Stabilization of Branching Queueing Networks

Tomáš Brázdil[1] Stefan Kiefer[2]

[1]Masaryk University, Brno, Czech Republic

[2]University of Oxford, UK

STACS, Paris
02 March 2012

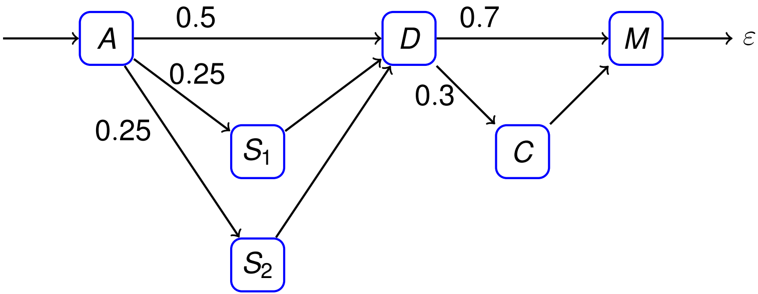
Motivation

Queueing networks are simple and general.

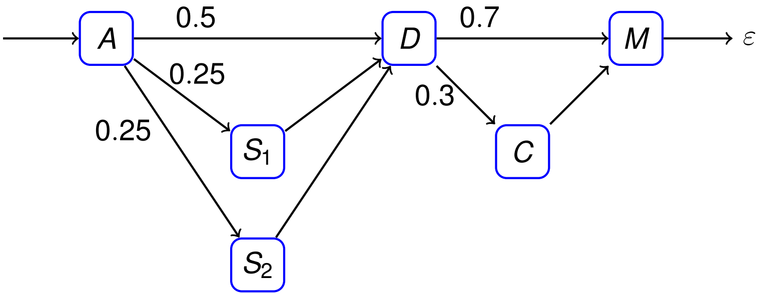
Attractive for modeling parallelism:

- Hardware, especially large-scale multi-core systems:
  - Full-system performance simulators do not scale.
  - Queueing theory gives a more abstract analysis.
- Software, especially message passing:
  - asynchronous programs on multi-core computers
  - distributed programs on a network

Example: Router



Example: Router



Textual representation of the same thing:

$$0 \overset{1}{\hookrightarrow} A$$

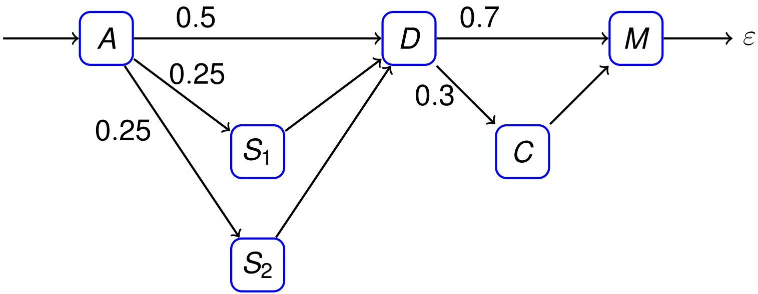
$$A \overset{0.5}{\hookrightarrow} D \qquad A \overset{0.25}{\hookrightarrow} S_1 \qquad A \overset{0.25}{\hookrightarrow} S_2$$

$$D \overset{0.7}{\hookrightarrow} M \qquad D \overset{0.3}{\hookrightarrow} C$$

$$S_1 \overset{1}{\hookrightarrow} D \qquad S_2 \overset{1}{\hookrightarrow} D$$

$$C \overset{1}{\hookrightarrow} M \qquad M \overset{1}{\hookrightarrow} \varepsilon$$

Example: Router



Textual representation of the same thing:

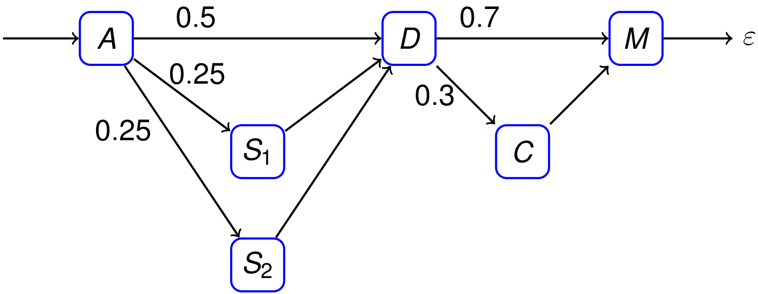
$$0 \overset{1}{\hookrightarrow} A \qquad \begin{array}{l} A \overset{0.5}{\hookrightarrow} D \\ A \overset{0.25}{\hookrightarrow} S_1 \\ A \overset{0.25}{\hookrightarrow} S_2 \end{array} \qquad \begin{array}{l} D \overset{0.7}{\hookrightarrow} M \\ D \overset{0.3}{\hookrightarrow} C \end{array} \qquad \begin{array}{l} S_1 \overset{1}{\hookrightarrow} D \\ S_2 \overset{1}{\hookrightarrow} D \end{array} \qquad \begin{array}{l} C \overset{1}{\hookrightarrow} M \\ M \overset{1}{\hookrightarrow} \varepsilon \end{array}$$

Also given: arrival rate α_A

queue rates $\mu_A, \mu_{S_1}, \ldots$ of busy queues

$\longrightarrow$ Continuous-Time Markov Chain – Infinite-State!

Jackson Networks



Such networks are called Jackson networks (classical model).
Nice properties:

- Stability is easy to determine.
- product form solutions in "steady state":

$$\Pr(S_1 = 3 \text{ and } D = 2) = \Pr(S_1 = 3) \cdot \Pr(D = 2)$$

But shortcomings in modeling:

- just one new task per one old task
- no (dynamic) control of the network

Branching Queueing Networks

New model: Branching Queueing Networks (BQNs)

$$0 \overset{1}{\hookrightarrow} A \qquad \begin{array}{l} A \overset{1/5}{\hookrightarrow} B, B \\ A \overset{4/5}{\hookrightarrow} \varepsilon \end{array} \qquad \begin{array}{l} B \overset{1/6}{\hookrightarrow} A, B \\ B \overset{5/6}{\hookrightarrow} \varepsilon \end{array}$$

Rates as before: $\alpha_A, \mu_A, \mu_B > 0$

Similar models:

- a bit like pushdown systems, but parallel
- a bit like stochastic Petri nets, but of a special form

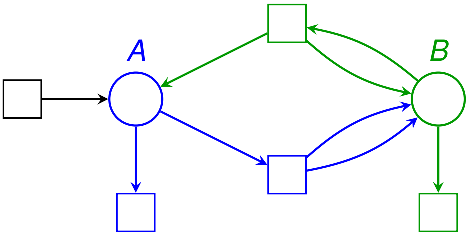
Branching Queueing Networks

New model: Branching Queueing Networks (BQNs)

$$0 \xhookrightarrow{1} A \qquad A \xhookrightarrow{1/5} B, B \qquad B \xhookrightarrow{1/6} A, B$$

$$A \xhookrightarrow{4/5} \varepsilon \qquad B \xhookrightarrow{5/6} \varepsilon$$

Rates as before: $\alpha_A, \mu_A, \mu_B > 0$



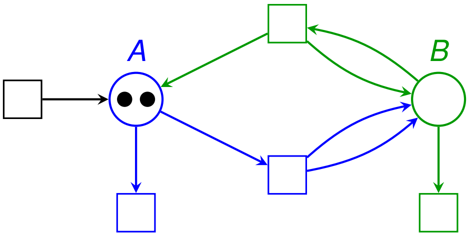
Branching Queueing Networks

New model: Branching Queueing Networks (BQNs)

$$0 \overset{1}{\hookrightarrow} A \qquad A \overset{1/5}{\hookrightarrow} B, B \qquad B \overset{1/6}{\hookrightarrow} A, B$$

$$A \overset{4/5}{\hookrightarrow} \varepsilon \qquad B \overset{5/6}{\hookrightarrow} \varepsilon$$

Rates as before: $\alpha_A, \mu_A, \mu_B > 0$

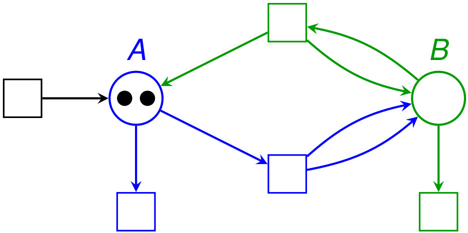


Branching Queueing Networks

New model: Branching Queueing Networks (BQNs)

$$0 \overset{1}{\hookrightarrow} A \qquad \begin{array}{l} A \overset{1/5}{\hookrightarrow} B, B \\ A \overset{4/5}{\hookrightarrow} \varepsilon \end{array} \qquad \begin{array}{l} B \overset{1/6}{\hookrightarrow} A, B \\ B \overset{5/6}{\hookrightarrow} \varepsilon \end{array}$$

Rates as before: $\alpha_A, \mu_A, \mu_B > 0$



Topic of this talk: Is a given BQN **stable**??

stable $\overset{\text{def}}{=}$ expected return time to "completely empty" is finite

No Product Form

Consider the BQN

$$0 \overset{1}{\hookrightarrow} A \qquad A \overset{1}{\hookrightarrow} B, C \qquad B \overset{1}{\hookrightarrow} \varepsilon \qquad C \overset{1}{\hookrightarrow} \varepsilon$$

No product form: Take $\alpha_A = 1,\ \mu_A = \mu_B = \mu_C = 3$.
Then in steady-state:

$$\Pr(C \geq 1) = 1/3$$
$$\Pr(C \geq 1 \mid B \geq 1) \geq 3/7$$

No Product Form

Consider the BQN

$$0 \overset{1}{\hookrightarrow} A \qquad A \overset{1}{\hookrightarrow} B, C \qquad B \overset{1}{\hookrightarrow} \varepsilon \qquad C \overset{1}{\hookrightarrow} \varepsilon$$

No product form: Take $\alpha_A = 1,\ \mu_A = \mu_B = \mu_C = 3$.
Then in steady-state:

$$\Pr(C \geq 1) = 1/3$$
$$\Pr(C \geq 1 \mid B \geq 1) \geq 3/7$$

$\longrightarrow$Analysis of BQNs harder than for Jackson networks

Balance Equations

$$0 \xhookrightarrow{1} A \qquad A \xhookrightarrow{1/5} B, B \qquad B \xhookrightarrow{1/6} A, B$$

$$A \xhookrightarrow{4/5} \varepsilon \qquad B \xhookrightarrow{5/6} \varepsilon$$

$$\alpha_A = 0.2 \qquad \mu_A = 0.4 \qquad \mu_B = 0.3$$

(Pre-)Suppose a form of balance:

tasks processed at X per sec $=$ # tasks arriving at X per sec

Here:
$$\lambda_A = 0.2 + \lambda_B \cdot (1/6)$$
$$\lambda_B = \lambda_A \cdot (1/5) \cdot 2 + \lambda_B \cdot (1/6)$$

We call the solution $\boldsymbol{\lambda}$ the "throughput".
Note: the speeds of the queues μ_A, μ_B do not occur.

Here:
$$\lambda_A = 0.22$$
$$\lambda_B = 0.10$$

A Stability Result

Proposition

Given a BQN.

- *Let* $\boldsymbol{\lambda}$ *be the throughput,*
  i.e., $\boldsymbol{\lambda}$ *solves the balance equations* $\boldsymbol{\lambda} = \boldsymbol{\alpha} + M\boldsymbol{\lambda}$.
  Suppose $\boldsymbol{\lambda} < \boldsymbol{\mu}$ *(in all components).*
  Then the BQN is stable.
- *Conversely, if the BQN is stable,*
  then there is $\boldsymbol{\lambda}$ *with* $\boldsymbol{\lambda} = \boldsymbol{\alpha} + M\boldsymbol{\lambda} < \boldsymbol{\mu}$.

A Stability Result

Proposition

Given a BQN.

- *Let* $\boldsymbol{\lambda}$ *be the throughput,*
  i.e., $\boldsymbol{\lambda}$ *solves the balance equations* $\boldsymbol{\lambda} = \boldsymbol{\alpha} + M\boldsymbol{\lambda}$.
  Suppose $\boldsymbol{\lambda} < \boldsymbol{\mu}$ *(in all components).*
  Then the BQN is stable.
- *Conversely, if the BQN is stable,*
  then there is $\boldsymbol{\lambda}$ *with* $\boldsymbol{\lambda} = \boldsymbol{\alpha} + M\boldsymbol{\lambda} < \boldsymbol{\mu}$.

Theorem

Stability of a BQN can be decided in polynomial time.

Proposition

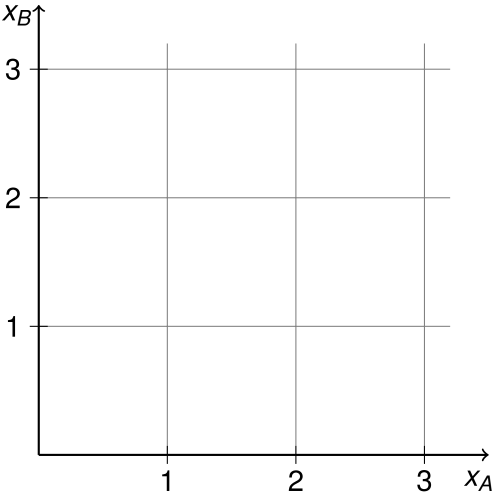
If a BQN is stable, it is "very much so": in steady state there is an exponential moment of the total queue size,
i.e., there is $\delta > 0$ *such that* $\sum_{\boldsymbol{x} \in \mathbb{N}^n} \exp(\delta \|\boldsymbol{x}\|) \Pr(\boldsymbol{x})$ *exists.*

Proof of the Stability Result

The stability result requires a delicate proof (no product form).

$$0 \xhookrightarrow{1} A \;\left|\; \begin{array}{l} A \xhookrightarrow{1/5} B, B \\ A \xhookrightarrow{4/5} \varepsilon \end{array} \right|\; \begin{array}{l} B \xhookrightarrow{1/6} A, B \\ B \xhookrightarrow{5/6} \varepsilon \end{array} \;\Big|$$

Consider "drift" for all points:



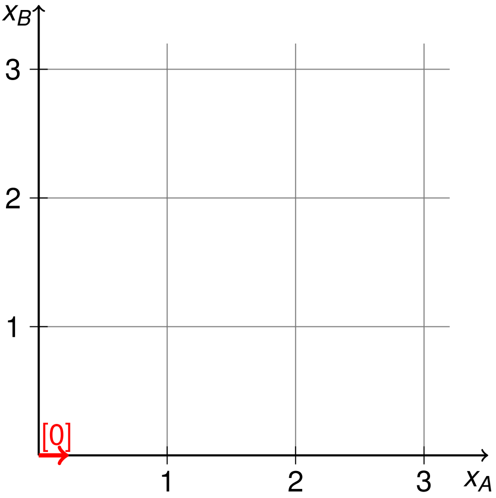
Proof of the Stability Result

The stability result requires a delicate proof (no product form).

$$0 \xhookrightarrow{1} A \;\middle|\; \begin{matrix} A \xhookrightarrow{1/5} B, B \\ A \xhookrightarrow{4/5} \varepsilon \end{matrix} \;\middle|\; \begin{matrix} B \xhookrightarrow{1/6} A, B \\ B \xhookrightarrow{5/6} \varepsilon \end{matrix} \;\middle|$$

Consider "drift" for all points:

For $(0,0)$: $[0] = \begin{pmatrix} \alpha_A \\ 0 \end{pmatrix}$



Proof of the Stability Result

The stability result requires a delicate proof (no product form).

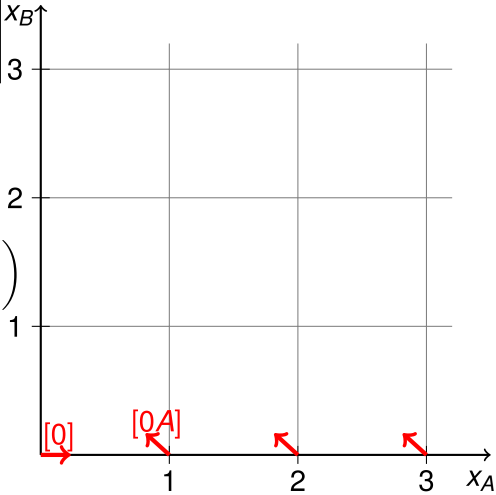
$$0 \overset{1}{\hookrightarrow} A \;\left|\; \begin{array}{l} A \overset{1/5}{\hookrightarrow} B, B \\ A \overset{4/5}{\hookrightarrow} \varepsilon \end{array} \;\right|\; \begin{array}{l} B \overset{1/6}{\hookrightarrow} A, B \\ B \overset{5/6}{\hookrightarrow} \varepsilon \end{array} \;\right|$$

Consider "drift" for all points:

For $(0,0)$: $[0] = \begin{pmatrix} \alpha_A \\ 0 \end{pmatrix}$

For $(+,0)$: $[0A] = [0] + [A]$ with

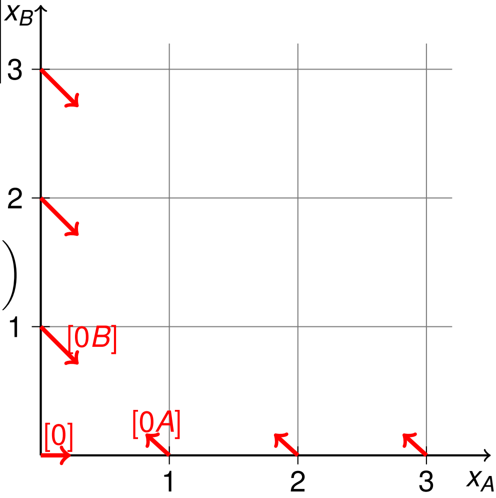
$$[A] = \mu_A \cdot \left(\begin{pmatrix} -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ (1/5) \cdot 2 \end{pmatrix} \right)$$



Proof of the Stability Result

The stability result requires a delicate proof (no product form).

$$0 \xhookrightarrow{1} A \;\Big|\; \begin{array}{l} A \xhookrightarrow{1/5} B, B \\ A \xhookrightarrow{4/5} \varepsilon \end{array} \;\Big|\; \begin{array}{l} B \xhookrightarrow{1/6} A, B \\ B \xhookrightarrow{5/6} \varepsilon \end{array} \;\Big|$$



Consider “drift” for all points:

For $(0, 0)$: $[0] = \begin{pmatrix} \alpha_A \\ 0 \end{pmatrix}$

For $(+, 0)$: $[0A] = [0] + [A]$ with

$$[A] = \mu_A \cdot \left(\begin{pmatrix} -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ (1/5) \cdot 2 \end{pmatrix} \right)$$

For $(0, +)$: $[0B] = [0] + [B]$

Proof of the Stability Result

The stability result requires a delicate proof (no product form).

$$0 \xhookrightarrow{1} A \;\Bigg|\; \begin{array}{l} A \xhookrightarrow{1/5} B, B \\ A \xhookrightarrow{4/5} \varepsilon \end{array} \;\Bigg|\; \begin{array}{l} B \xhookrightarrow{1/6} A, B \\ B \xhookrightarrow{5/6} \varepsilon \end{array} \;\Bigg|$$

Consider “drift” for all points:

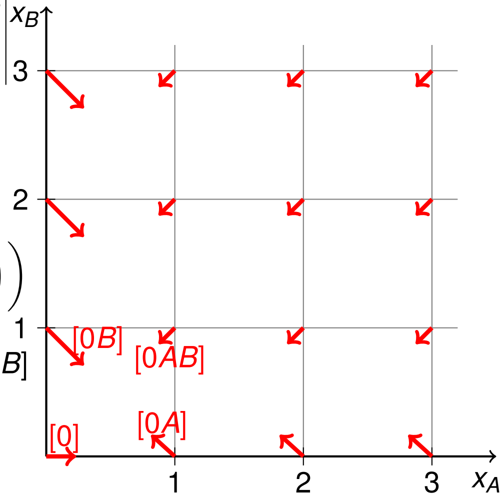
For $(0,0)$: $[0] = \begin{pmatrix} \alpha_A \\ 0 \end{pmatrix}$

For $(+,0)$: $[0A] = [0] + [A]$ with

$$[A] = \mu_A \cdot \left(\begin{pmatrix} -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ (1/5) \cdot 2 \end{pmatrix} \right)$$

For $(0,+)$: $[0B] = [0] + [B]$

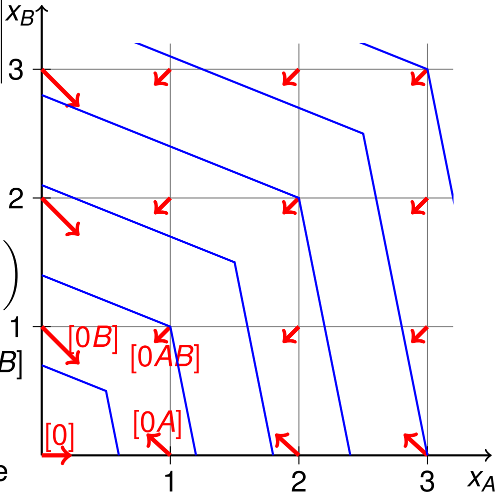
For $(+,+)$: $[0AB] = [0] + [A] + [B]$



Proof of the Stability Result

The stability result requires a delicate proof (no product form).

$$0 \xhookrightarrow{1} A \;\Bigg|\; \begin{array}{l} A \xhookrightarrow{1/5} B, B \\ A \xhookrightarrow{4/5} \varepsilon \end{array} \;\Bigg|\; \begin{array}{l} B \xhookrightarrow{1/6} A, B \\ B \xhookrightarrow{5/6} \varepsilon \end{array} \;\Bigg|$$



Consider "drift" for all points:

For $(0,0)$: $[0] = \begin{pmatrix} \alpha_A \\ 0 \end{pmatrix}$

For $(+,0)$: $[0A] = [0] + [A]$ with

$$[A] = \mu_A \cdot \left(\begin{pmatrix} -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ (1/5) \cdot 2 \end{pmatrix} \right)$$

For $(0,+)$: $[0B] = [0] + [B]$

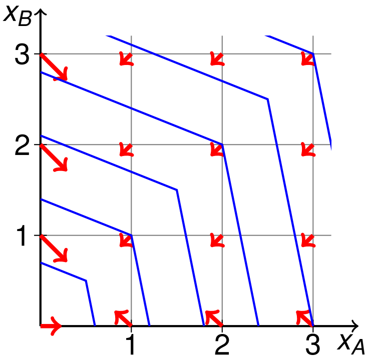
For $(+,+)$: $[0AB] = [0] + [A] + [B]$

Find a Lyapunov function
w.r.t. which the drift is negative

Proof of the Stability Result (cont'd)

Key steps:

- Construct piecewise-linear Lyapunov function w.r.t. which the drift is negative almost everywhere (hard).
  - use throughput λ and apply Farkas' lemma (wouldn't work for general stochastic Petri nets)
- Smooth the Lyapunov function (standard)
- Derive (strong) stability using Foster's criterion (standard)



Controlled Networks

$$0 \xhookrightarrow{1} A \qquad A \xhookrightarrow{1/5} B, B \qquad A \xhookrightarrow{4/5} \varepsilon$$

$$\sigma_1: \quad B \xhookrightarrow{1/6} A, B \qquad B \xhookrightarrow{5/6} \varepsilon$$

$$\sigma_2: \quad B \xhookrightarrow{2/3} A \qquad B \xhookrightarrow{1/3} \varepsilon$$

$\longrightarrow$ Continuous-Time Markov Decision Process – Infinite-State

Generalize balance equations:

Subdivide λ_B in $\lambda_B = \lambda_{B,1} + \lambda_{B,2}$.

Intention: $\lambda_{B,i}$ = rate of B-tasks processed according to σ_i.

$$\lambda_A = \alpha_A + \lambda_{B,1} \cdot (1/6) + \lambda_{B,2} \cdot (2/3)$$

$$\lambda_{B,1} + \lambda_{B,2} = \cdots$$

Similarly to the uncontrolled case:

$$\exists \text{ stabilizing scheduler}$$

$$\iff \quad \exists \text{ solution } \boldsymbol{\lambda} \text{ with } \lambda_A < \mu_A \text{ and } \lambda_{B,1} + \lambda_{B,2} < \mu_B$$

Controlled Networks

$$0 \xhookrightarrow{1} A \qquad A \xhookrightarrow{1/5} B, B \qquad A \xhookrightarrow{4/5} \varepsilon$$

$$\sigma_1 : \quad B \xhookrightarrow{1/6} A, B \qquad B \xhookrightarrow{5/6} \varepsilon$$

$$\sigma_2 : \quad B \xhookrightarrow{2/3} A \qquad B \xhookrightarrow{1/3} \varepsilon$$

$\longrightarrow$Continuous-Time Markov Decision Process – Infinite-State

Generalize balance equations:

Subdivide λ_B in $\lambda_B = \lambda_{B,1} + \lambda_{B,2}$.

Intention: $\lambda_{B,i}$ = rate of B-tasks processed according to σ_i.

$$\lambda_A = \alpha_A + \lambda_{B,1} \cdot (1/6) + \lambda_{B,2} \cdot (2/3)$$

$$\lambda_{B,1} + \lambda_{B,2} = \cdots$$

Similarly to the uncontrolled case:

$$\exists \text{ stabilizing scheduler}$$

$$\iff \exists \text{ solution } \boldsymbol{\lambda} \text{ with } \lambda_A < \mu_A \text{ and } \lambda_{B,1} + \lambda_{B,2} < \mu_B \qquad \textbf{LP!!}$$

General Main Result

Theorem

Given a controlled BQN.

1. *It is decidable in polynomial time whether there exists an (arbitrary) stabilizing scheduler.*
2. *If it exists, one can compute in polynomial time a static randomized scheduler, which is stabilizing in a strong sense, i.e., in steady state there is an exponential moment of the total queue size.*

The theorem implies:
Any stabilizing scheduler can efficiently be made

- static and – at the same time –
- "strongly" stabilizing.

Summary

- Queueing networks can be used to model parallelism.
- Classical Jackson networks lack branching and control.
  $\longrightarrow$New model: Branching Queueing Networks
- Stability and existence of stabilizing schedulers
  can be determined in polynomial time.
- If a stabilizing scheduler does exist,
  a static randomized scheduler suffices
  and can be computed in polynomial time.

Summary

- Queueing networks can be used to model parallelism.
- Classical Jackson networks lack branching and control.
  $\longrightarrow$New model: Branching Queueing Networks
- Stability and existence of stabilizing schedulers
  can be determined in polynomial time.
- If a stabilizing scheduler does exist,
  a static randomized scheduler suffices
  and can be computed in polynomial time.

Future work:

- Performance beyond stability,
  e.g., long-term average queue size
  $\longrightarrow$Can non-static schedulers help to minimize it?

Thank you!