A Theory of System Fault Tolerance

Fossacs 06

Adrian Francalanza and Matthew Hennessy

adrian.francalanza@um.edu.mt matthew.hennessy@sussex.ac.uk

Aim of The Paper

- Formalise the notion of **Fault Tolerance** (in a distributed setting)
- Develop proof techniques to show fault-tolerance.

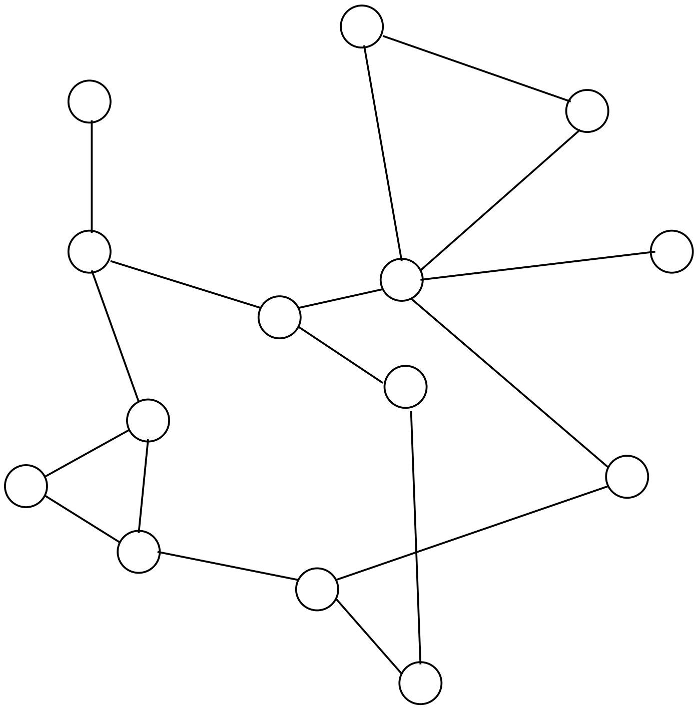
Talk Overview

- Fault Tolerance Intuitions
- Language
- Formal Definition
- Proof Techniques.

Talk Overview

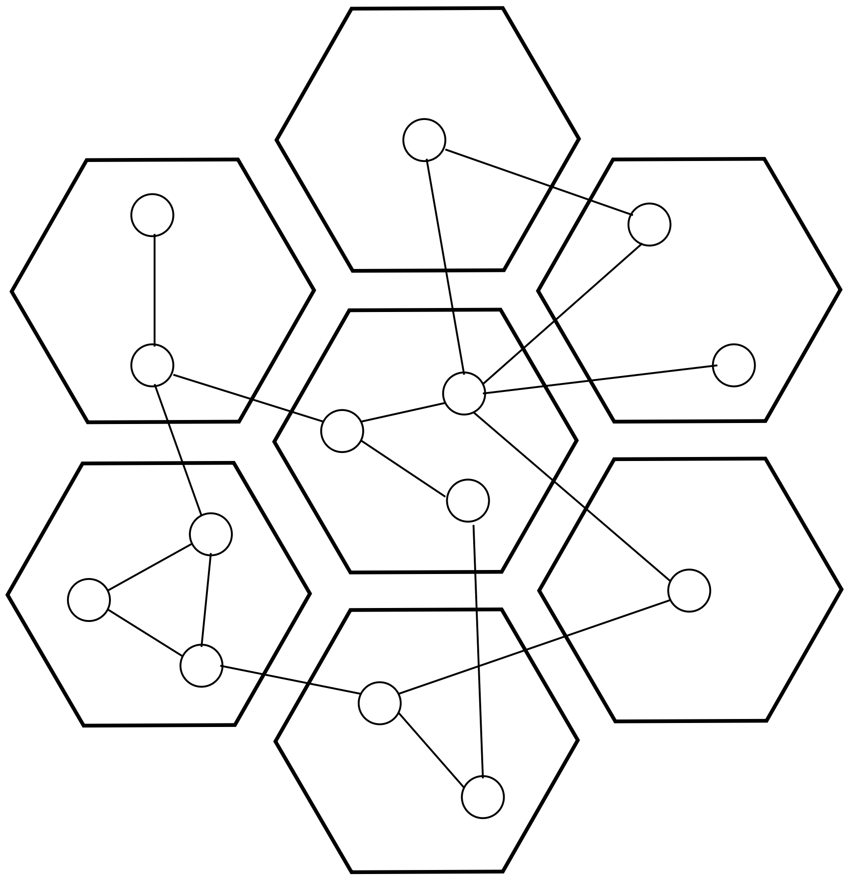
- Fault Tolerance Intuitions
- Language
- Formal Definition
- Proof Techniques.

Fault Tolerance



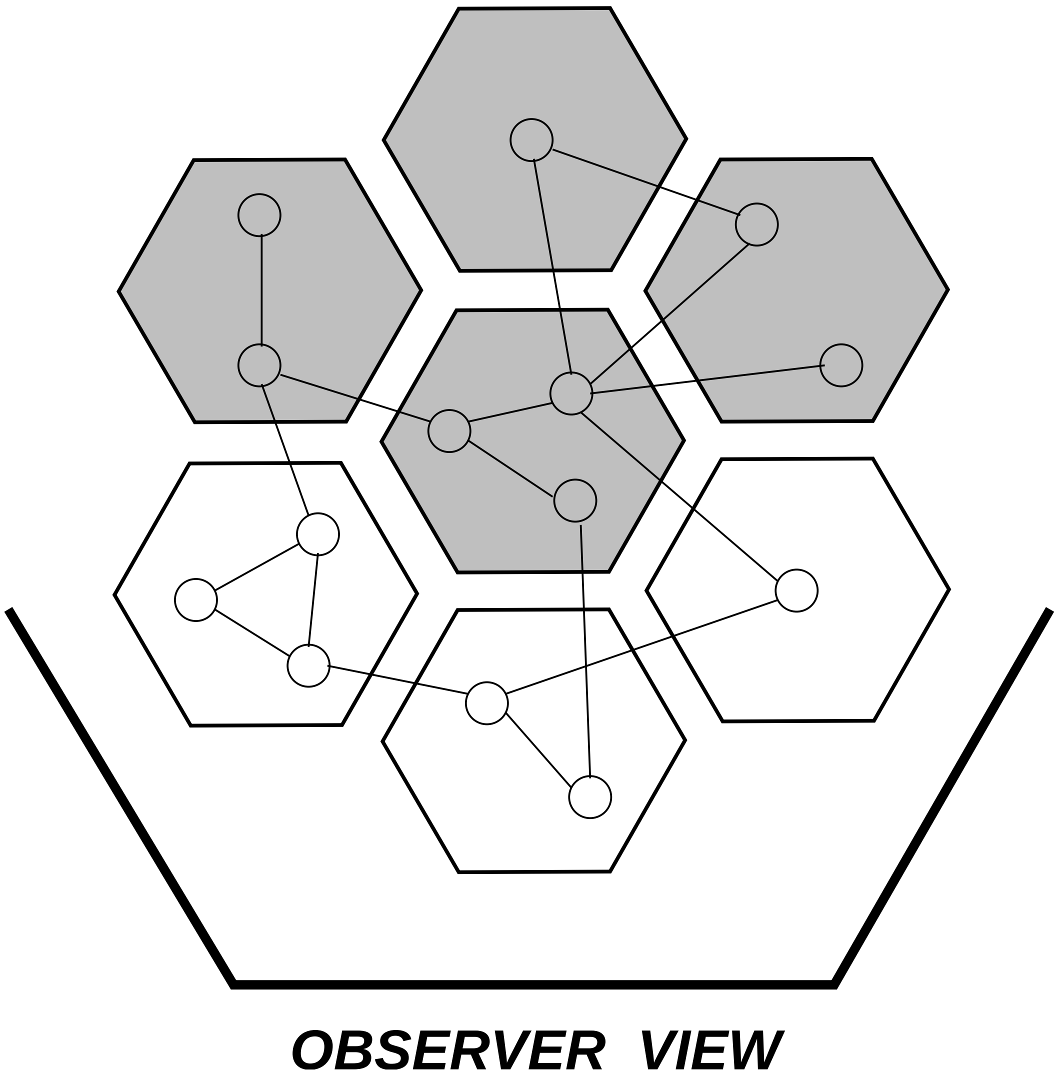
processes executing in parallel and interacting

Fault Tolerance



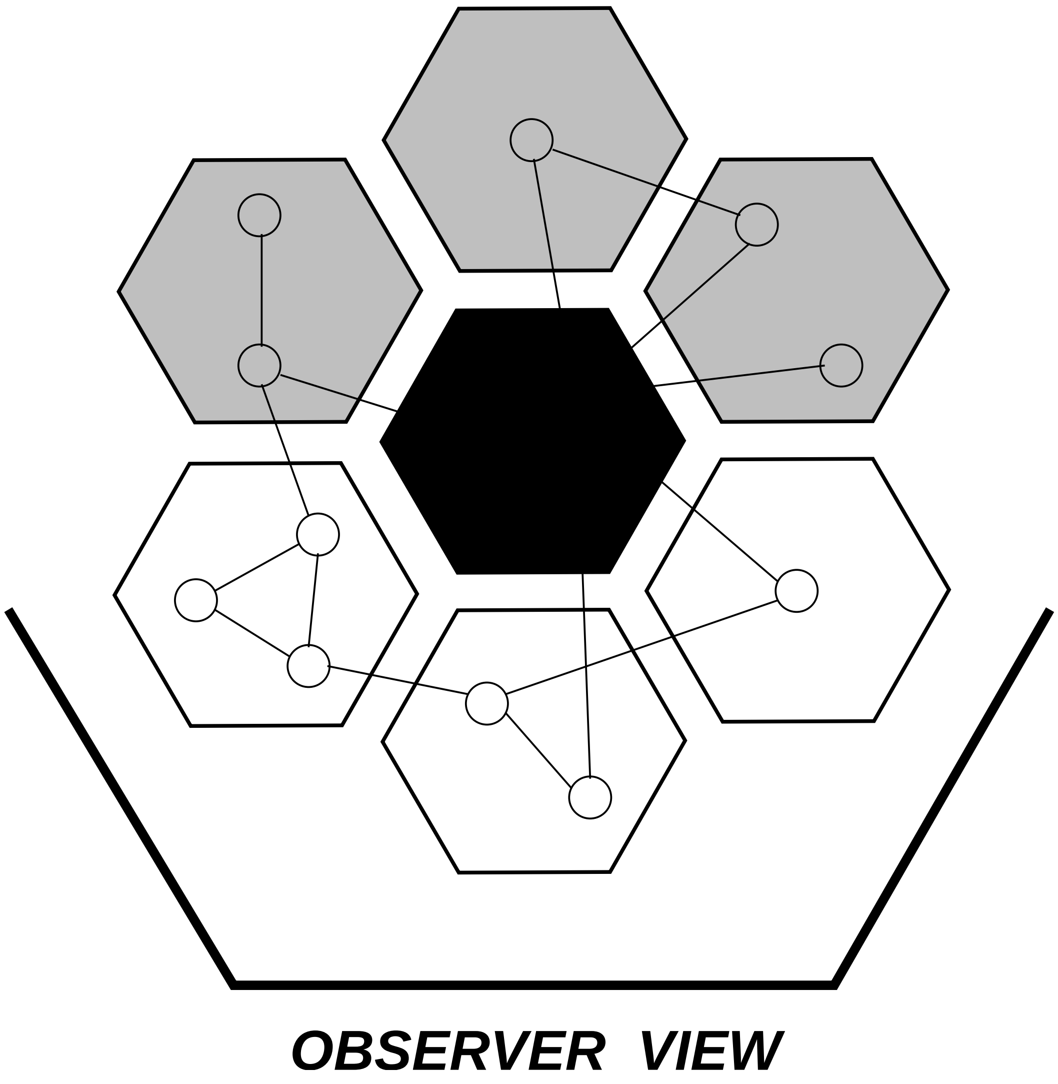
partitioned
across
container
units

Fault Tolerance



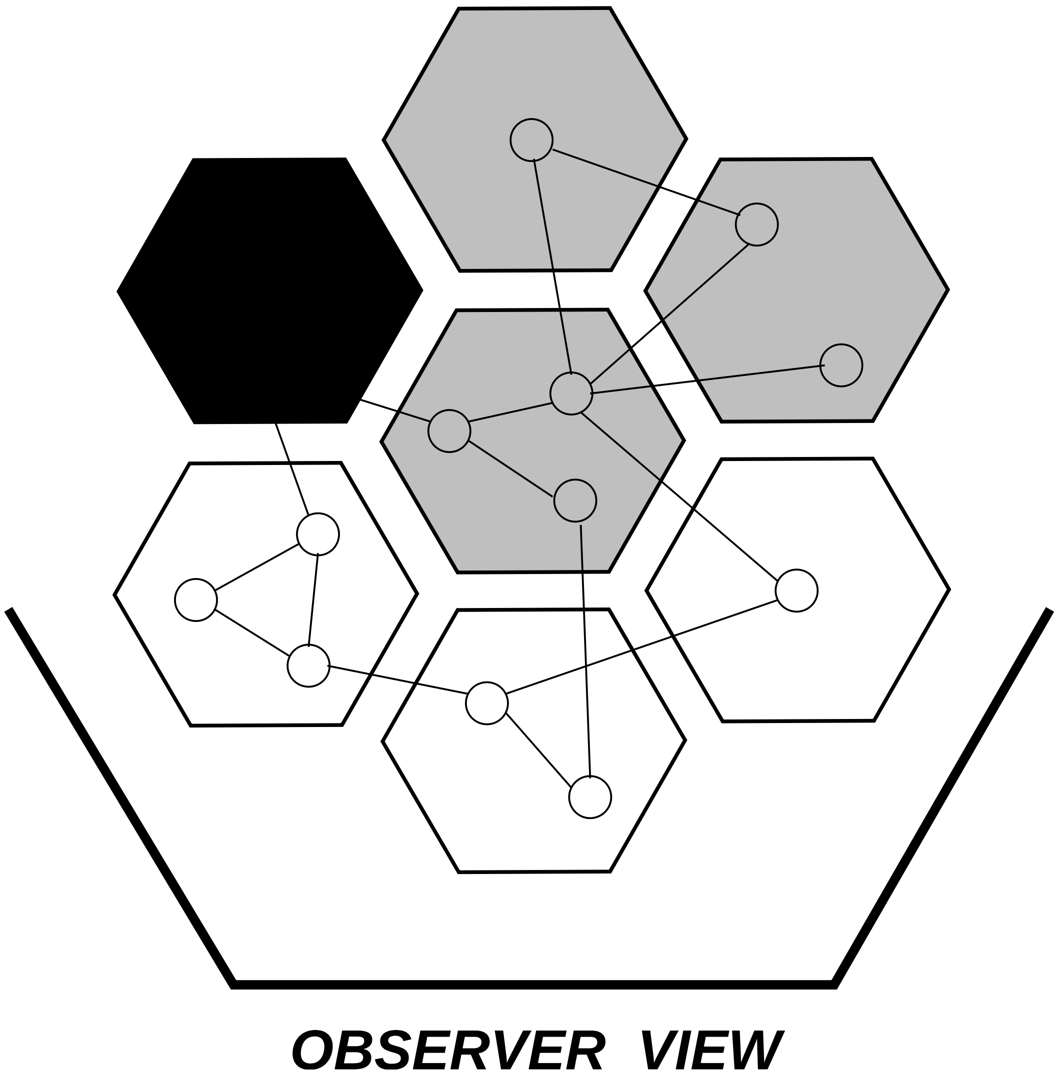
Observed
behaviour
is partial

Fault Tolerance



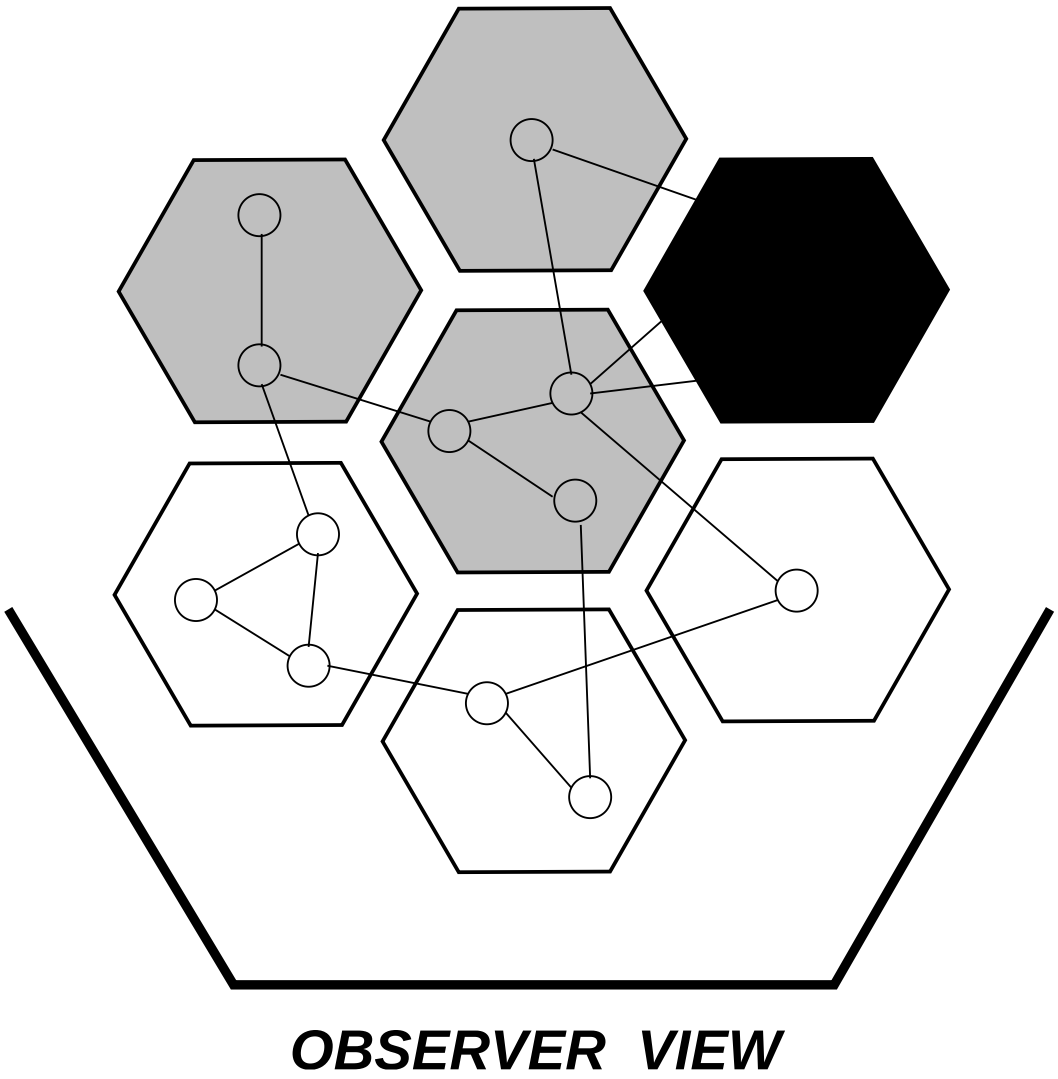
Observed behaviour preserved up to 1 failure

Fault Tolerance



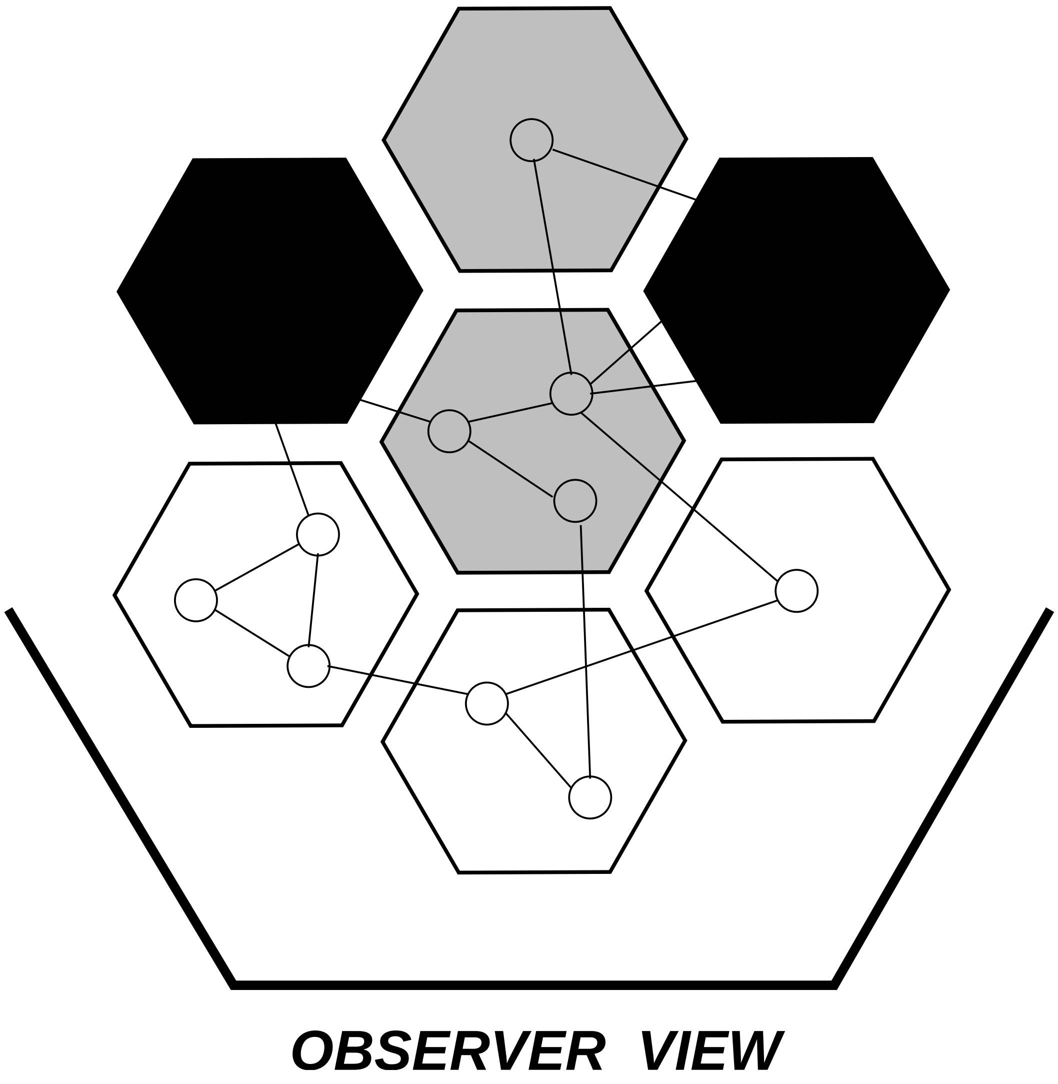
Observed behaviour preserved up to 1 failure

Fault Tolerance



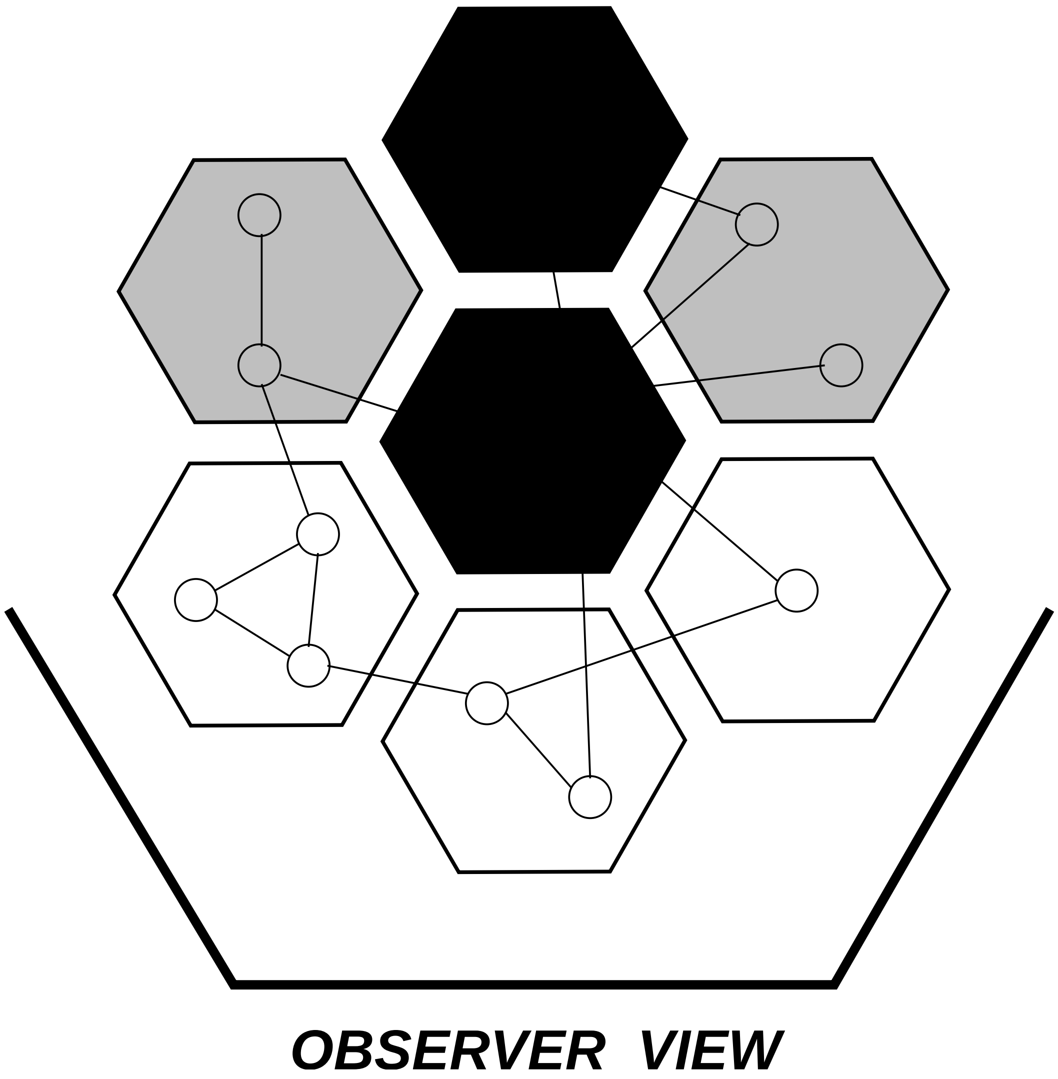
Observed behaviour preserved up to 1 failure

Fault Tolerance



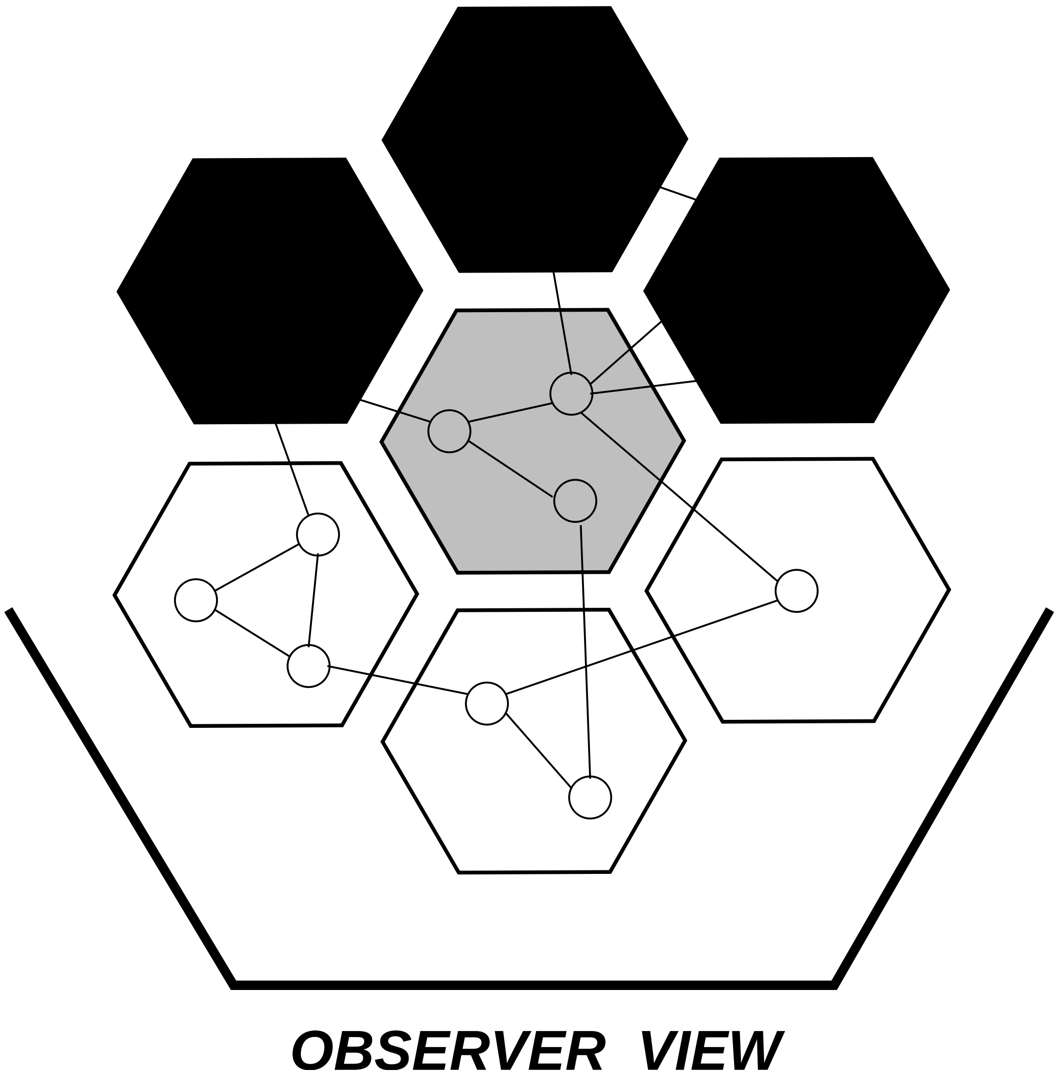
Observed behaviour preserved up to 2 failures

Fault Tolerance



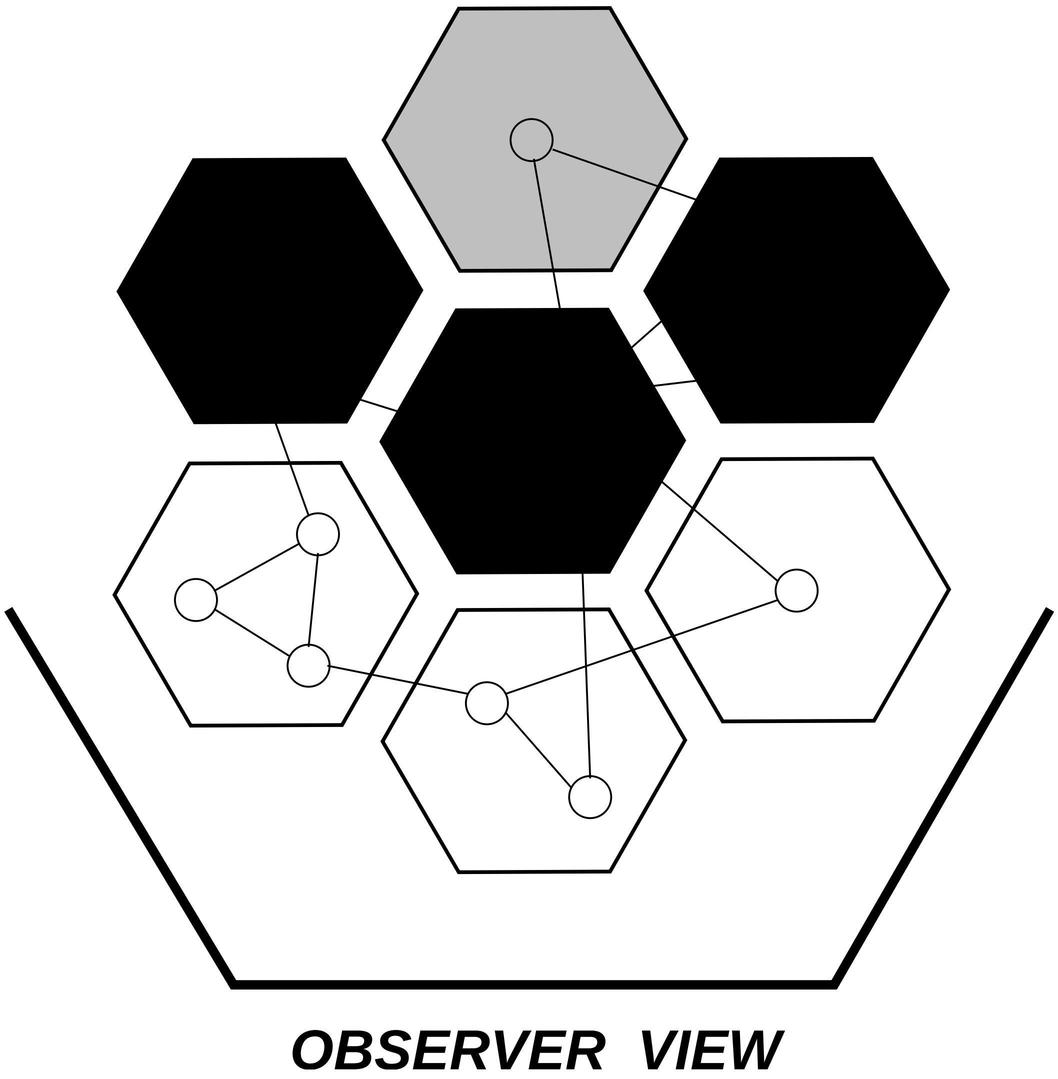
Observed
behaviour
preserved
up to 2 failures

Fault Tolerance



Observed behaviour preserved up to 3 failures

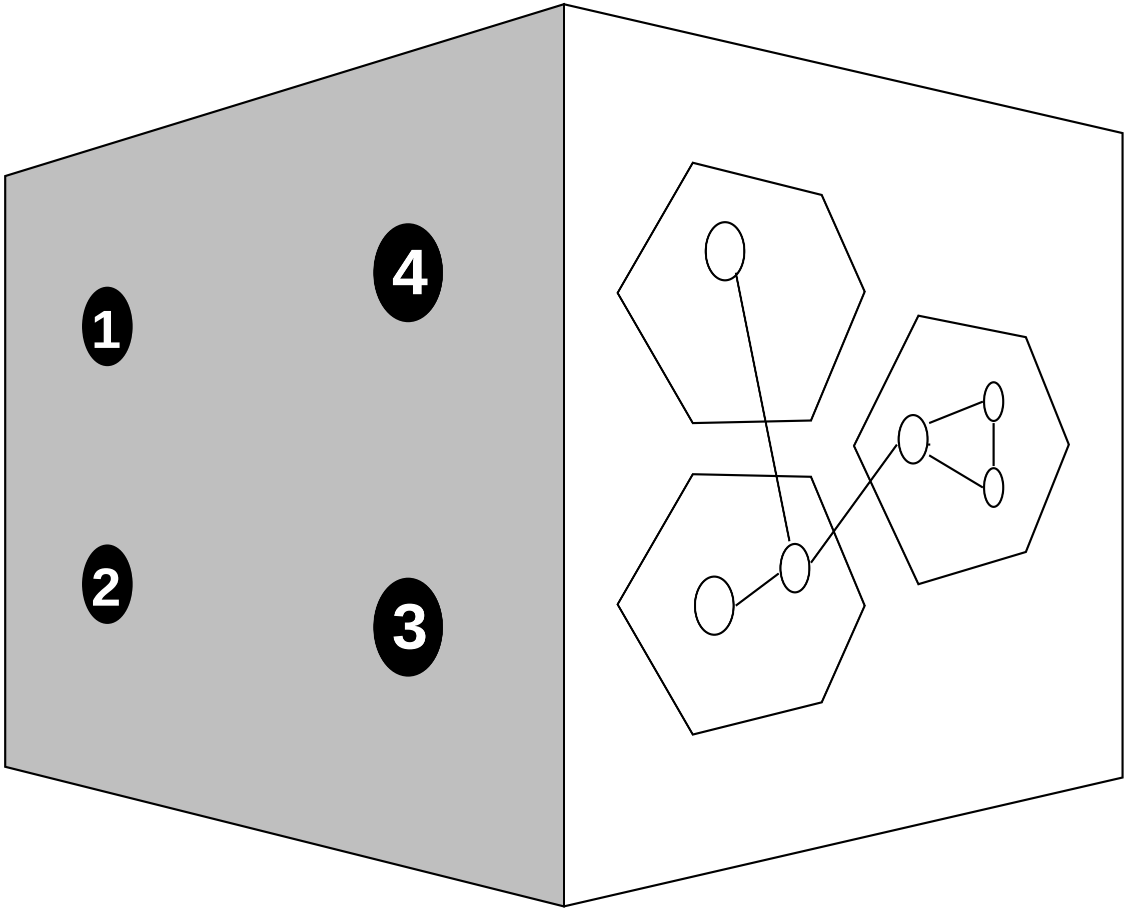
Fault Tolerance



Observed
behaviour
preserved
up to 3 failures

Fault Tolerance Analysis

Induce Faults



Observer View

Talk Overview

- Fault Tolerance Intuitions
- Language
- Formal Definition
- Proof Techniques.

The Language

Processes

$$
\begin{array}{rcll}
P, Q & ::= & u!\langle V\rangle.P & \mid u?(X).P \\
& \mid & \text{if } v = u \text{ then } P \text{ else } Q & \mid \ * u?(X).P \\
& \mid & (\nu\, n : \mathrm{T})\, P & \mid \text{go } u.P \\
& \mid & \mathbf{0} & \mid P|Q \\
& \mid & \text{ping } u.P \text{ else } Q &
\end{array}
$$

Systems

$$
\begin{array}{rcll}
M, N, O & ::= & l[\![P]\!] & \mid N|M \\
& \mid & (\nu\, n : \mathrm{T})N &
\end{array}
$$

The Language

Assuming $\Gamma \vdash l : \mathbf{alive}$

(r-comm)

$$\frac{}{\Gamma \triangleright l[\![a!\langle V\rangle.P]\!] \mid l[\![a?(X).Q]\!] \;\longrightarrow\; \Gamma \triangleright l[\![P]\!] \mid l[\![Q\{^{V}/X\}]\!]}$$

(r-go)

$$\frac{}{\Gamma \triangleright l[\![\text{go } k.P]\!] \longrightarrow \Gamma \triangleright k[\![P]\!]}\ \Gamma \vdash k : \mathbf{alive}$$

(r-ngo)

$$\frac{}{\Gamma \triangleright l[\![\text{go } k.P]\!] \longrightarrow \Gamma \triangleright k[\![\mathbf{0}]\!]}\ \Gamma \nvdash k : \mathbf{alive}$$

The Language

Assuming $\Gamma \vdash l : \textbf{alive}$

(r-ping)

$$\frac{}{\Gamma \triangleright l[\![\text{ping } k.P \text{ else } Q]\!] \longrightarrow \Gamma \triangleright l[\![P]\!]} \; \Gamma \vdash k : \textbf{alive}$$

(r-nping)

$$\frac{}{\Gamma \triangleright l[\![\text{ping } k.P \text{ else } Q]\!] \longrightarrow \Gamma \triangleright l[\![Q]\!]} \; \Gamma \nvdash k : \textbf{alive}$$

Examples

$$\mathsf{server}_1 \Leftarrow (\nu\, \mathit{data})\left(\begin{array}{l} l[\![\mathit{req}?(x,y).\ \mathsf{go}\ k_1.\ \mathit{data}!\langle x,y,l\rangle]\!] \\ \mid k_1[\![\mathit{data}?(x,y,z).\ \mathsf{go}\ z.\ y!\langle f(x)\rangle]\!] \end{array}\right)$$

Examples

$$\mathsf{server}_1 \;\Leftarrow\; (\nu\, \mathit{data})\left(\begin{array}{l} l[\![\mathit{req}?(x,y).\ \mathsf{go}\ k_1.\ \mathit{data}!\langle x,y,l\rangle]\!] \\ \mid k_1[\![\mathit{data}?(x,y,z).\ \mathsf{go}\ z.\ y!\langle f(x)\rangle]\!] \end{array}\right)$$

Examples

$$\text{server}_1 \;\Leftarrow\; (\nu\, \mathit{data})\left(\begin{array}{l} l[\![\mathit{req}?(x,y).\ \text{go}\ k_1.\ \mathit{data}!\langle x,y,l\rangle]\!] \\ \mid k_1[\![\mathit{data}?(x,y,z).\ \text{go}\ z.\ y!\langle f(x)\rangle]\!] \end{array}\right)$$

Examples

$$\mathsf{server}_2 \;\Leftarrow\; (\nu\, data)\left(\begin{array}{l} l\left[\!\!\left[req?(x,y).\,(\nu \mathsf{s})\left(\begin{array}{l} \mathsf{go}\; k_1.data!\langle x,\mathsf{s},l\rangle \\ \mid \mathsf{go}\; k_2.data!\langle x,\mathsf{s},l\rangle \\ \mid \mathsf{s}?(x).y!\langle x\rangle \end{array}\right)\right]\!\!\right] \\ \mid k_1[\![data?(x,y,z).\mathsf{go}\; z.y!\langle f(x)\rangle]\!] \\ \mid k_2[\![data?(x,y,z).\mathsf{go}\; z.y!\langle f(x)\rangle]\!] \end{array}\right)$$

Examples

$$\text{server}_2 \;\Leftarrow\; (\nu\, \mathit{data}) \left(\begin{array}{l} l \left[\!\left[\mathit{req}?(x,y).\,(\nu \mathsf{s}) \left(\begin{array}{l} \text{go } k_1.\mathit{data}!\langle x, \mathsf{s}, l\rangle \\ \mid \text{go } k_2.\mathit{data}!\langle x, \mathsf{s}, l\rangle \\ \mid \mathit{s}?(x).y!\langle x\rangle \end{array} \right) \right]\!\right] \\ \mid k_1[\![\mathit{data}?(x,y,z).\text{go } z.y!\langle f(x)\rangle]\!] \\ \mid k_2[\![\mathit{data}?(x,y,z).\text{go } z.y!\langle f(x)\rangle]\!] \end{array} \right)$$

Examples

$$\mathsf{server}_2 \;\Leftarrow\; (\nu\, data)\left(\begin{array}{l} l\left[\!\!\left[req?(x,y).\,(\nu s)\left(\begin{array}{l} \mathsf{go}\; k_1.data!\langle x,s,l\rangle \\ \mid \mathsf{go}\; k_2.data!\langle x,s,l\rangle \\ \mid s?(x).y!\langle x\rangle \end{array}\right)\right]\!\!\right] \\ \mid k_1[\![data?(x,y,z).\mathsf{go}\; z.y!\langle f(x)\rangle]\!] \\ \mid k_2[\![data?(x,y,z).\mathsf{go}\; z.y!\langle f(x)\rangle]\!] \end{array}\right)$$

Examples

$$\mathsf{server}_3 \Leftarrow (\nu\, data)\left(\begin{array}{l} l\left[\!\left[req?(x,y).\,(\nu s)\left(\begin{array}{l} \mathsf{go}\; k_1.data!\langle x, s, l\rangle \\ \mid \mathsf{go}\; k_2.data!\langle x, s, l\rangle \\ \mid \mathsf{go}\; k_3.data!\langle x, s, l\rangle \\ \mid s?(x).y!\langle x\rangle \end{array}\right)\right]\!\right] \\ \mid k_1[\![data?(x,y,z).\mathsf{go}\; z.y!\langle f(x)\rangle]\!] \\ \mid k_2[\![data?(x,y,z).\mathsf{go}\; z.y!\langle f(x)\rangle]\!] \\ \mid k_3[\![data?(x,y,z).\mathsf{go}\; z.y!\langle f(x)\rangle]\!] \end{array}\right)$$

Examples

$$\mathrm{server}_3 \Leftarrow (\nu\, data)\left(\begin{array}{l} l\left[\!\left[req?(x,y).\,(\nu s)\left(\begin{array}{l} \mathrm{go}\ k_1.data!\langle x,s,l\rangle \\ \mid \mathrm{go}\ k_2.data!\langle x,s,l\rangle \\ \mid \mathrm{go}\ k_3.data!\langle x,s,l\rangle \\ \mid s?(x).y!\langle x\rangle \end{array}\right)\right]\!\right] \\ \mid k_1[\![data?(x,y,z).\mathrm{go}\ z.y!\langle f(x)\rangle]\!] \\ \mid k_2[\![data?(x,y,z).\mathrm{go}\ z.y!\langle f(x)\rangle]\!] \\ \mid k_3[\![data?(x,y,z).\mathrm{go}\ z.y!\langle f(x)\rangle]\!] \end{array}\right)$$

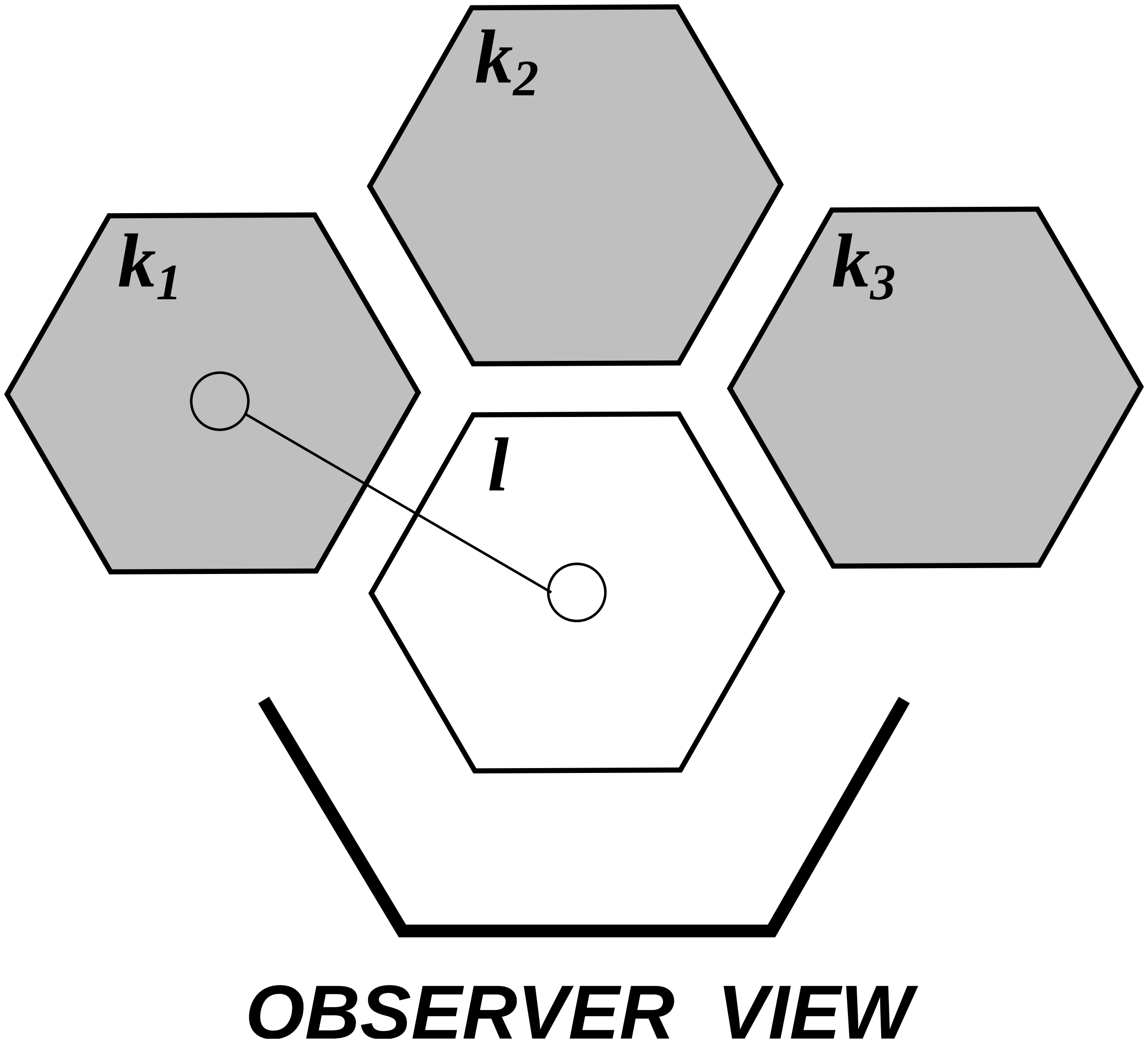
Examples

$$\mathsf{sPing} \Leftarrow (\nu\, data)\left(\begin{array}{l} l\left[\!\!\left[\begin{array}{r} serv?(x,y).\mathsf{ping}\, k_1.\mathsf{go}\, k_1.data!\langle x,y,l\rangle \\ \mathsf{else\ go}\, k_2.data!\langle x,y,l\rangle \end{array}\right]\!\!\right] \\ \mid k_1[\![data?(x,y,z).\mathsf{go}\, z\, .y!\langle f(x)\rangle]\!] \\ \mid k_2[\![data?(x,y,z).\mathsf{go}\, z\, .y!\langle f(x)\rangle]\!] \end{array}\right)$$

Examples

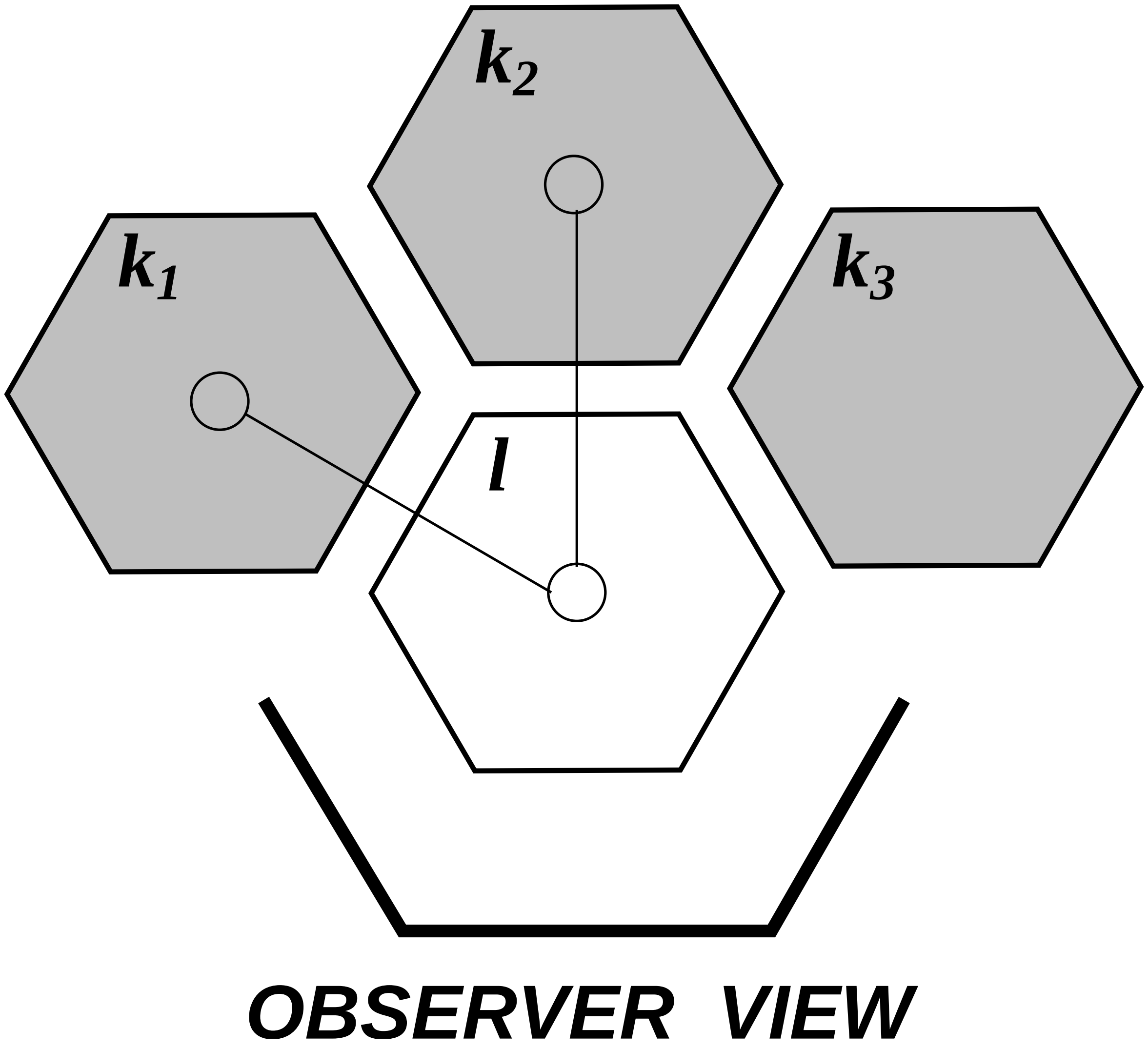
$$\mathsf{sPing} \;\Leftarrow\; (\nu\, data)\left(\begin{array}{l} l\left[\!\!\left[\begin{array}{r} serv?(x,y).\mathsf{ping}\, k_1.\mathsf{go}\, k_1.data!\langle x,y,l\rangle \\ \mathsf{else}\ \mathsf{go}\, k_2.data!\langle x,y,l\rangle \end{array}\right]\!\!\right] \\ \mid k_1[\![data?(x,y,z).\mathsf{go}\, z\ .y!\langle f(x)\rangle]\!] \\ \mid k_2[\![data?(x,y,z).\mathsf{go}\, z\ .y!\langle f(x)\rangle]\!] \end{array}\right)$$

Server Fault Tolerance



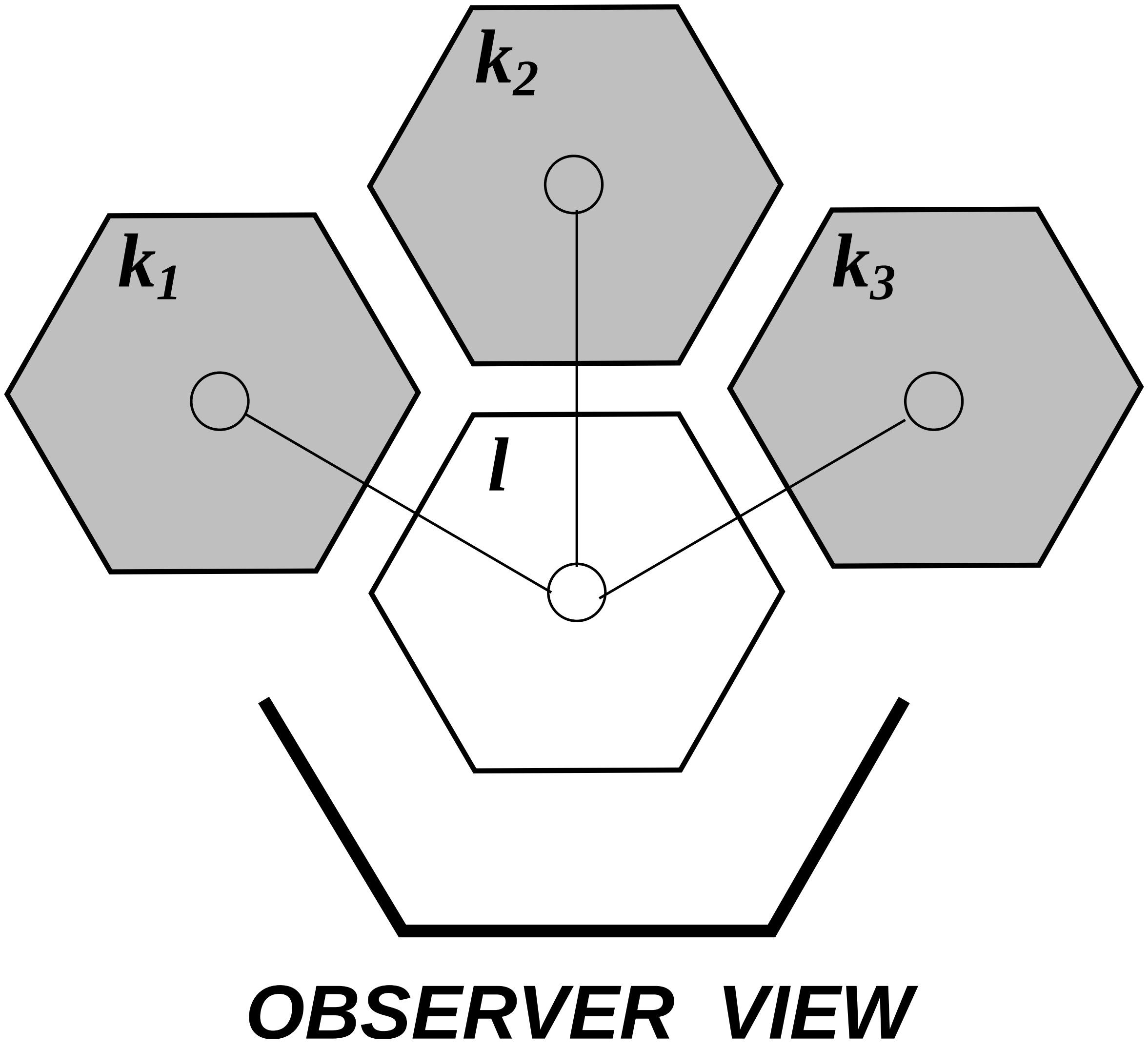
$$\mathsf{server}_1 \;=\; (\nu\, \mathit{data})\left(\begin{array}{l} l[\![\ldots]\!] \\ \mid k_1[\![\ldots]\!] \end{array}\right)$$

Server Fault Tolerance



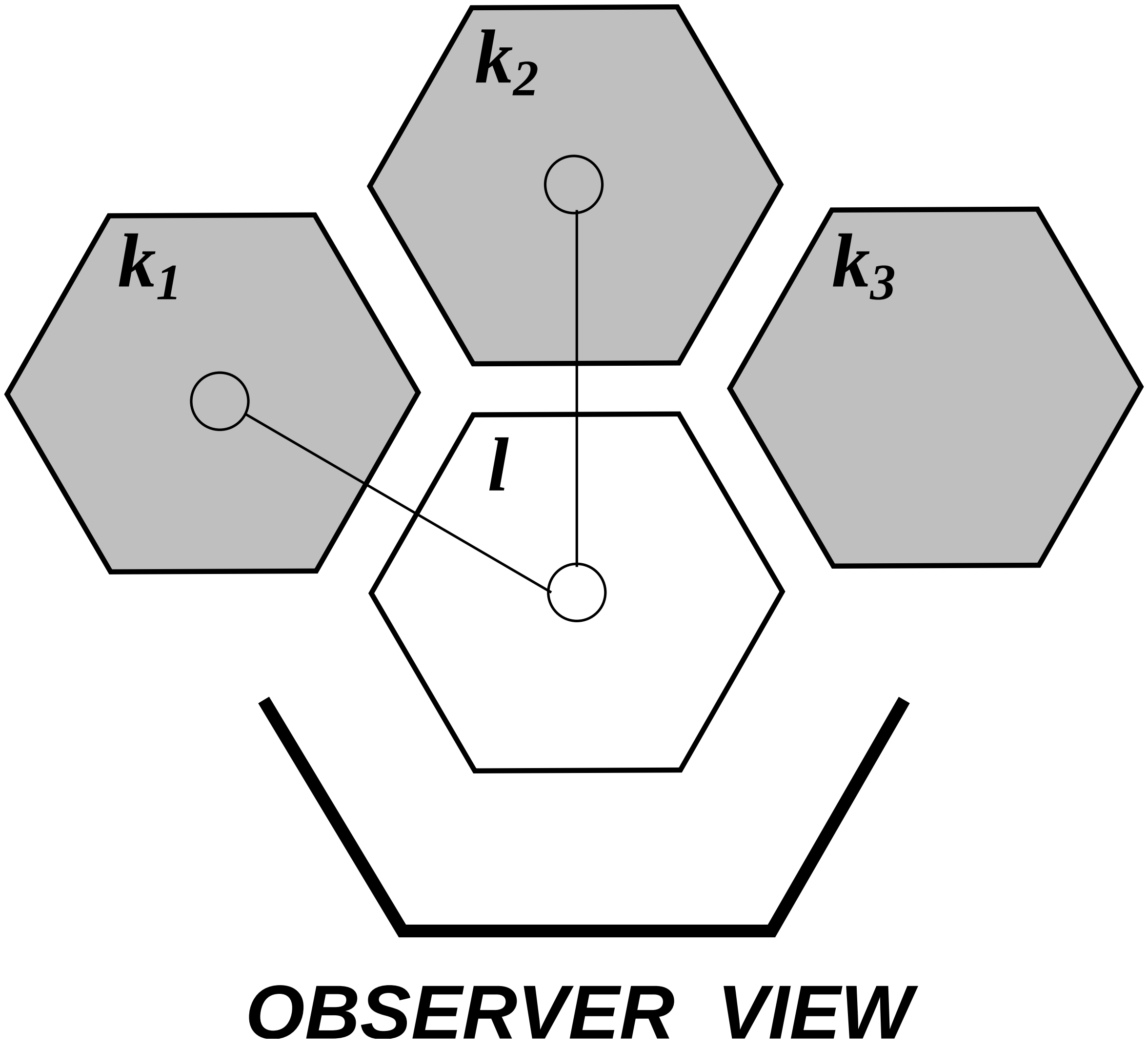
$$\mathsf{server}_2 \;=\; (\nu\, \mathit{data})\begin{pmatrix} l[\![\ldots]\!] \\ \mid k_1[\![\ldots]\!] \\ \mid k_2[\![\ldots]\!] \end{pmatrix}$$

Server Fault Tolerance



$$\text{server}_2 \;=\; (\nu\, data)\left(\begin{array}{l} l[\![\ldots]\!] \\ \mid k_1[\![\ldots]\!] \\ \mid k_2[\![\ldots]\!] \\ \mid k_3[\![\ldots]\!] \end{array}\right)$$

Server Fault Tolerance



$$\mathsf{sPing} \;=\; (\nu\, \mathit{data})\begin{pmatrix} l[\![\ldots]\!] \\ \mid k_1[\![\ldots]\!] \\ \mid k_2[\![\ldots]\!] \end{pmatrix}$$

Talk Overview

- Fault Tolerance Intuitions
- Language
- Formal Definition
- Proof Techniques

Defining Fault Tolerance Preliminaries

- We partition Γ into two sets of live locations $\langle\mathcal{R},\mathcal{U}\rangle$

Defining Fault Tolerance Preliminaries

- We partition Γ into two sets of live locations $\langle \mathcal{R}, \mathcal{U} \rangle$
  Reliable: denoted by $\mathcal{R}$. They are **immortal**!

Defining Fault Tolerance Preliminaries

- We partition Γ into two sets of live locations $\langle \mathcal{R}, \mathcal{U} \rangle$

  Reliable: denoted by $\mathcal{R}$. They are **immortal**!

  Unreliable: denoted by $\mathcal{U}$. They **may fail**!

Defining Fault Tolerance Preliminaries

- We partition Γ into two sets of live locations $\langle \mathcal{R}, \mathcal{U} \rangle$
  Reliable: denoted by $\mathcal{R}$. They are **immortal**!
  Unreliable: denoted by $\mathcal{U}$. They **may fail**!
- We limit observations to reliable locations

Defining Fault Tolerance Preliminaries

- We partition Γ into two sets of live locations $\langle \mathcal{R}, \mathcal{U} \rangle$

  Reliable: denoted by $\mathcal{R}$. They are **immortal**!

  Unreliable: denoted by $\mathcal{U}$. They **may fail**!

- We limit observations to reliable locations

  Contexts: for all $[-] \,|\, N$ we have $\mathbf{fl}(N) \subseteq \mathcal{R}$

Defining Fault Tolerance Preliminaries

- We partition Γ into two sets of live locations $\langle \mathcal{R}, \mathcal{U} \rangle$

  Reliable: denoted by $\mathcal{R}$. They are **immortal**!

  Unreliable: denoted by $\mathcal{U}$. They **may fail**!

- We limit observations to reliable locations

  Contexts: for all $[-] \,|\, N$ we have $\mathbf{fl}(N) \subseteq \mathcal{R}$

  Barbs: $\Gamma \triangleright M \Downarrow_{a@l}$ iff $\Gamma \triangleright M \longrightarrow^{*} \equiv \Gamma \triangleright (\nu \tilde{n})\, M | l[\![a!\langle V \rangle.P]\!]$
  where $l, a \notin \tilde{n}$ and $l \in \mathcal{R}$

Defining Fault Tolerance Preliminaries

- We partition Γ into two sets of live locations $\langle \mathcal{R}, \mathcal{U} \rangle$

  Reliable: denoted by $\mathcal{R}$. They are **immortal**!

  Unreliable: denoted by $\mathcal{U}$. They **may fail**!

- We limit observations to reliable locations

  Contexts: for all $[-] \,|\, N$ we have $\mathbf{fl}(N) \subseteq \mathcal{R}$

  Barbs: $\Gamma \triangleright M \Downarrow_{a@l}$ iff $\Gamma \triangleright M \longrightarrow^{*} \equiv \Gamma \triangleright (\nu \tilde{n})\, M | l[\![a!\langle V \rangle.P]\!]$
  where $l, a \notin \tilde{n}$ and $l \in \mathcal{R}$

- We define reduction barbed congruence $\cong$ for configurations with the same reliable network $\mathcal{R}$

$$\langle \mathcal{R}, \mathcal{U} \rangle \triangleright M \quad \cong \quad \langle \mathcal{R}, \mathcal{U}' \rangle \triangleright N$$

Definition Fault Tolerance

Inducing Faults

Static: $\langle \mathcal{R}, \mathcal{U} \rangle - l \;=\; \langle \mathcal{R}, \mathcal{U}/\{l\} \rangle$

Dynamic:

$$\text{(r-kill)}\;\frac{}{\Gamma \triangleright l[\![\mathsf{kill}]\!] \longrightarrow (\Gamma - l) \triangleright l[\![\mathbf{0}]\!]}$$

Fault Contexts

Static: $F^n_S(\Gamma) \;=\; \Gamma - l_1 \ldots - l_n$

Dynamic: $F^n_D(M) \;=\; M \,|\, l_1[\![\mathsf{kill}]\!] \,|\, \ldots \,|\, l_n[\![\mathsf{kill}]\!]$

Definition Fault Tolerance

Static Fault Tolerance

$\Gamma \triangleright M$ is statically fault tolerant up to n faults if for any $F^n_S(-)$ we have

$$\Gamma \triangleright M \quad \cong \quad F^n_S(\Gamma) \triangleright M$$

Dynamic Fault Tolerance

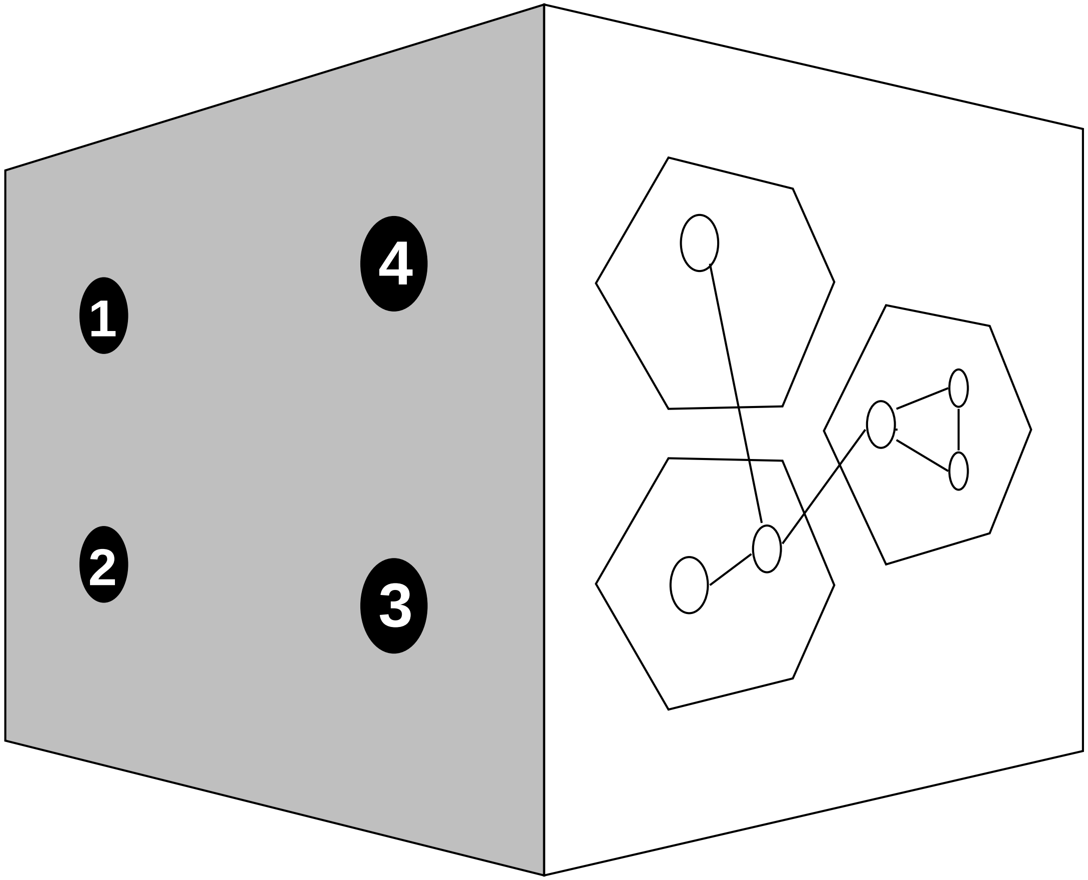
$\Gamma \triangleright M$ is dynamically fault tolerant up to n faults if for any $F^n_D(-)$ we have

$$\Gamma \triangleright M \quad \cong \quad \Gamma \triangleright F^n_D(M)$$

Examples

Recall...

Induce Faults



Observer View

Examples

Good to show **negative** results. Assuming $\Gamma = \langle \{l\}, \{k_1, k_2, k_3\} \rangle$:

- $\Gamma \triangleright \mathsf{server}_1$ is **not** 1-statically fault tolerant because

$$\Gamma \triangleright \mathsf{server}_1 \;\not\cong\; \Gamma - k_1 \triangleright \mathsf{server}_1$$

- $\Gamma \triangleright \mathsf{server}_2$ is **not** 2-dynamically fault tolerant because

$$\Gamma \triangleright \mathsf{server}_2 \;\not\cong\; \Gamma \triangleright \mathsf{server}_2 \,|\, k_1[\![\mathsf{kill}]\!] \,|\, k_2[\![\mathsf{kill}]\!]$$

- $\Gamma \triangleright \mathsf{sPing}$ is **not** 1-dynamically fault tolerant because

$$\Gamma \triangleright \mathsf{sPing} \;\not\cong\; \Gamma \triangleright \mathsf{sPing} \,|\, k_1[\![\mathsf{kill}]\!]$$

Examples

Hard to prove positive results:

It is difficult to prove that $\Gamma \triangleright \text{server}_2$ is 1-dynamic fault tolerant because:

1. $\cong$ **quantifies over all** valid **contexts**.
2. Dynamic fault tolerance definition **quantifies over all fault contexts**, amongst which there is **considerable overlap**.
3. There are a number of **confluent** reductions that increase the burden of our analysis.

Talk Overview

- Fault Tolerance Intuitions
- Language
- Formal Definition
- Proof Techniques

Problems we need to address

Hard to prove positive results with our fault tolerance definition because:

1. $\cong$ **quantifies over all** valid **contexts**.
2. Dynamic fault tolerance definition **quantifies over all fault contexts**, amongst which there is **considerable overlap**.
3. There are a number of **confluent** reductions that increase the burden of our analysis.

Solving Observer Quantification

- Define lts over configurations

(l-out)

$$\frac{}{\langle \mathcal{R}, \mathcal{U} \rangle \triangleright l[\![a!\langle V \rangle.P]\!] \xrightarrow{l:a!\langle V \rangle} \langle \mathcal{R}, \mathcal{U} \rangle \triangleright l[\![P]\!]} \; l \in \mathcal{R}$$

Solving Observer Quantification

- Define lts over configurations

(l-out)

$$\frac{}{\langle \mathcal{R}, \mathcal{U} \rangle \triangleright l[\![a!\langle V \rangle.P]\!] \xrightarrow{l:a!\langle V \rangle} \langle \mathcal{R}, \mathcal{U} \rangle \triangleright l[\![P]\!]} \; l \in \mathcal{R}$$

- Define bisimulation, $\approx$, for configurations based on lts

Solving Observer Quantification

- Define lts over configurations

(l-out)

$$\frac{}{\langle \mathcal{R}, \mathcal{U} \rangle \triangleright l[\![a!\langle V \rangle.P]\!] \xrightarrow{l:a!\langle V \rangle} \langle \mathcal{R}, \mathcal{U} \rangle \triangleright l[\![P]\!]} \; l \in \mathcal{R}$$

- Define bisimulation, $\approx$, for configurations based on lts
- Prove **Soundness**:

$$\langle \mathcal{R}, \mathcal{U} \rangle \triangleright M \approx \langle \mathcal{R}, \mathcal{U}' \rangle \triangleright N$$

implies

$$\langle \mathcal{R}, \mathcal{U} \rangle \triangleright M \cong \langle \mathcal{R}, \mathcal{U}' \rangle \triangleright N$$

Problems we need to address

Hard to prove positive results with our fault tolerance definition because:

1. $\cong$ **quantifies over all** valid **contexts**.
2. Dynamic fault tolerance definition **quantifies over all fault contexts**, amongst which there is **considerable overlap**.
3. There are a number of **confluent** reductions that increase the burden of our analysis.

Fault Context Quantification and Overlap (1)

...recall

Dynamic Fault Tolerance

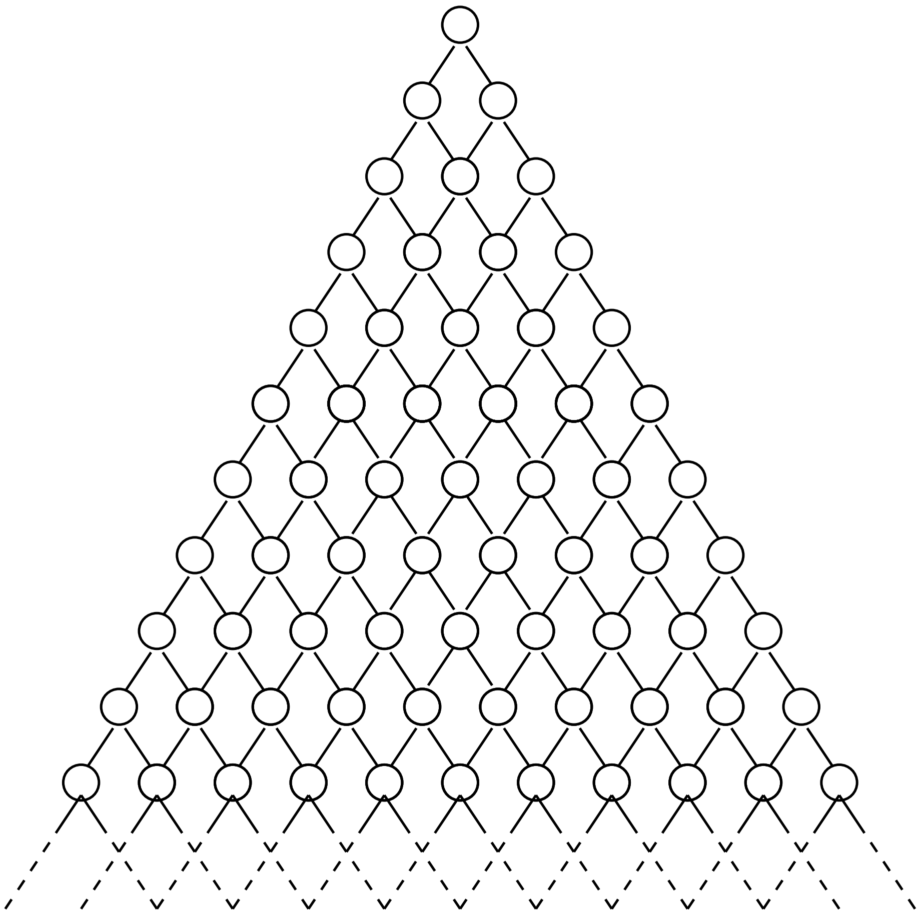
$\Gamma \triangleright M$ is dynamically fault tolerant up to n faults if for any $F^n_D(-)$ we have

$$\Gamma \triangleright M \quad \cong \quad \Gamma \triangleright F^n_D(M)$$

Thus **for every** $F^n_D(-)$ we have to show

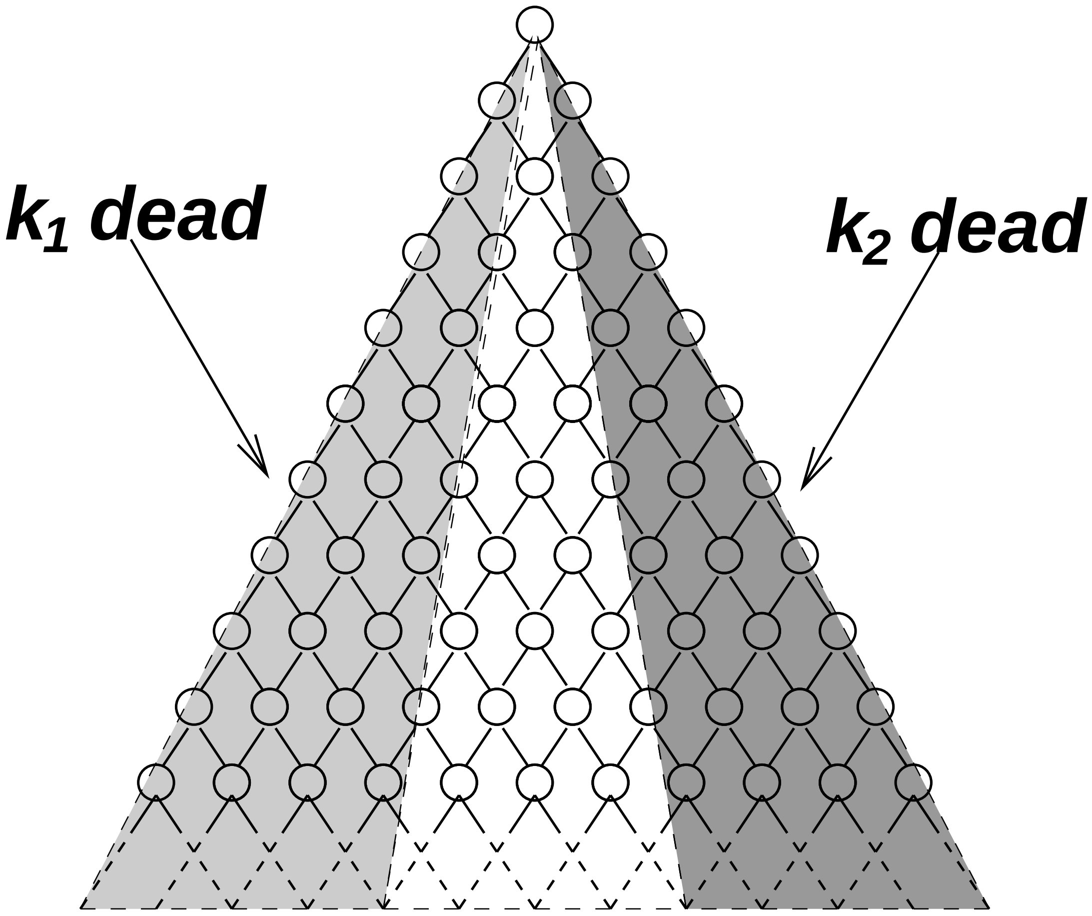
$$\Gamma \triangleright M \;\approx\; \Gamma \triangleright F^n_D(M)$$

Fault Context Quantification and Overlap (1)



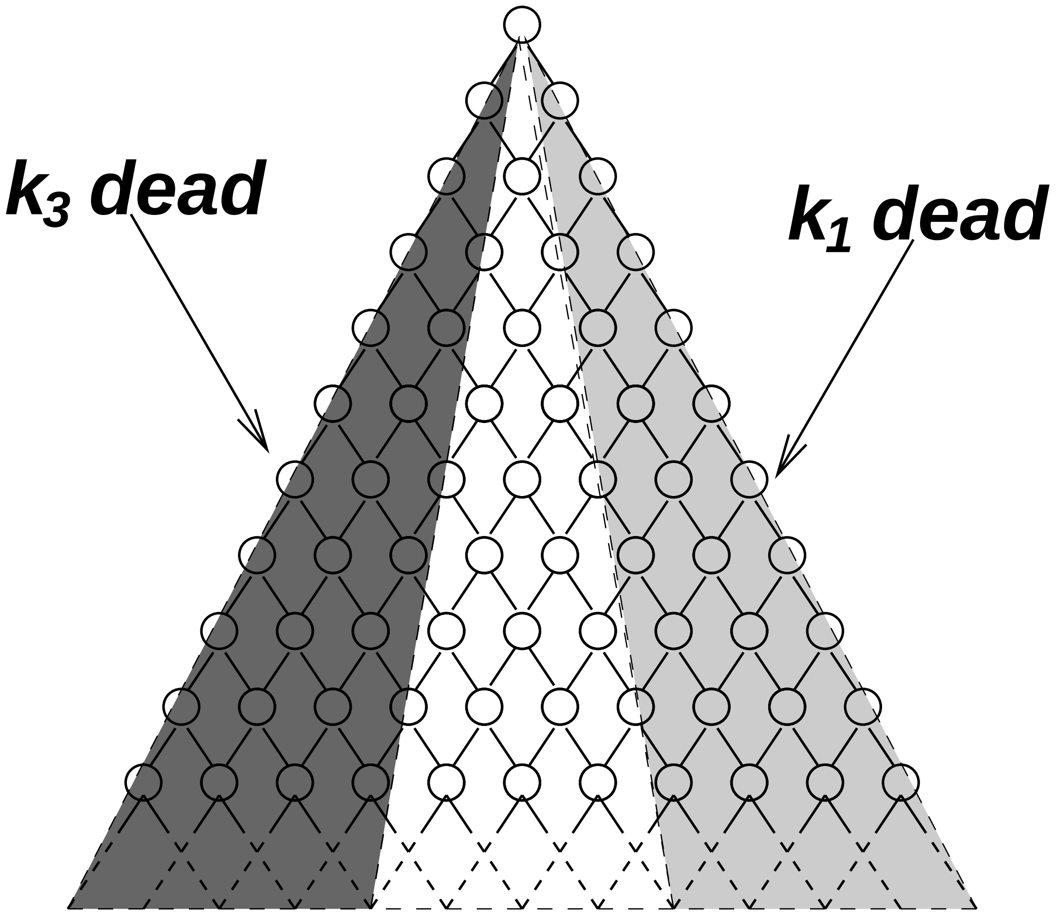
Nodes are bisimular tuples. Edges are transitions.

Fault Context Quantification and Overlap (1)



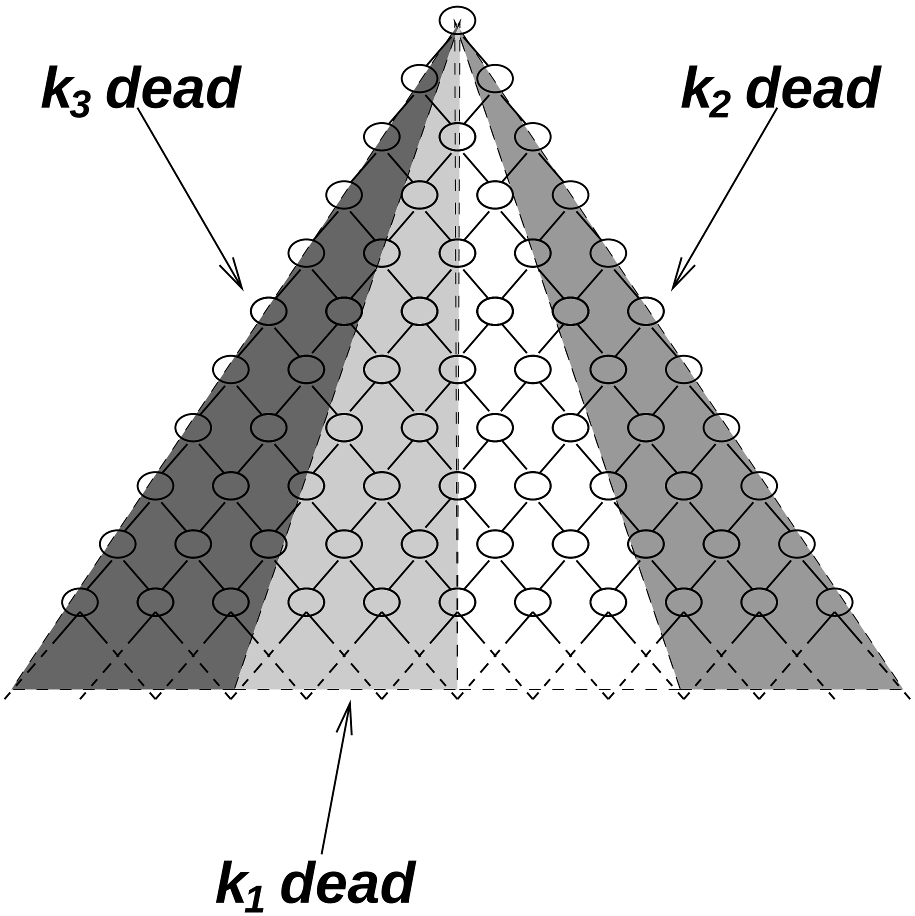
$$F_D^2 = \begin{cases} \text{kill}(k_1) \\ \mid \text{kill}(k_2) \\ \mid [-] \end{cases}$$

Fault Context Quantification and Overlap (1)



$$F_D^2 = \begin{cases} \text{kill}(k_1) \\ \mid \text{kill}(k_3) \\ \mid [-] \end{cases}$$

Fault Context Quantification and Overlap (1)



Merging the two relations

Fault Context Quantification and Overlap (2)

Define new transition that **counts** failures

(l-fail)

$$\frac{}{\langle \mathcal{R}, \mathcal{U} \rangle \triangleright N \xrightarrow{\text{fail}} \langle \mathcal{R}, \mathcal{U} \rangle - l \triangleright N} \; l \in \mathcal{U}$$

Fault Context Quantification and Overlap (2)

Define **Fault Tolerant Simulation**, $\preceq_D^n$, the largest *asymmetric* relation over configurations such that $\Gamma_1 \triangleright M_1 \preceq_D^n \Gamma_2 \triangleright M_2$ implies

- $\Gamma_1 \triangleright M_1 \xrightarrow{\gamma} \Gamma_1' \triangleright M_1'$ implies $\Gamma_2 \triangleright M_2 \overset{\widehat{\gamma}}{\Longrightarrow} \Gamma_2' \triangleright M_2'$ such that $\Gamma_1' \triangleright M_1' \preceq_D^n \Gamma_2' \triangleright M_2'$
- $\Gamma_2 \triangleright M_2 \xrightarrow{\gamma} \Gamma_2' \triangleright M_2'$ implies $\Gamma_1 \triangleright M_1 \overset{\widehat{\gamma}}{\Longrightarrow} \Gamma_1' \triangleright M_1'$ such that $\Gamma_1' \triangleright N_1' \preceq_D^n \Gamma_2' \triangleright M_2'$
- if $n > 0$, $\Gamma_2 \triangleright M_2 \xrightarrow{\text{fail}} \Gamma_2' \triangleright M_2'$ implies $\Gamma_1 \triangleright M_1 \Longrightarrow \Gamma_1' \triangleright M_1'$ such that $\Gamma_1' \triangleright M_1' \preceq_D^{n-1} \Gamma_2' \triangleright M_2'$

Fault Context Quantification and Overlap (2)

Give an alternative definition for Fault Tolerance up to n-dynamic faults.

$$\Gamma \triangleright M \preceq_D^n \Gamma \triangleright M$$

Prove its **Soundness** with respect to the previous definition

$$\Gamma_1 \triangleright M_1 \preceq_D^n \Gamma_2 \triangleright M_2$$

$$\text{implies } \forall\, F_D^n(-)$$

$$\Gamma_1 \triangleright M_1 \approx \Gamma_2 \triangleright F_D^n(M_2)$$

Problems we need to address

Hard to prove positive results with our fault tolerance definition because:

1. $\cong$ **quantifies over all** valid **contexts**.
2. Dynamic fault tolerance definition **quantifies over all fault contexts**, amongst which there is **considerable overlap**.
3. There are a number of **confluent** reductions that increase the burden of our analysis.

Confluent τ-transitions

Identify Confluent Moves

(b-eq)

$$\frac{}{\Gamma \triangleright l[\![\text{if } u = u \text{ then } P \text{ else } Q]\!] \xrightarrow{\tau}_{\beta} \Gamma \triangleright l[\![P]\!]}$$

(b-ngo)

$$\frac{}{\langle \mathcal{R}, \mathcal{U} \rangle \triangleright l[\![\text{go } k.P]\!] \xrightarrow{\tau}_{\beta} \langle \mathcal{R}, \mathcal{U} \rangle \triangleright k[\![\mathbf{0}]\!]} \; k \notin \mathcal{R} \cup \mathcal{U}$$

Confluent τ-transitions

Extend Equivalence Relation

((bs-dead))

$$\frac{}{\langle \mathcal{R}, \mathcal{U} \rangle \triangleright l[\![P]\!] \equiv_f \langle \mathcal{R}, \mathcal{U} \rangle \triangleright l[\![Q]\!]} \; l \notin \mathcal{R} \cup \mathcal{U}$$

Proving Confluence

$$\begin{array}{ccc} \Gamma \triangleright N & \overset{\tau}{\underset{\beta}{\longmapsto}} & \Gamma \triangleright M \\ \Big\downarrow{\scriptstyle \mu} & & \\ \Gamma' \triangleright N' & & \end{array}$$

Confluent τ-transitions

Proving Confluence

$$\begin{array}{ccc} \Gamma \triangleright N & \overset{\tau}{\underset{\beta}{\longmapsto}} & \Gamma \triangleright M \\ \mu \big\downarrow & & \big\downarrow \mu \\ \Gamma' \triangleright N' & \overset{\tau}{\underset{\beta}{\longmapsto}} & \Gamma' \triangleright M' \end{array} \qquad \text{or} \qquad \begin{array}{ccc} \Gamma \triangleright N & \overset{\tau}{\underset{\beta}{\longmapsto}} & \Gamma \triangleright M \\ \mu \big\downarrow & & \big\downarrow \mu \\ \Gamma' \triangleright N' & \equiv_f & \Gamma' \triangleright M' \end{array}$$

or $\mu = \tau$ and $\Gamma \triangleright M = \Gamma' \triangleright N'$

Fault Tolerance up to β-moves

$\Gamma_1 \triangleright M_1 \preceq^n_\beta \Gamma_2 \triangleright M_2$ implies

- $\Gamma_1 \triangleright M_1 \xrightarrow{\mu} \Gamma'_1 \triangleright M'_1$ implies $\Gamma_2 \triangleright M_2 \overset{\hat{\mu}}{\Longrightarrow} \Gamma'_2 \triangleright M'_2$ such that $\Gamma'_1 \triangleright M'_1 \; \mathcal{A}_l \circ \preceq^n_\beta \circ \approx_{cnt} \Gamma'_2 \triangleright M'_2$
- $\Gamma_2 \triangleright M_2 \xrightarrow{\mu} \Gamma'_2 \triangleright M'_2$ implies $\Gamma_1 \triangleright M_1 \overset{\hat{\mu}}{\Longrightarrow} \Gamma'_1 \triangleright M'_1$ such that $\Gamma'_2 \triangleright M'_2 \; \mathcal{A}_l \circ \preceq^n_\beta \circ \approx \Gamma'_1 \triangleright M'_1$
- If $n > 0$ then $\Gamma_2 \triangleright M_2 \xrightarrow{\text{fail}} \Gamma'_2 \triangleright M'_2$ implies $\Gamma_1 \triangleright M_1 \Longrightarrow \Gamma'_1 \triangleright M'_1$ such that $\Gamma'_2 \triangleright M'_2 \preceq^{n-1}_\beta \circ \approx \Gamma'_1 \triangleright M'_1$

where $\mathcal{A}_l$ is the relation $\models\!\!\Longrightarrow_\beta \circ \equiv$

$\approx_{cnt}$ is a bisimulation ranging over μ and the new counting action fail.

Confluent τ-transitions

Soundness of $\preceq_\beta^n$

$$\Gamma_1 \triangleright M_1 \preceq_\beta^n \Gamma_2 \triangleright M_2$$

implies

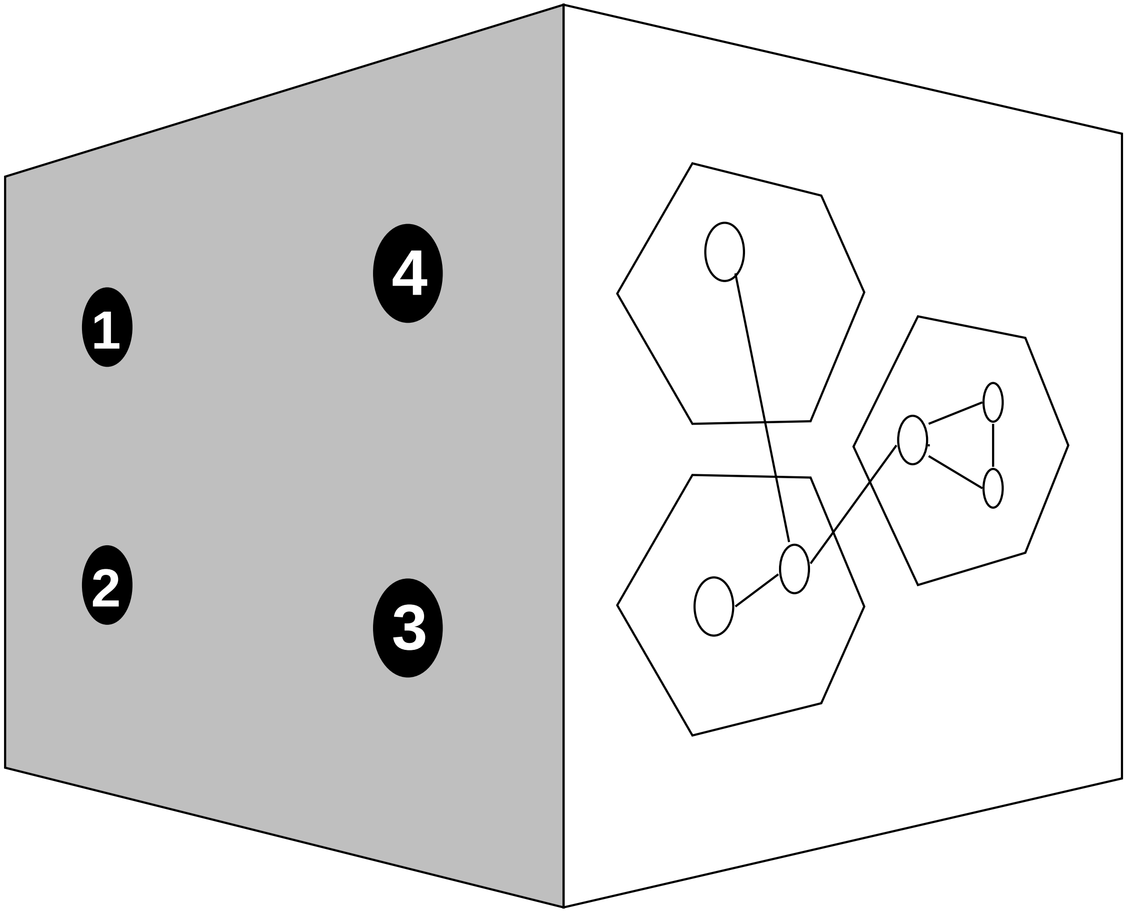
$$\Gamma_1 \triangleright M_1 \preceq_D^n \Gamma_2 \triangleright M_2$$

Talk Summary

- Fault Tolerance Intuitions
- Language
- Formal Definition
- Proof Techniques

Main Result

Induce Faults



Observer View

Main Result

To show that $\Gamma \triangleright M$ is fault tolerant up to n faults we just have to give a witness fault tolerant simulation up to β-moves satifying

$$\Gamma \triangleright M \preceq_{\beta}^{n} \Gamma \triangleright M$$