UNIVERSITÄT
DES
SAARLANDES

Continuity and Robustness of Programs

Seminar: Robustness of Hardware and Software Systems

Prof. Dr.-Ing. Jan Reineke

Markus Schneider

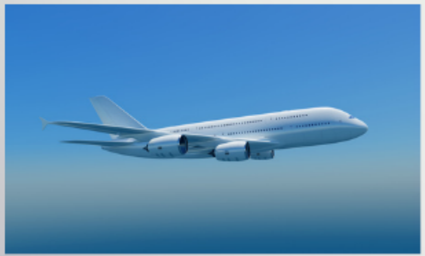
Saarbrücken, February 21, 2014

ROBUSTNESS

caches

Lipschitz continuity

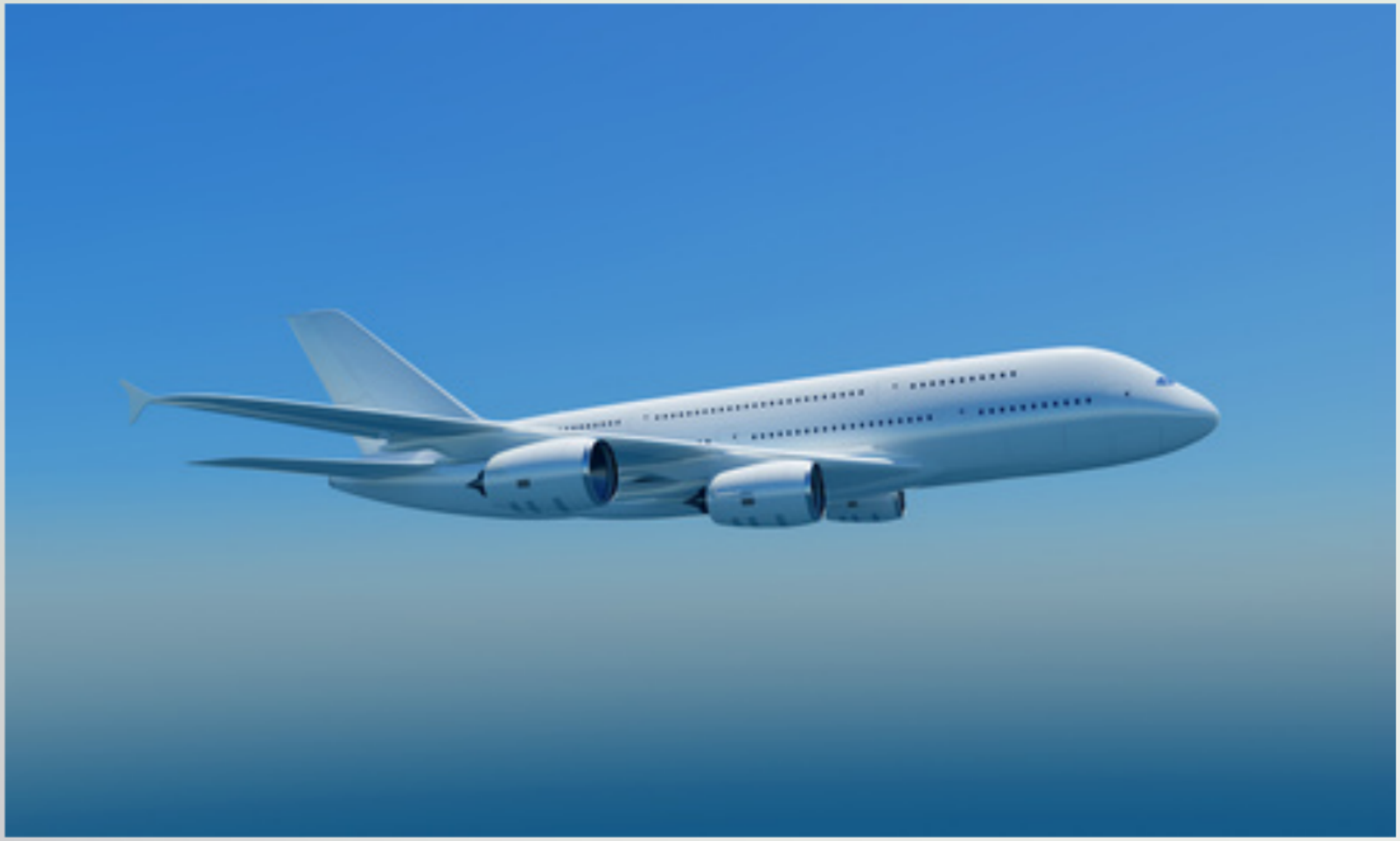
continuity



safety-critical systems

(delta,epsilon)-robustness

networked systems



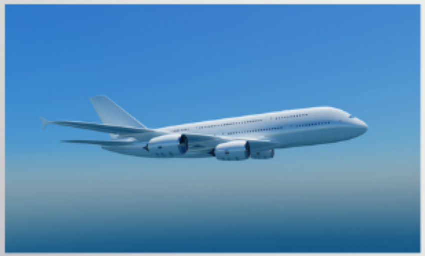
safety-critical systems

ROBUSTNESS

caches

Lipschitz continuity

continuity



safety-critical systems

(delta,epsilon)-robustness

networked systems

continuity

A Necessary Tool: Metrics

- We consider a program as the mathematical function that it implements.
- To be able to talk about continuity we have to **define a metric for each datatype**.
- Examples of metrics:
  - *integer* and *real*, associated with the **Euclidean metric**

$$d(x, y) = |x - y|$$

  - *integer arrays* and *real arrays*, associated with the **maximum norm**

$$d(A_1, A_2) = L_\infty(A_1, A_2) = \max_i(|A_1[i] - A_2[i]|)$$

Closeness of Program States

Continuity analysis of programs requires a definition of a **“distance” between two program states**.

Given two states σ and $\sigma' \in \Sigma(P)$ and any $\epsilon > 0$, we define:

- σ and σ' are ϵ-**close** with respect to variable x_i and write

$$\sigma \approx_{\epsilon,i} \sigma' \;:\Leftrightarrow\; d(\sigma(i), \sigma'(i)) < \epsilon$$

- σ' is an ϵ-**perturbation of** σ with respect to variable x_i and write

$$\sigma \equiv_{\epsilon,i} \sigma' \;:\Leftrightarrow\; \sigma \approx_{\epsilon,i} \sigma' \wedge \forall j \neq i : \sigma(j) = \sigma'(j)$$

Continuity of a Program

Well-known ϵ-δ-Definition of Continuous Functions:
A function $f : D \to \mathbb{R}$ is continuous at a point $x \in D$, if

$$\forall \epsilon > 0 \; \exists \delta > 0 \; \forall y \in D : \; |x - y| < \delta \Rightarrow |f(x) - f(y)| < \epsilon$$

Continuity of a Program:
A program P is **continuous** at a state σ with respect to an input variable x_i and an output variable x_j, if

$$\forall \epsilon > 0 \; \exists \delta > 0 \; \forall \sigma' \in \Sigma(P) : \; \sigma \equiv_{\delta,i} \sigma' \Rightarrow [\![P]\!](\sigma) \approx_{\epsilon,j} [\![P]\!](\sigma')$$

Verifying Continuity

Breaking down a program into its syntactic substructures we get a set of **inference rules** of the style

$$\frac{P \text{ is SKIP or } x := e}{b \vdash \mathrm{Cont}(P, \mathrm{In}, \mathrm{Out})}$$

to derive **continuity judgements**.

Disallowing divisions the critical statements are **conditional branches**.

- The branches have to be *output-equivalent* at the decision boundary of the branch.

if $x > 2$ **then**
    $y := \frac{1}{2} \cdot x$
else
    $y := -5x + 11$
end if

Lipschitz continuity

Lipschitz Continuity of a Program

Definition of Lipschitz continuous Functions:
A function $f : D \rightarrow \mathbb{R}$ is Lipschitz continuous, if there is a constant K so that any $\pm\epsilon$-change to x can change $f(x)$ at most by $\pm K \cdot \epsilon$.

Lipschitz Continuity of a Program:
Let $K : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ be a function that takes the size of variable x_i as its input. A program P is K-**Lipschitz** with respect to an input variable x_i and an output variable x_j, if $\forall \sigma, \sigma' \in \Sigma(P)$ and $\forall \epsilon > 0$

$$\sigma \equiv_{\epsilon,i} \sigma' \wedge \left(||\sigma(i)|| = ||\sigma'(i)||\right) \Rightarrow [\![P]\!](\sigma) \approx_{K\cdot\epsilon,j} [\![P]\!](\sigma')$$

where K only depends on the size of $\sigma(i)$. The size of a variable v is defined as

- $||v|| := 1$, if v is an integer or a real,
- $||v|| := N$, if v is an array of size N.

Robustness of Programs

For Lipschitz continuous programs we can state:

- The output changes proportionally to any change on the inputs.
- The upper bound $K \cdot \epsilon$ on the output changes does not depend on the values of the input variables.

$\longrightarrow$ The program behaves predictably on uncertain inputs.

A program is called robust, if it is K-Lipschitz for some Lipschitz constant K.

Verifying Lipschitz Continuity

The sequence of assignment or SKIP-statements that P executes on some input is called a **control flow path** of P.

Let x_j be the input and x_i be the output variable of our program.

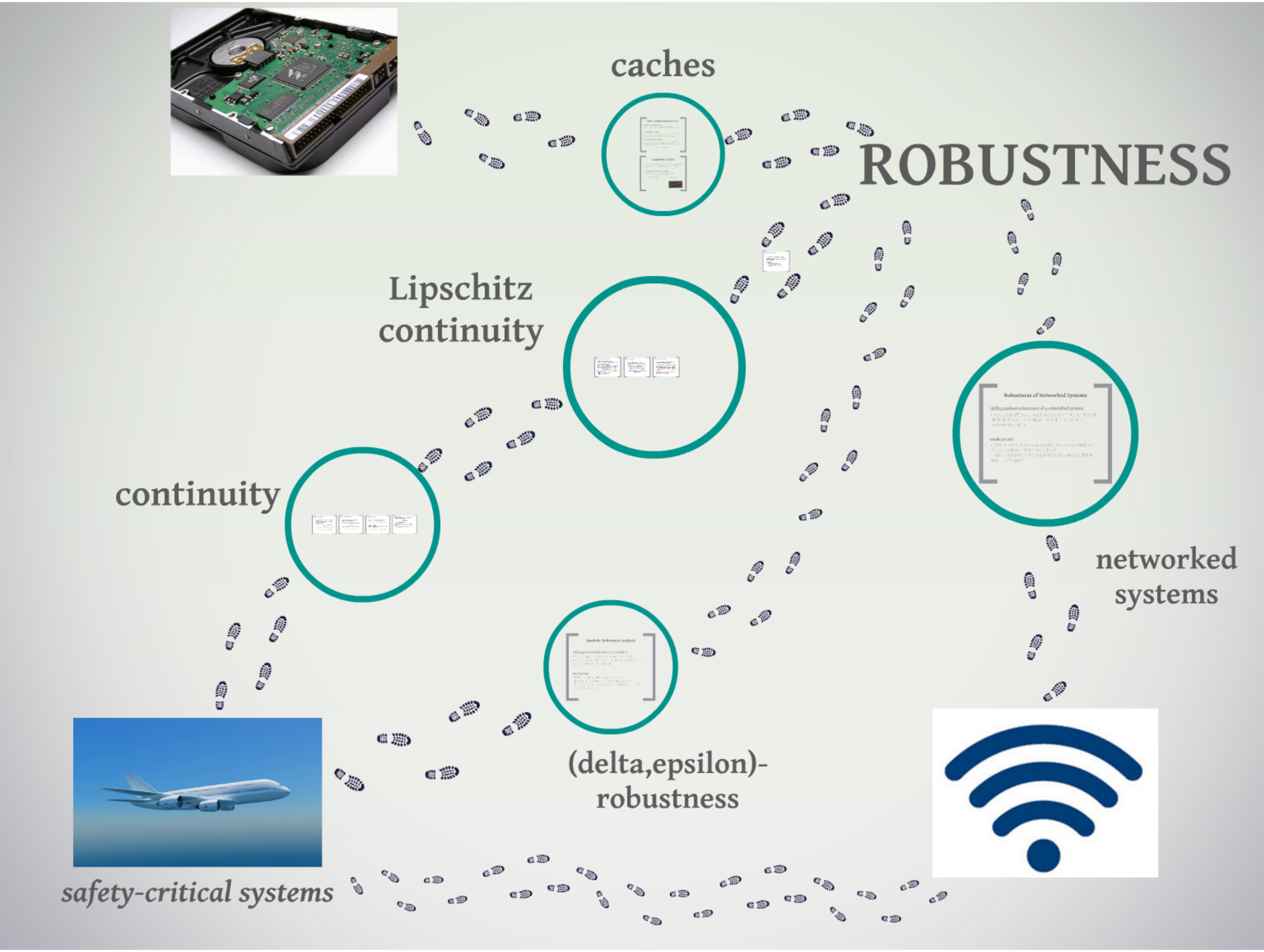
Lipschitz continuity of a program is proven by establishing that

1. P is continuous in all states w.r.t. input x_j and output x_i.
2. Each control flow path of P is K-Lipschitz w.r.t. input x_j and output x_i.

What remains to show is step 2. In doing so, we derive a set of **Lipschitz matrices** for the given program.

Conclusion for this Approach

- We asked for a theory about **robustness** of programs to uncertainty.
- **Lipschitz continuity** is an adequate answer to this question. It is a strong property.
- **Weak points:**
  - Is it satisfactory to live without divisions?
  - The degree of automation remains unclear.
  - No reasonable handling of discrete input data.
  - Not applicable to reactive or concurrent systems.


ROBUSTNESS
caches
Lipschitz continuity
continuity
(delta,epsilon)-robustness
networked systems
safety-critical systems
Robustness of Networked Systems

Symbolic Robustness Analysis

(delta,epsilon)-robustness w.r.t. variable x:

The outputs differ at most by epsilon if the input variable x is perturbed at most by delta (and all other variables remain unchanged).

weak points:

- How to choose the parameter delta?
- No direct adaption to closed loop systems.
- Floating-point numbers and non-linear arithmetic cannot be handled.

(delta,epsilon)-robustness

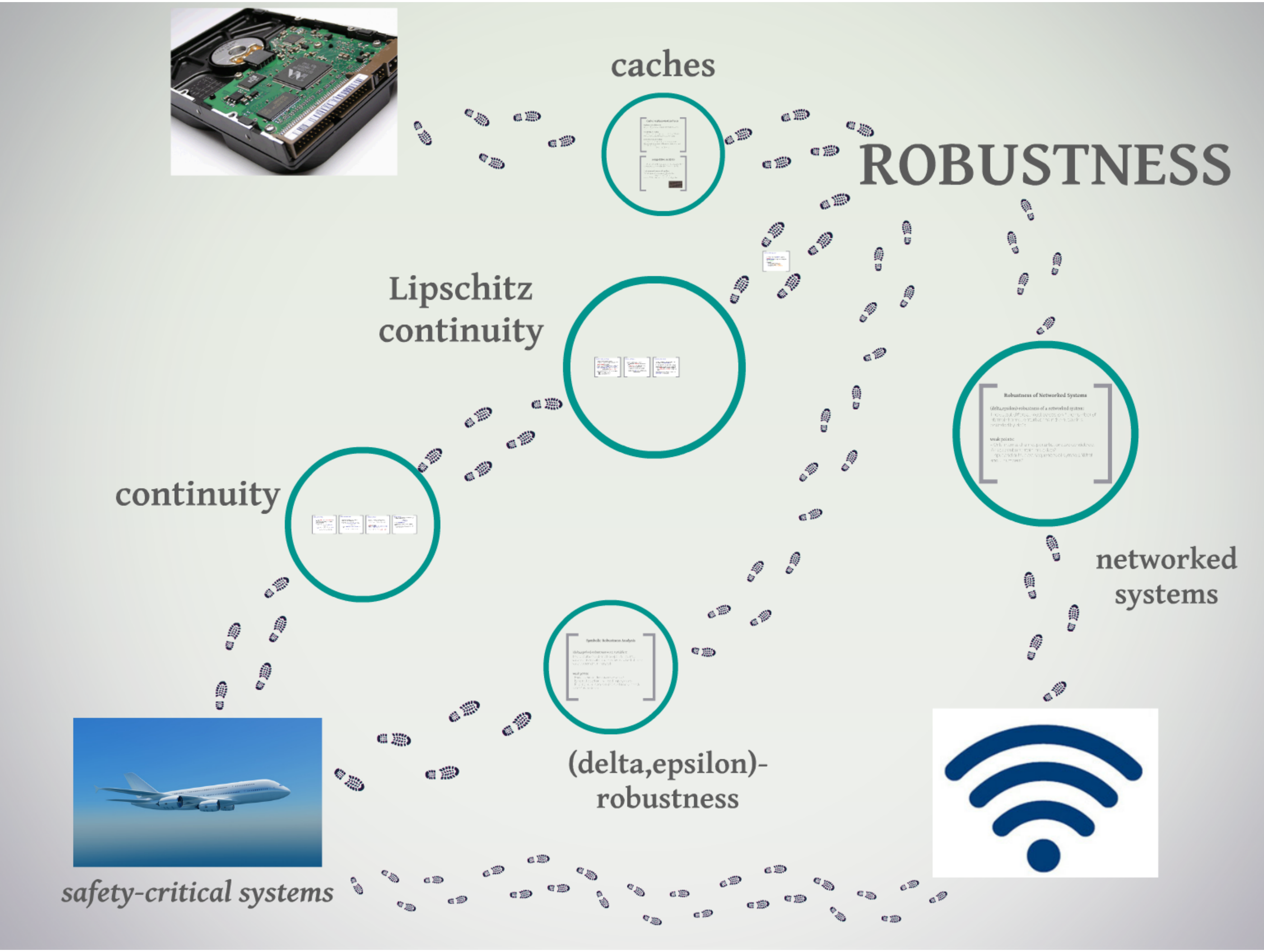
Symbolic Robustness Analysis

(delta,epsilon)-robustness w.r.t. variable x:

The outputs differ at most by epsilon if the input variable x is perturbed at most by delta (and all other variables remain unchanged).

weak points:

- How to choose the parameter delta?
- No direct adaption to closed loop systems.
- Floating-point numbers and non-linear arithmetic cannot be handled.



ROBUSTNESS
caches
Lipschitz continuity
continuity
networked systems
(delta,epsilon)-robustness
safety-critical systems

Robustness of Networked Systems

(delta,epsilon)-robustness of a networked system:

The output differs at most by epsilon if the number of internal channel perturbations in the network is bounded by delta.

weak points:

- Only internal channel perturbations are considered. What about uncertain input data?
- Input and output are sequences of symbols. What about numbers?

networked systems

Robustness of Networked Systems

(delta,epsilon)-robustness of a networked system:

The output differs at most by epsilon if the number of internal channel perturbations in the network is bounded by delta.

weak points:

- Only internal channel perturbations are considered. What about uncertain input data?
- Input and output are sequences of symbols. What about numbers?

ROBUSTNESS

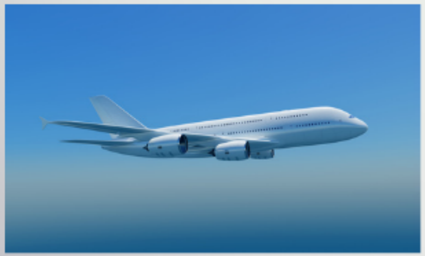
caches

Lipschitz continuity

continuity

(delta,epsilon)-robustness

networked systems



safety-critical systems

caches

Cache replacement policies

Replacement policies ask:
Which memory block should we replace upon a cache miss?

Sensitivity of a policy:
To what extent does the execution history influence the number of cache hits and cache misses?

(r,c)-robustness of a policy:
Let $A(s)$ be the number of cache misses on input sequence s. Whenever d(s,s') <= delta for a fixed delta, it holds that

$A(s) <= r * A(s') + c$

competitive analysis

Analyze the performance of an online algorithm compared to the optimal offline algorithm.

(r,c)-competitiveness of a policy:
For all input sequences s it holds that:

$A(s) <= r * OPT(s) + c$

where OPT is the optimal offline algorithm

Cache replacement policies

Replacement policies ask:
Which memory block should we replace upon a cache miss?

Sensitivity of a policy:
To what extent does the execution history influence the number of cache hits and cache misses?

(r,c)-robustness of a policy:

Let A(s) be the number of cache misses on input sequence s. Whenever dist(s1,s2) <= delta for a fixed delta, it holds that

$$A(s1) <= r * A(s2) + c$$

competitive analysis

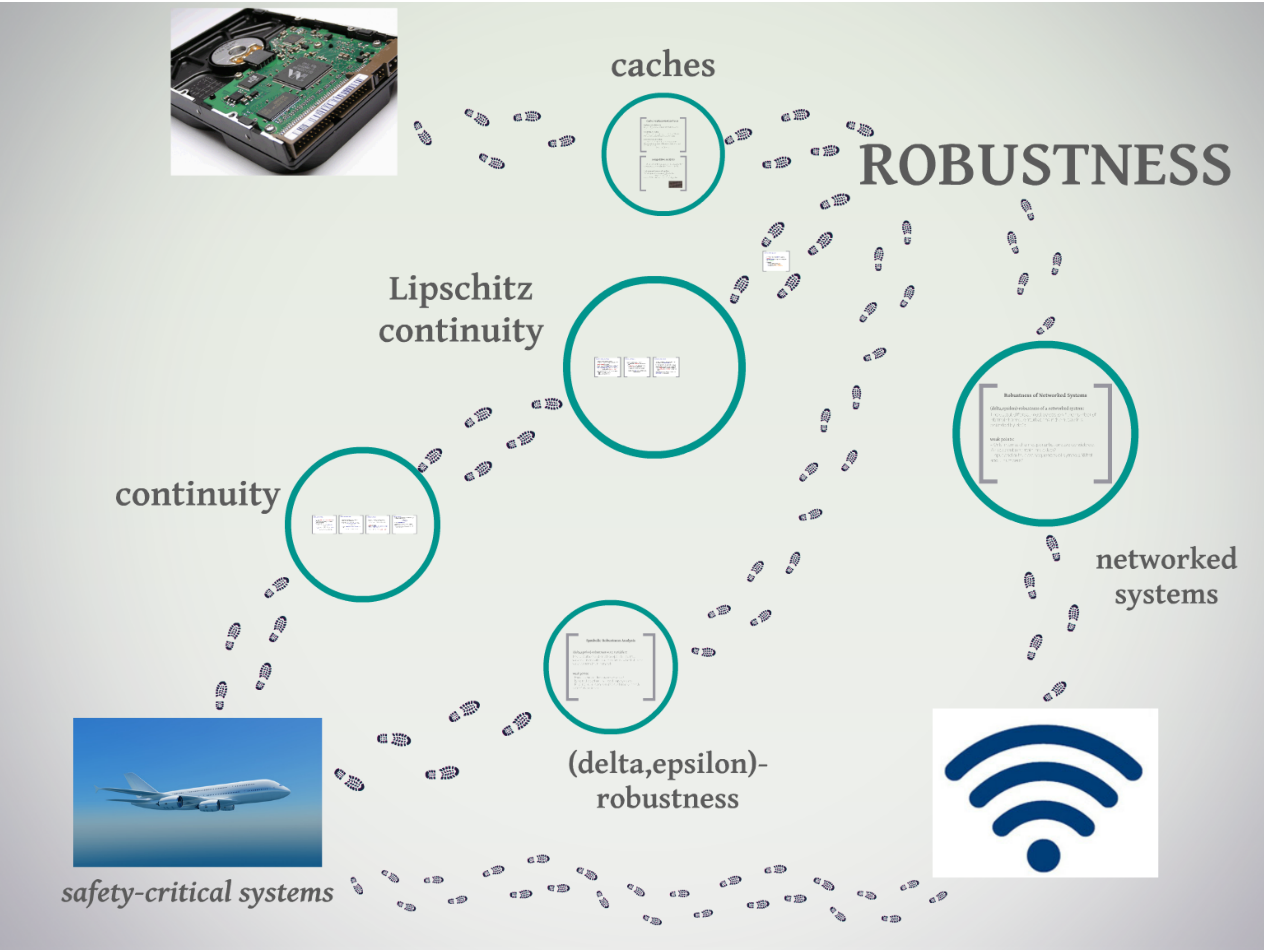
Analyze the performance of an online algorithm compared to the optimal offline algorithm.

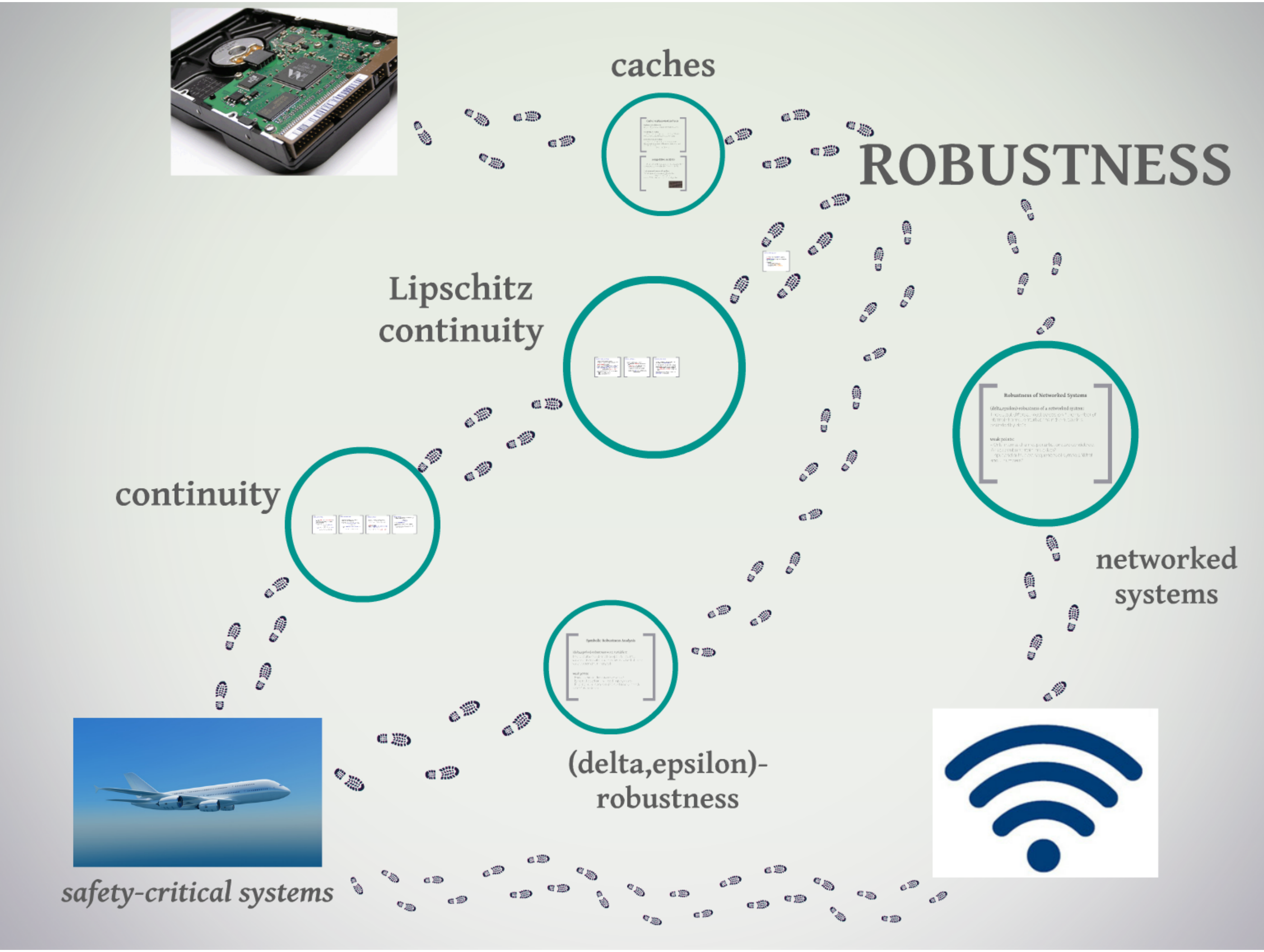
(r,c)-competitiveness of a policy:

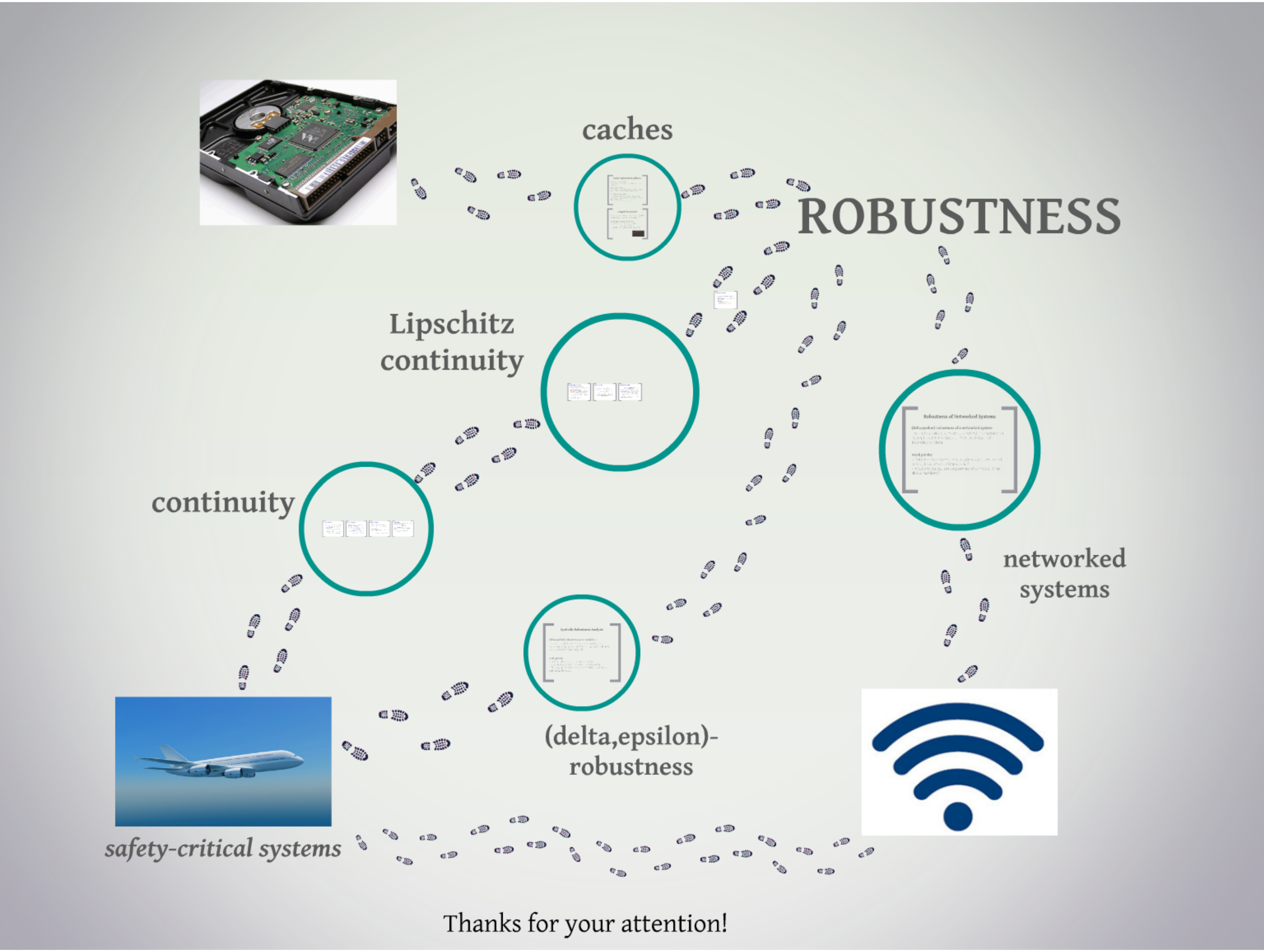
For all input sequences s it holds that

$$A(s) <= r * OPT(s) + c$$

where OPT is the optimal offline algorithm.


caches
ROBUSTNESS
Lipschitz continuity
continuity
networked systems
(delta,epsilon)-robustness
safety-critical systems


ROBUSTNESS
caches
Lipschitz continuity
continuity
networked systems
(delta,epsilon)-robustness
safety-critical systems


caches
ROBUSTNESS
Lipschitz continuity
continuity
networked systems
(delta,epsilon)-robustness
safety-critical systems
Thanks for your attention!

Literature

- Chaudhuri, S.,Gulwani, S. & Lublinerman, R. (2010). *Continuity Analysis of Programs.* POPL, 57-70.
- Chaudhuri, S., Gulwani, S., Navidpour, S. & Lublinerman, R. (2011). *Proving Programs Robust.* FSE, 102-112.
- Chaudhuri, S.,Gulwani, S. & Lublinerman, R. (2012). *Continuity and Robustness of Programs.* CACM, 107-115.
- Majumdar, R. and Saha, I. (2009). *Symbolic Robustness Analysis.* RTSS.
- Reineke, J. and Grund, D. (2013). *Sensitivity of cache replacement policies.* ACM Transactions on Embedded Computing Systems (TECS).
- Samanta, R., Deshmukh, J., Chaudhuri, S. (2013). *Robustness Analysis of Networked Systems.* VMCAI.

Image Sources

http://www.topwerte.info/Flugzeug.jpg

http://media2.giga.de/2013/11/WLAN.jpg

http://festplatte-intern.de/wp-content/uploads/festplatte-intern.gif

http://www.cicero.de/sites/default/files/field/image/boerse_0.jpg