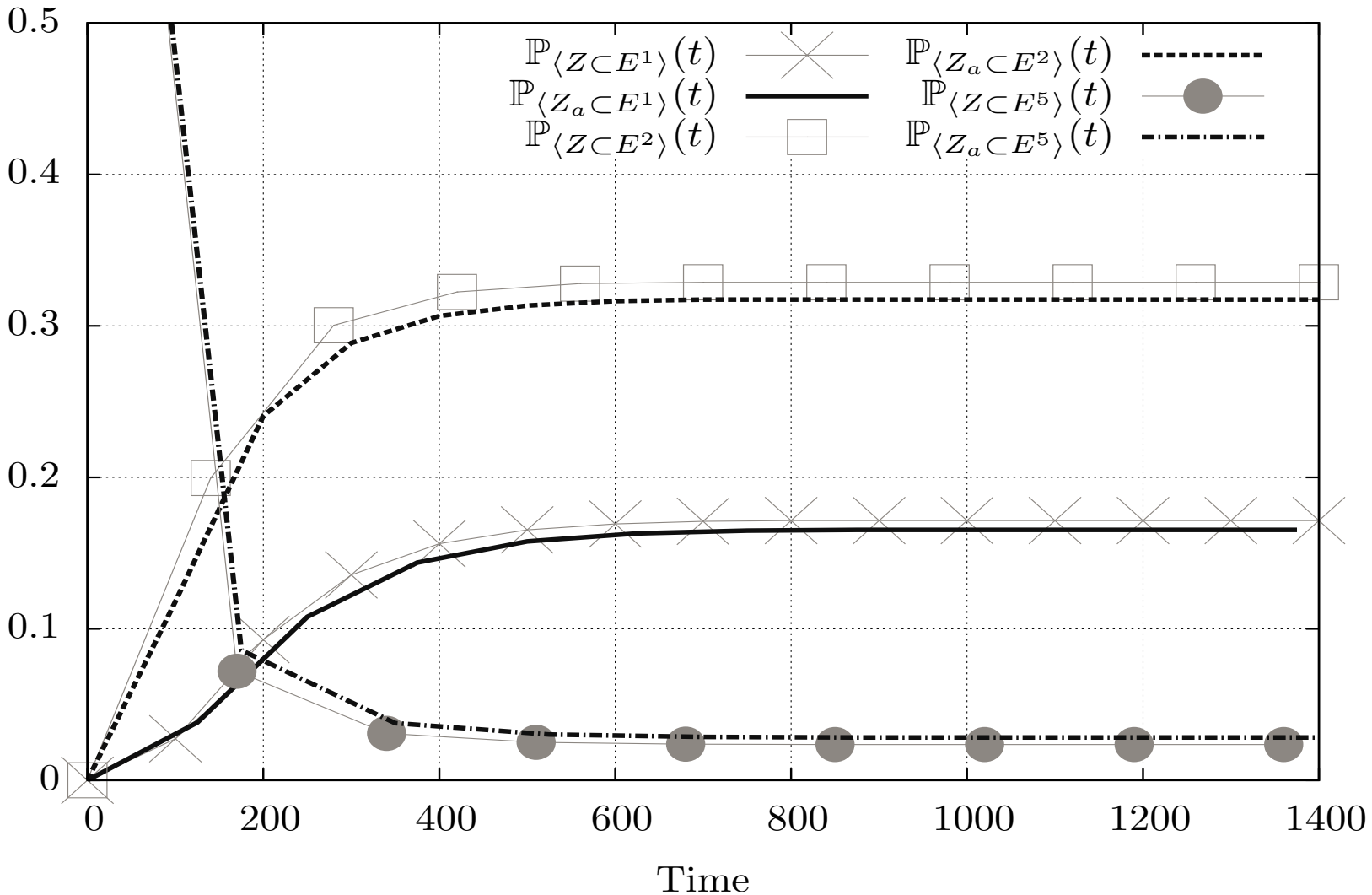
r_s	r_l	r_b	r_c	r_t	n_s	n_c
10	50	0.005	$\top$	2	5	100

- Clients spend longer thinking; i.e. the load on the servers is decreased.
- Compared to the first case, this set of parameters increases the probability of $C_r = 0$.
- Again, we compare the distribution associated with Z and Z_a across three representative states.

Comparison of Exact and Approximate Marginal Distributions



Comparison of Exact and Approximate Marginal Distributions



Exact and Approximate Marginal Distribution

	Second Case		
	exact	approximate	error (%)
$\mathbb{P}_{\langle Z_a=500\rangle}$	0.015	0.011	26
$\mathbb{P}_{\langle Z_a=311\rangle}$	0.038	0.056	47
$\mathbb{P}_{\langle Z_a=005\rangle}$	0.035	0.034	2
$\mathbb{P}_{\langle Z_a\subset E^5\rangle}$	0.023	0.028	21
$\mathbb{P}_{\langle Z_a\subset E^2\rangle}$	0.328	0.317	3
$\mathbb{P}_{\langle Z_a\subset E^1\rangle}$	0.171	0.165	3.5

Comparison of the equilibrium probabilities when the probability of being in boundary states is larger.

Conclusion

- We presented an aggregation method which is suitable for the class of PEPA models with non-uniform populations.
- The aggregation condition can be readily checked by considering the model description.
- The aggregated state space can be generated directly from the model and can be used for deriving a marginal distribution over the model's small groups.

Future Work

- The aggregation can be used to detect important properties about the model at a reduced cost, and we are exploring its connections with time-scale decomposability.
- We are also exploring relaxing the assumption of the large populations being passive with respect to the shared activity.
- Finally we hope to use a hybrid mean-field approach for the analysis of large scale non-uniform population models. This would allow us to study the impact of the stochastic behaviour of the small groups on the large ones.