Robustness Analysis of Networked Systems

Pascal Berrang

Are they reliable?

- Verification:
  System is **correct** or **incorrect**.

- Robustness:
  considers *uncertainty.*

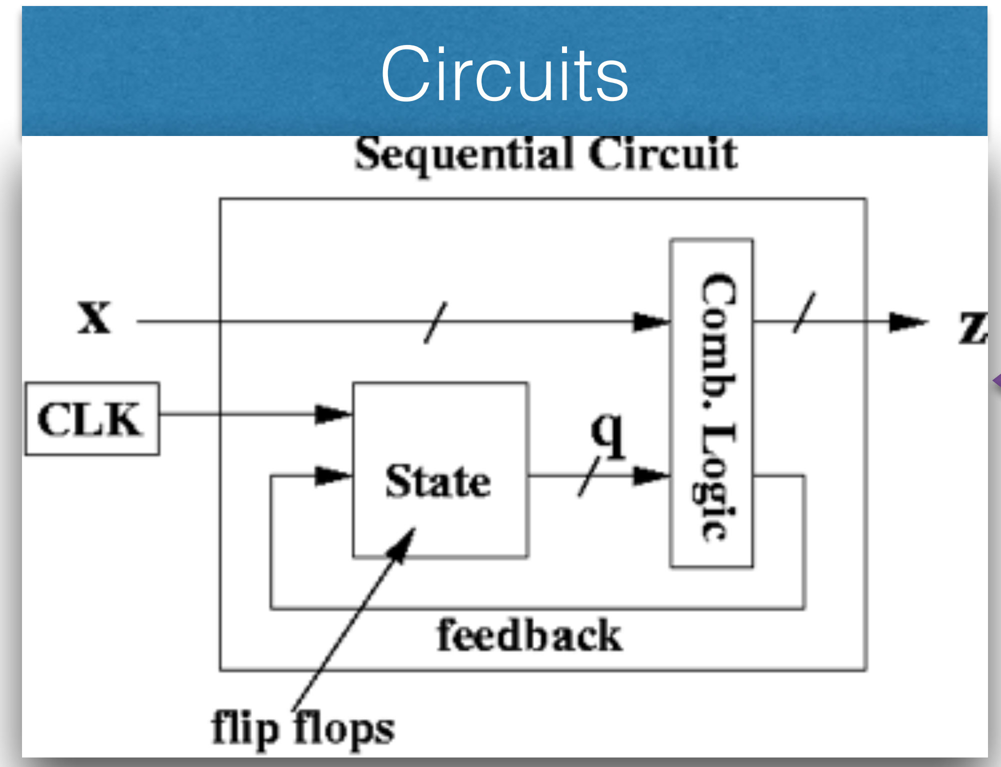
Sensors

Network Channels

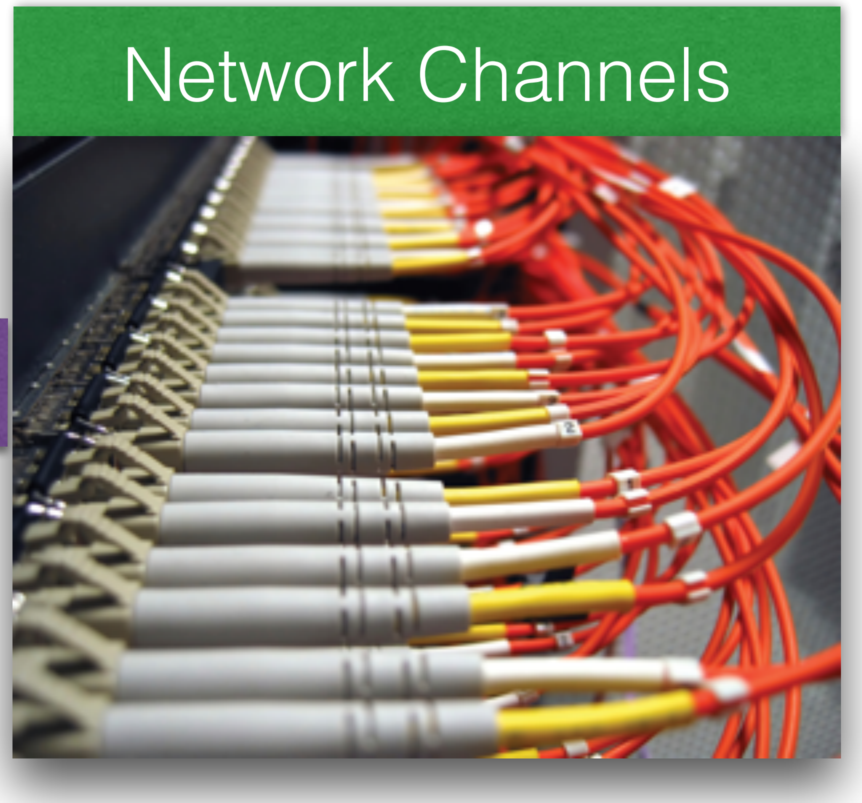
Inputs

„Small perturbations to the environment or parameters do not change the observable behavior substantially.“

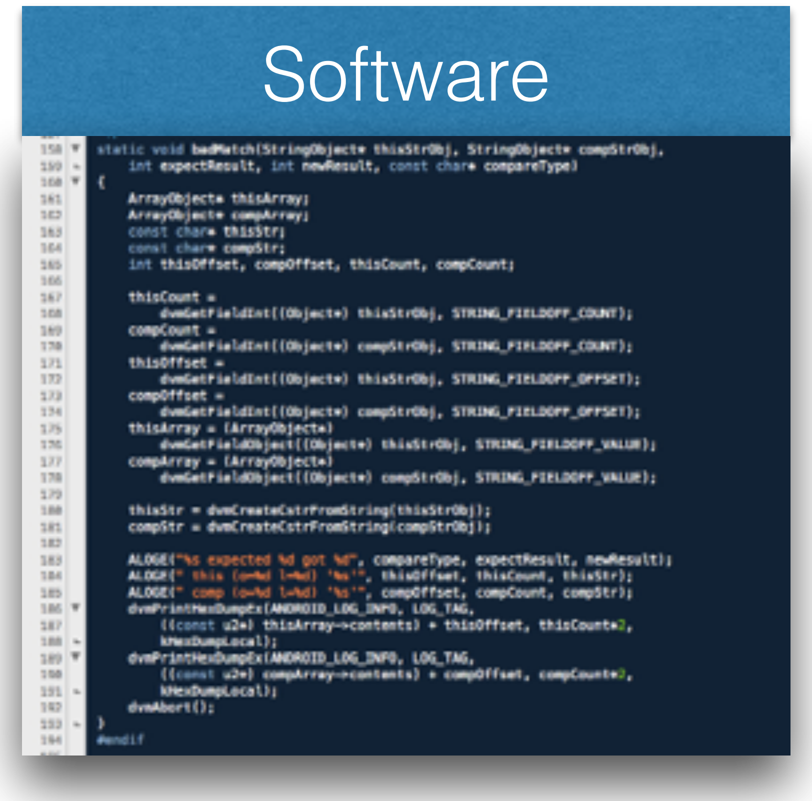
Circuits



Network Channels



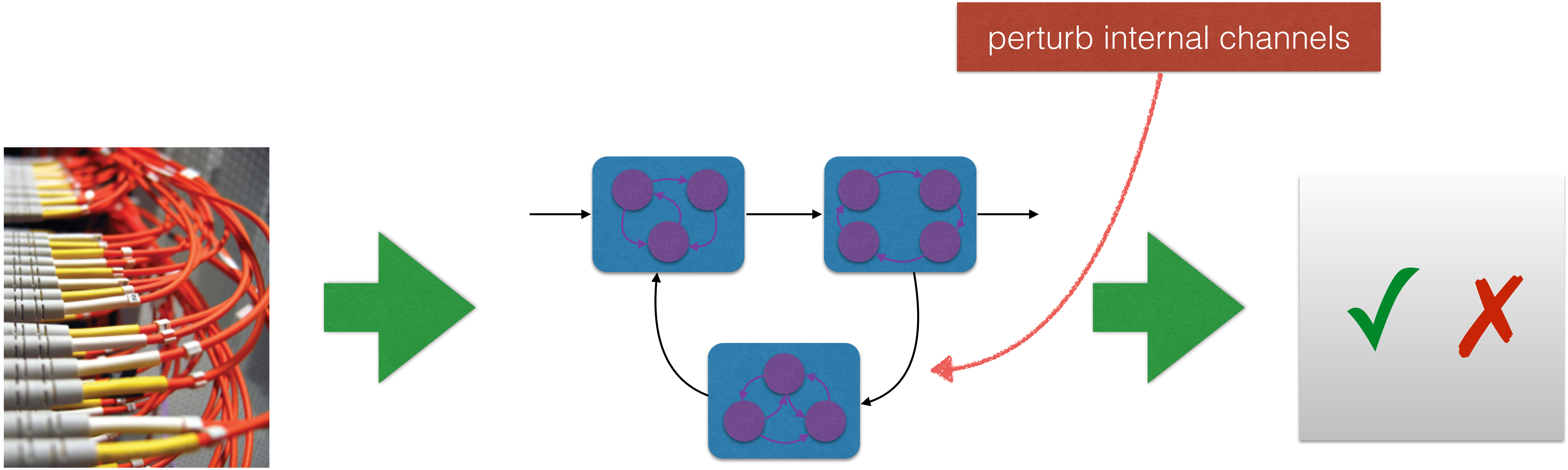
Software



```
static void badMatch(StringObject* thisStrObj, StringObject* compStrObj,
    int expectResult, int newResult, const char* compareType)
{
    ArrayObject* thisArray;
    ArrayObject* compArray;
    const char* thisStr;
    const char* compStr;
    int thisOffset, compOffset, thisCount, compCount;

    thisCount =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIELDOFF_COUNT);
    compCount =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIELDOFF_COUNT);
    thisOffset =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIELDOFF_OFFSET);
    compOffset =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIELDOFF_OFFSET);
    thisArray = (ArrayObject*)
        dvmGetFieldObject((Object*) thisStrObj, STRING_FIELDOFF_VALUE);
    compArray = (ArrayObject*)
        dvmGetFieldObject((Object*) compStrObj, STRING_FIELDOFF_VALUE);

    thisStr = dvmCreateCstrFromString(thisStrObj);
    compStr = dvmCreateCstrFromString(compStrObj);

    ALOGE("%s expected %d got %d", compareType, expectResult, newResult);
    ALOGE(" this (o=%d l=%d) '%s'", thisOffset, thisCount, thisStr);
    ALOGE(" comp (o=%d l=%d) '%s'", compOffset, compCount, compStr);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) thisArray->contents) + thisOffset, thisCount*2,
        kHexDumpLocal);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) compArray->contents) + compOffset, compCount*2,
        kHexDumpLocal);
    dvmAbort();
}
#endif
```

Caches

„Small <u>perturbations</u> to the environment or parameters do <u>not change</u> the <u>observable behavior</u> substantially.“



perturb internal channels

Networked System

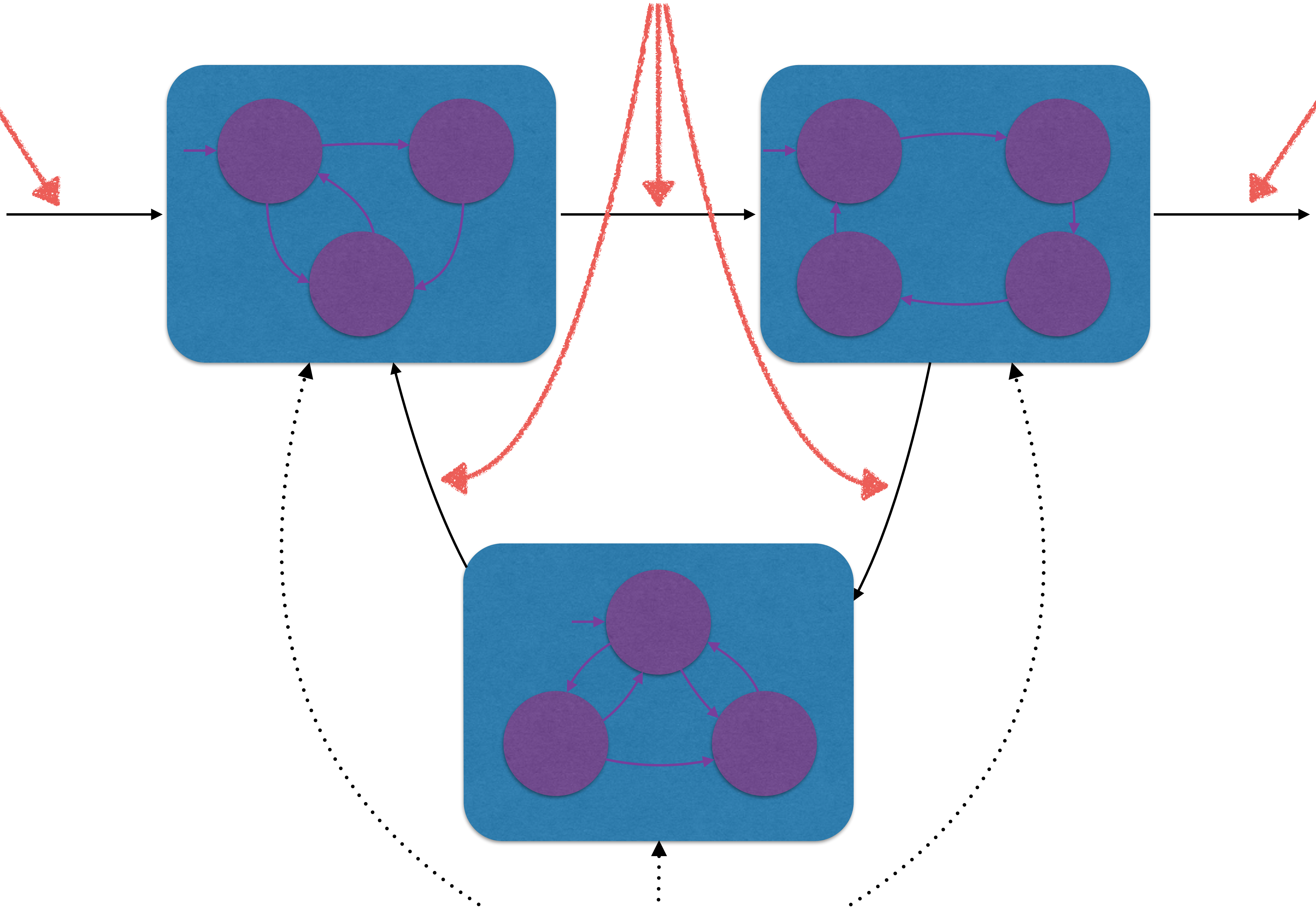
Model

Check Robustness

Input Channels

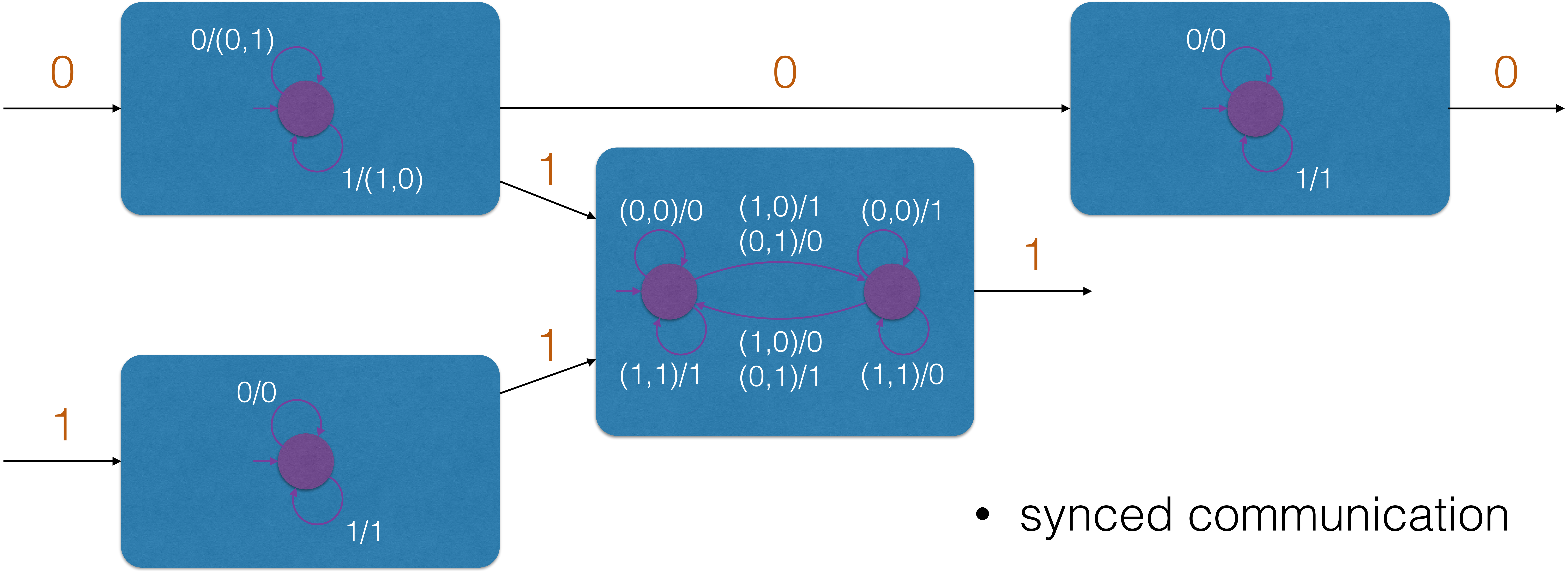
Internal Channels

Output Channels

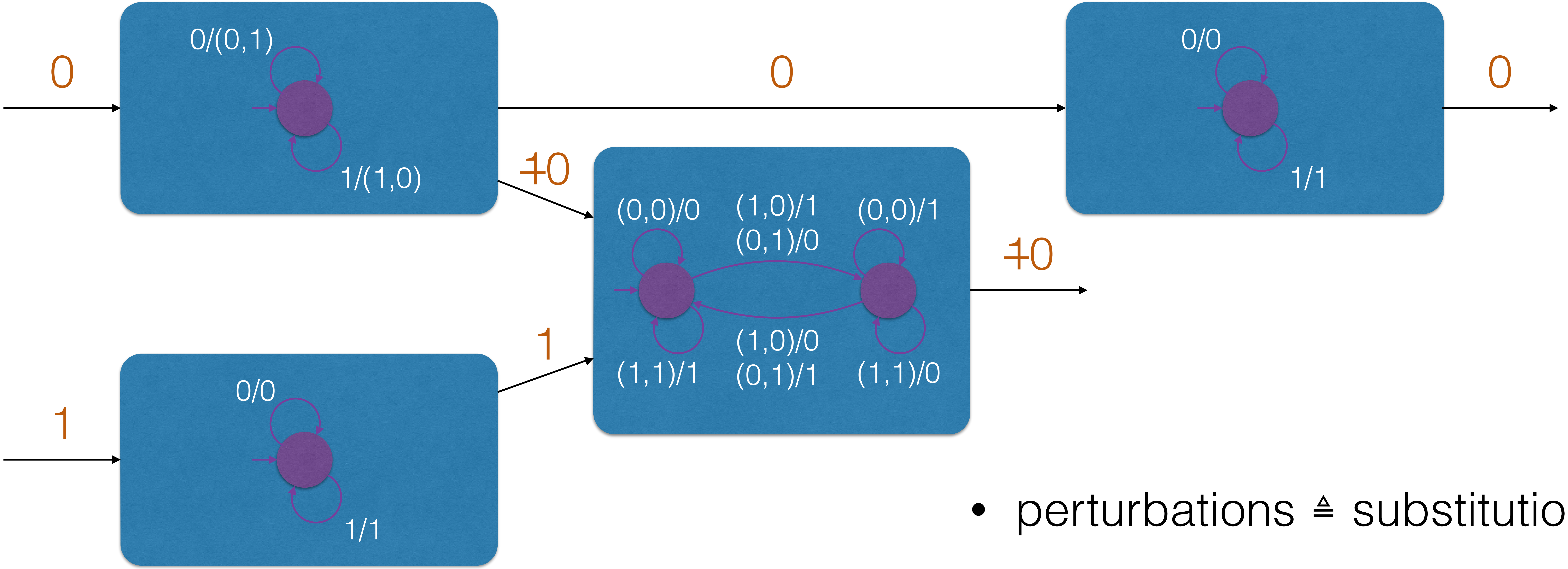


Processes

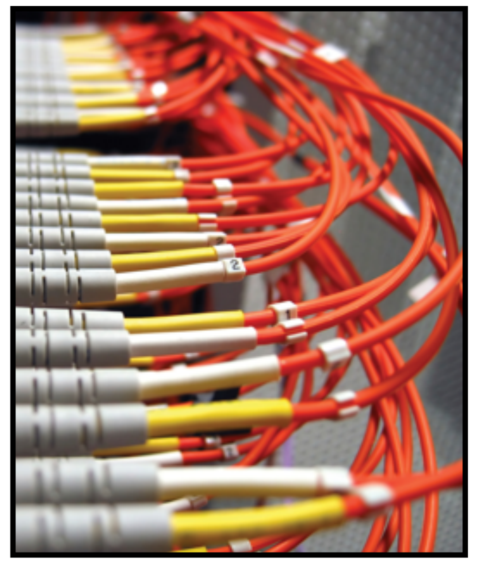
Mealy Machines



- synced communication
- instant message delivery

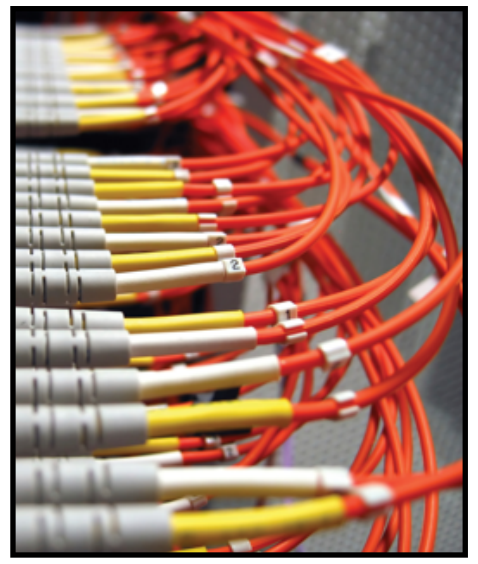


- perturbations ≜ substitutions
- deletions ≜ extra symbol



(δ,ε)-robustness

- **if** perturbations ≤ δ **then** error in output channels ≤ ε
- error measure: *d(normal output, perturbed output)*
  - Levenshtein distance
  - L_1 distance



emptiness

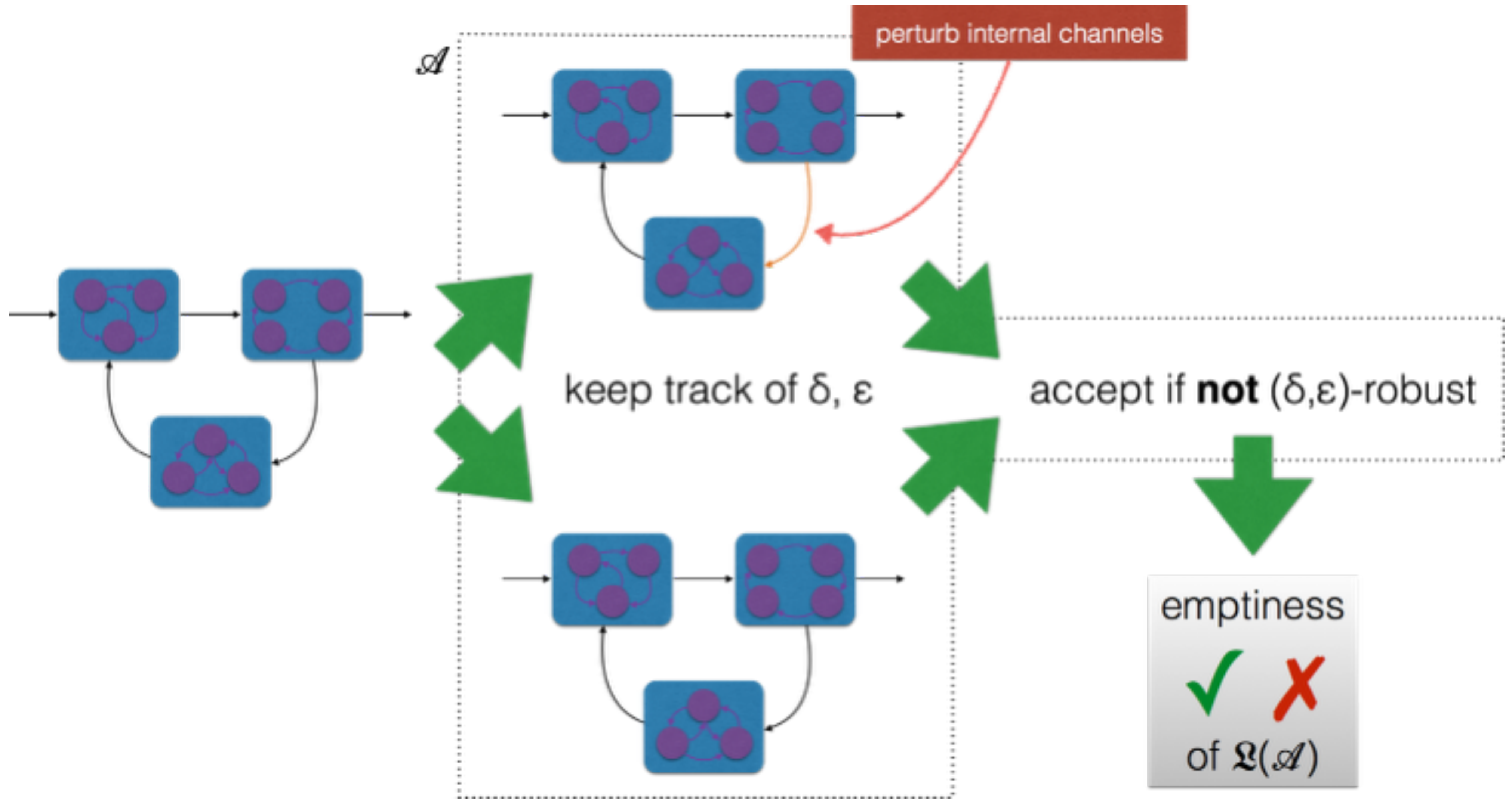
✓ ✗

of $\mathfrak{L}(\mathscr{A})$

- $\mathscr{A}^{\delta,\varepsilon}$ certifies *non-robustness*

- Input: string *s*
  - simulate unperturbed execution
  - simulate perturbed execution
  - keep track of the perturbations
  - keep track of the distance of the outputs

➡ 1-reversal-bounded counter machine



$\mathscr{A}$

perturb internal channels

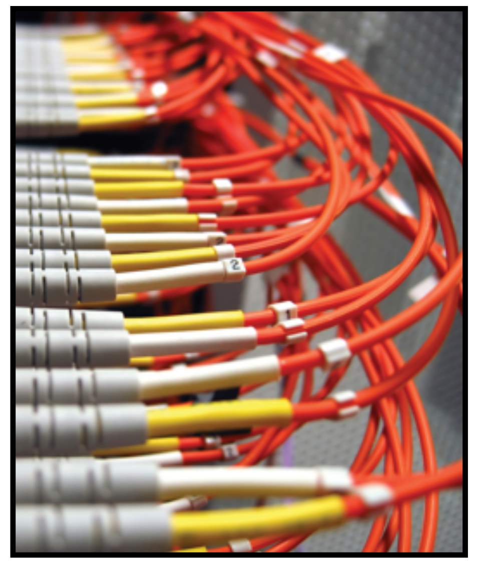
keep track of δ, ε

accept if **not** (δ,ε)-robust

emptiness

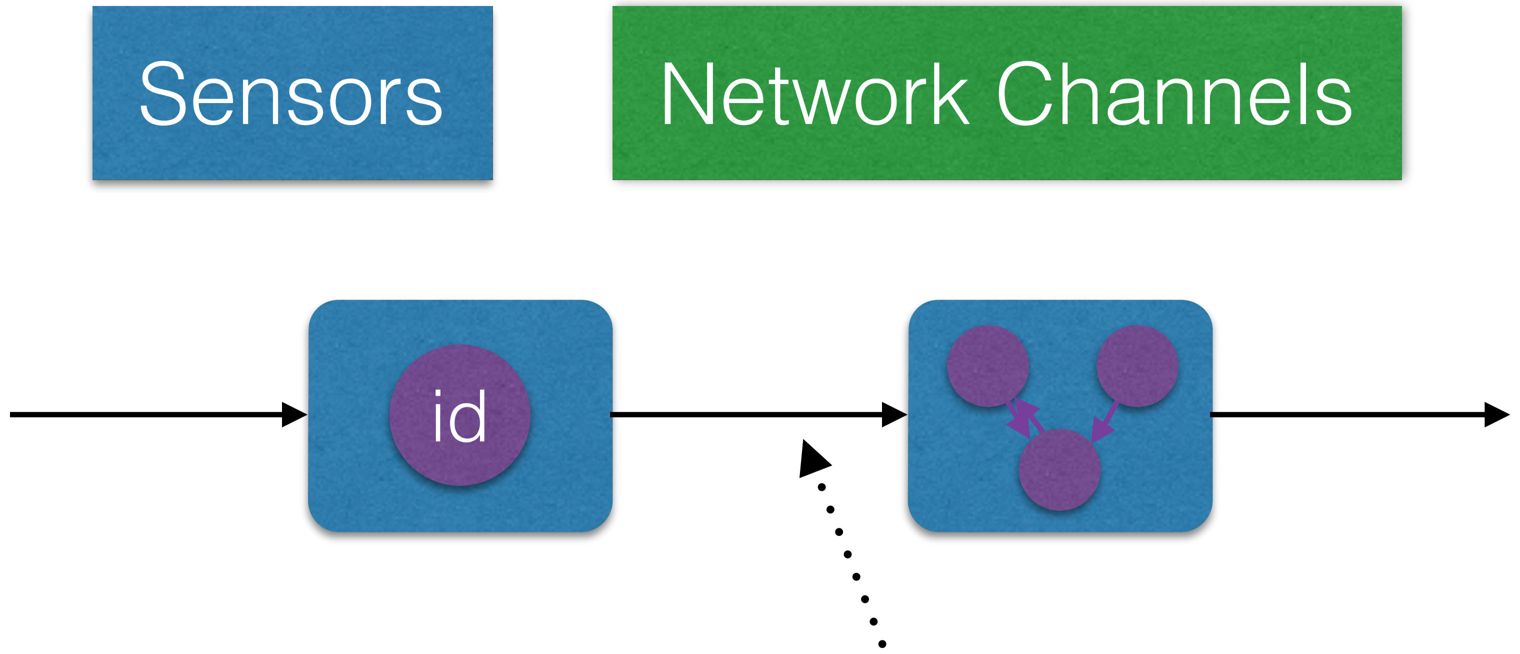
✓ ✗

of $\mathfrak{L}(\mathscr{A})$



Limitations

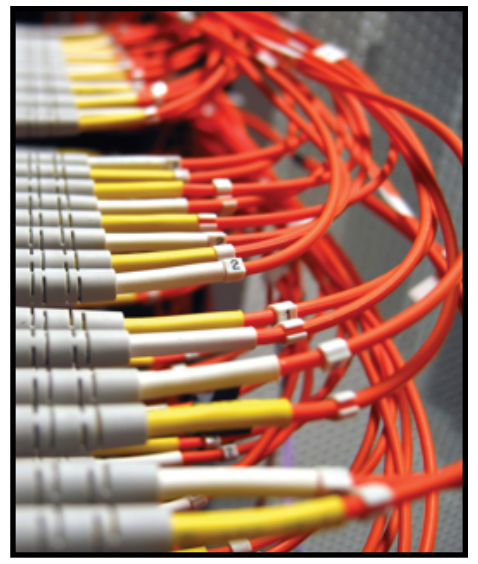
- digital signals:
    - *d(house, mouse)* = 1
    - *d(10, 9)* = ?
- uncertainty:



Sensors

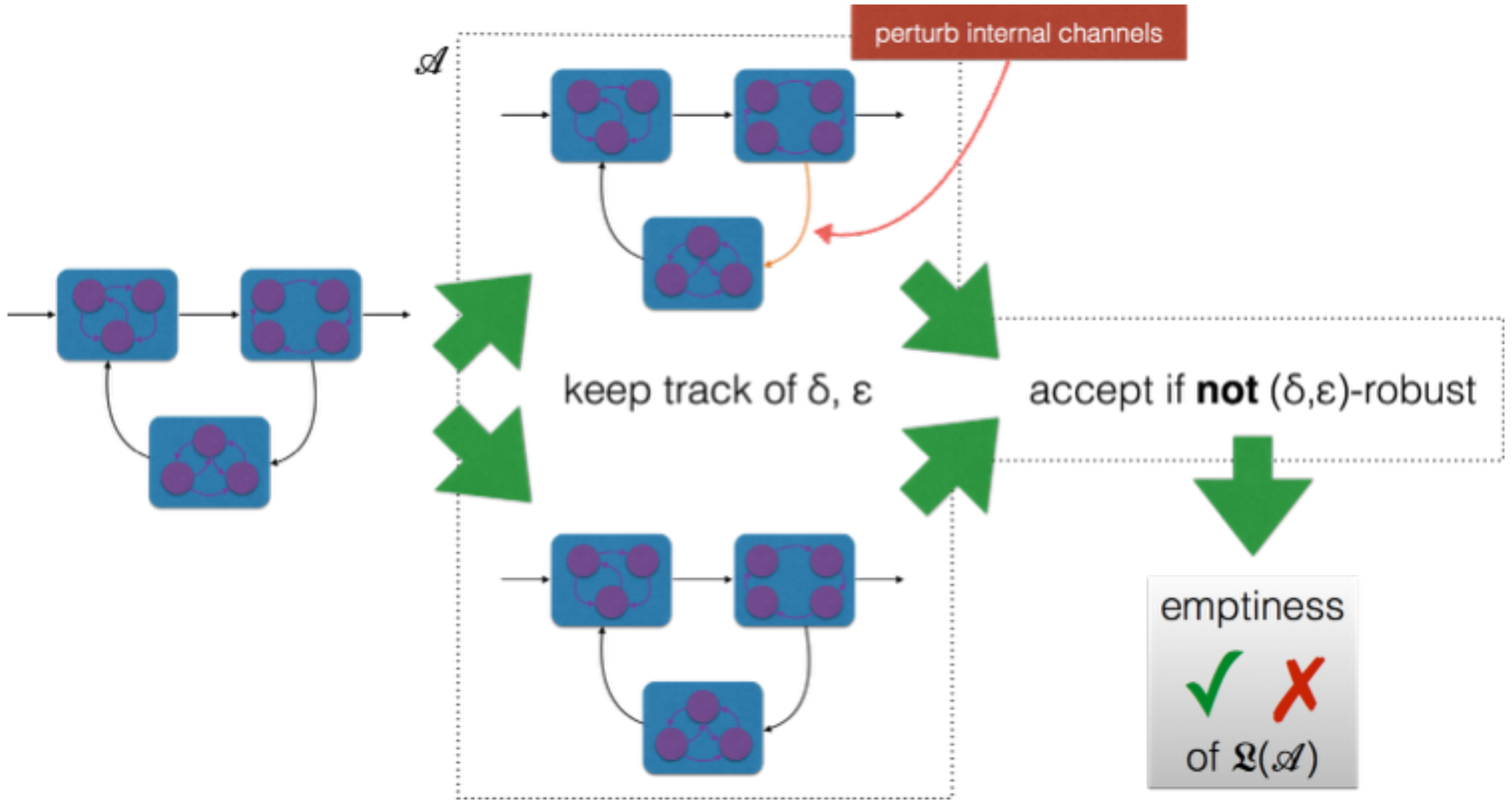
Network Channels

behaves like an input channel



First Conclusion

- Networked systems often safety critical.
- Robustness is crucial in networked systems!
- Easy model for error-prone networks.
- Many distance metrics possible.
- Possible extension: generalize error model.



perturb internal channels

$\mathscr{A}$

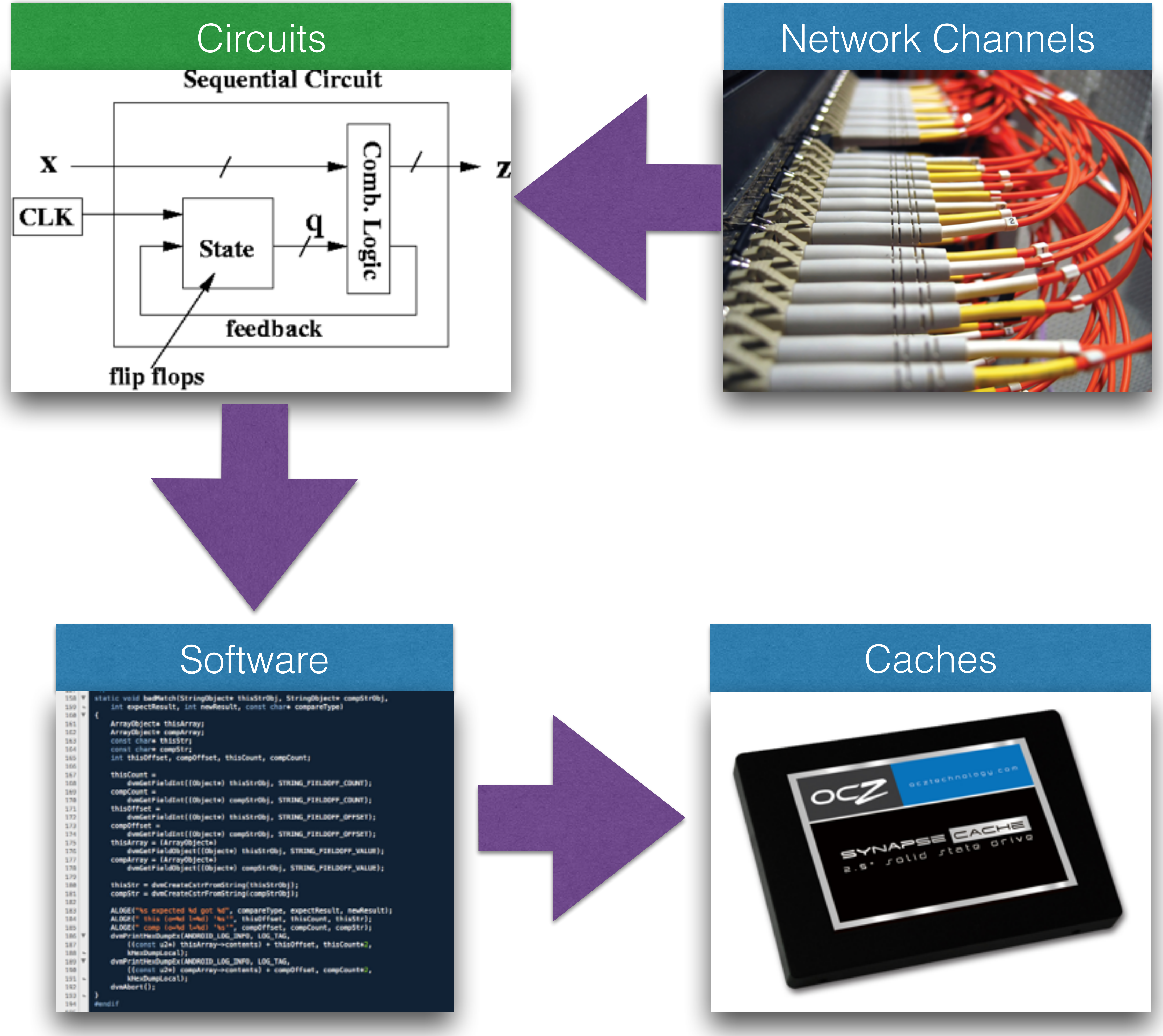
keep track of δ, ε

accept if **not** (δ,ε)-robust

emptiness

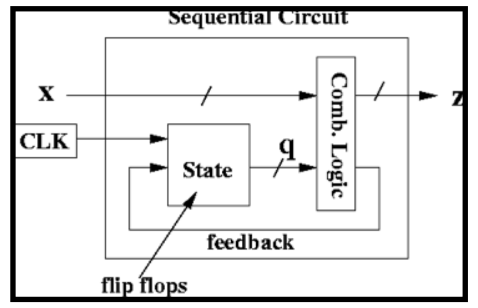
✓ ✗

of $\mathfrak{L}(\mathscr{A})$



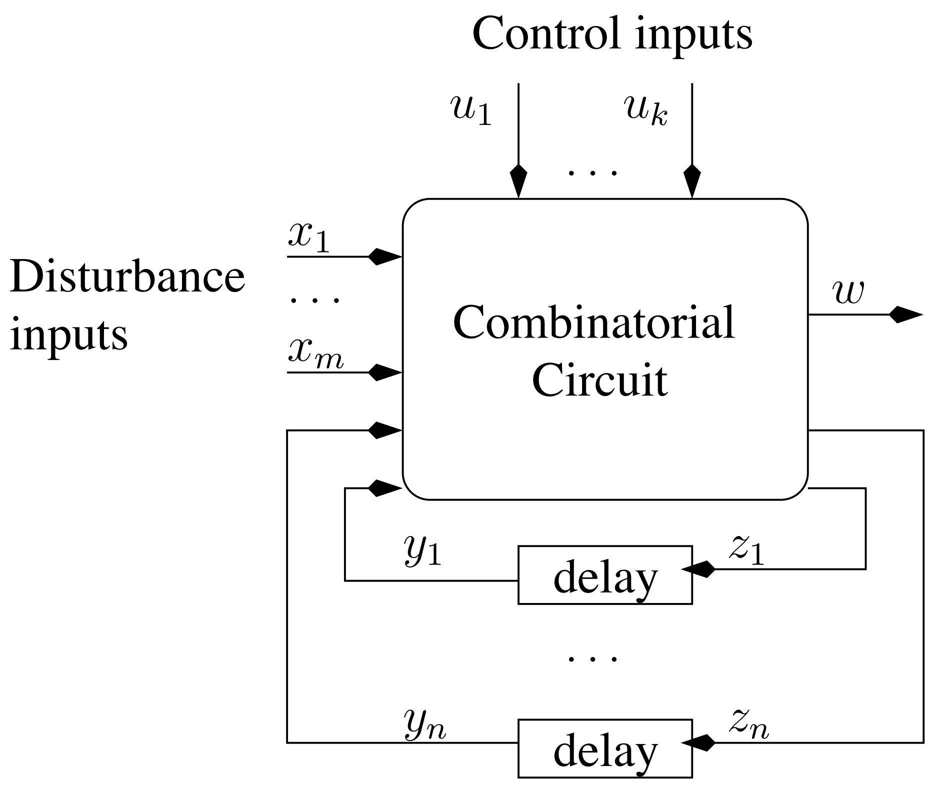
```
static void badMatch(StringObject* thisStrObj, StringObject* compStrObj,
    int expectResult, int newResult, const char* compareType)
{
    ArrayObject* thisArray;
    ArrayObject* compArray;
    const char* thisStr;
    const char* compStr;
    int thisOffset, compOffset, thisCount, compCount;

    thisCount =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIELDOFF_COUNT);
    compCount =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIELDOFF_COUNT);
    thisOffset =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIELDOFF_OFFSET);
    compOffset =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIELDOFF_OFFSET);
    thisArray = (ArrayObject*)
        dvmGetFieldObject((Object*) thisStrObj, STRING_FIELDOFF_VALUE);
    compArray = (ArrayObject*)
        dvmGetFieldObject((Object*) compStrObj, STRING_FIELDOFF_VALUE);

    thisStr = dvmCreateCstrFromString(thisStrObj);
    compStr = dvmCreateCstrFromString(compStrObj);

    ALOGE("%s expected %d got %d", compareType, expectResult, newResult);
    ALOGE(" this (o=%d l=%d) '%s'", thisOffset, thisCount, thisStr);
    ALOGE(" comp (o=%d l=%d) '%s'", compOffset, compCount, compStr);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) thisArray->contents) + thisOffset, thisCount*2,
        kHexDumpLocal);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) compArray->contents) + compOffset, compCount*2,
        kHexDumpLocal);
    dvmAbort();
}
#endif
```

„Small <u>perturbations</u> to the environment or parameters do <u>not change</u> the <u>observable behavior</u> substantially.“

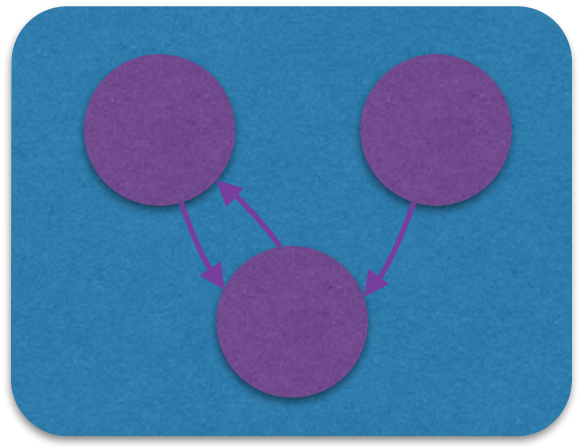


Σ_D-robustness

if last mismatch in disturbance inputs < k
then last mismatch in output < k+b



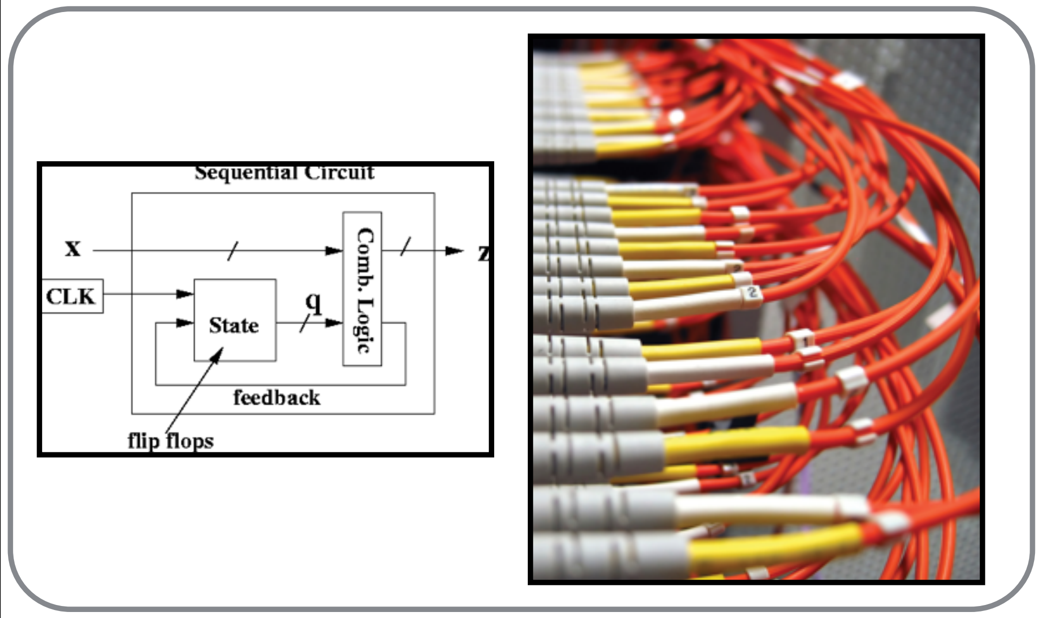
Equivalent Mealy Machine



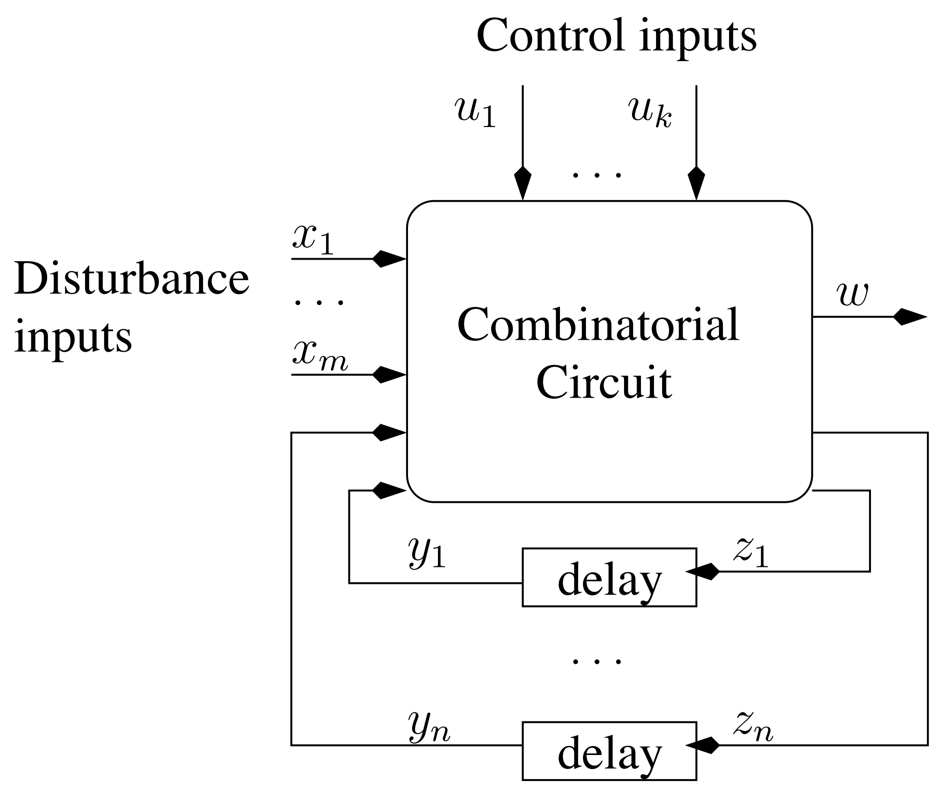
two different disturbance inputs
reach a reset state
after next ≤ b identical inputs

Limitations:

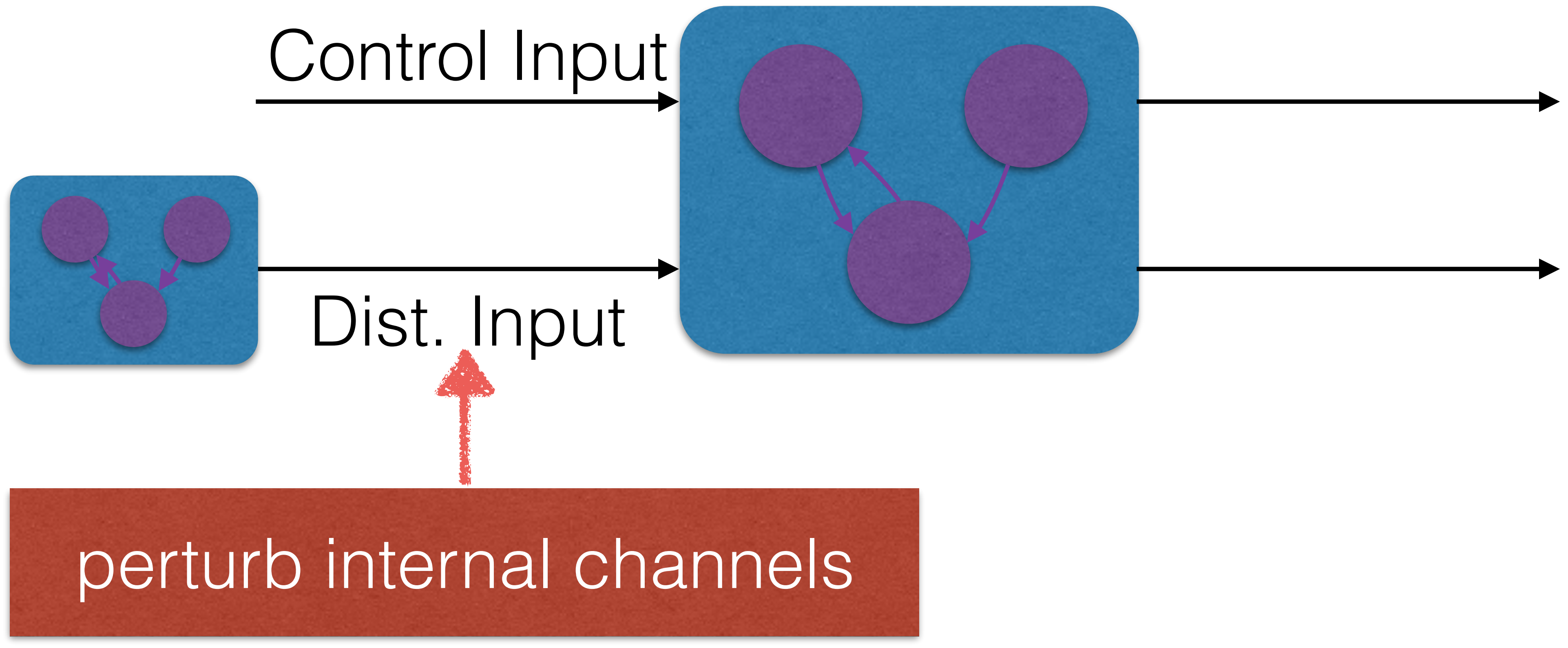
- only for synchronous circuits
- distance not suitable for comparison



Networked Circuits



Equivalent Mealy Machine


Control Input

Dist. Input

perturb internal channels

if last mismatch in disturbance inputs < k
then last mismatch in output < k+b

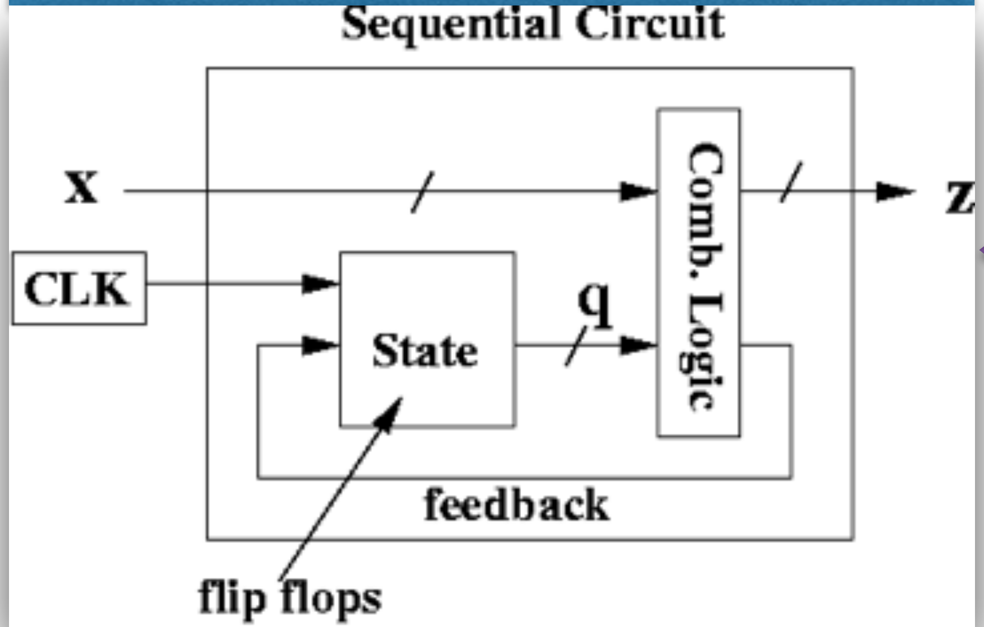
if perturbations ≤ δ **then** error in output channels ≤ ε

0101**0**1010 → 1001**<u>01</u>**00
0101**0**0010 → 1001**<u>10</u>**00
k = 7, b = 1

for **one** perturbation and a fixed Mealy machine,
d(normal output, perturbed output) ≤ b + 1

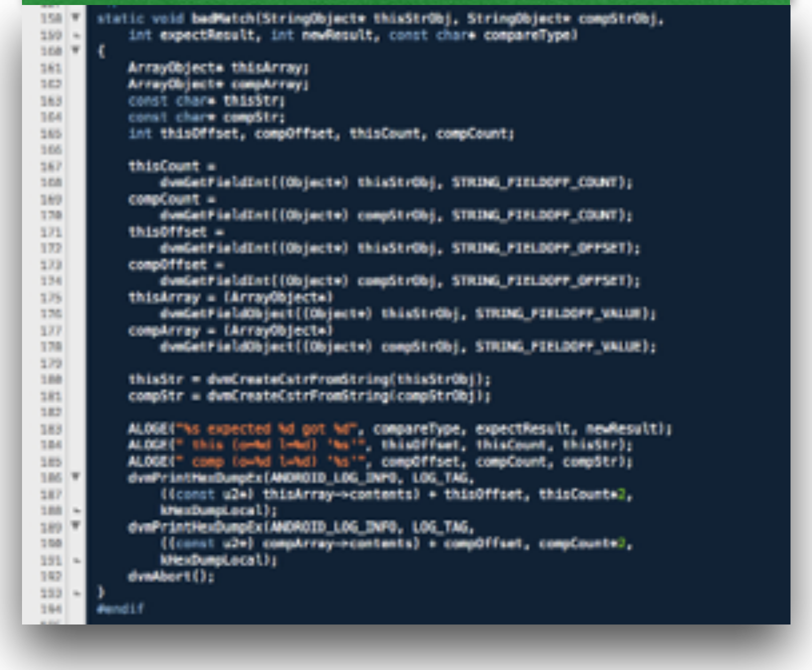
perturbed output **propagates**
to the next input

Circuits



Network Channels

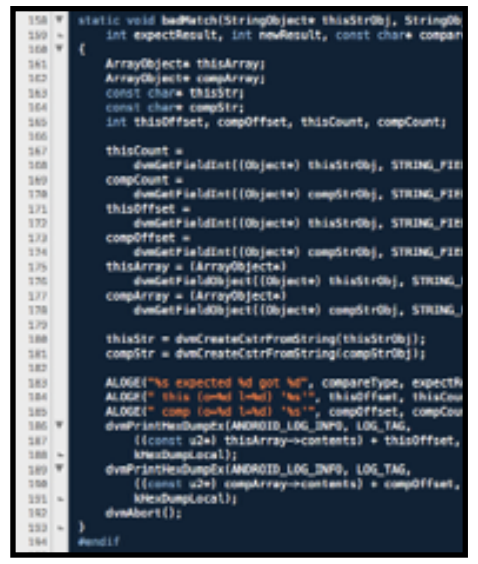
Software



```
static void badMatch(StringObject* thisStrObj, StringObject* compStrObj,
    int expectResult, int newResult, const char* compareType)
{
    ArrayObject* thisArray;
    ArrayObject* compArray;
    const char* thisStr;
    const char* compStr;
    int thisOffset, compOffset, thisCount, compCount;

    thisCount =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIELDOFF_COUNT);
    compCount =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIELDOFF_COUNT);
    thisOffset =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIELDOFF_OFFSET);
    compOffset =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIELDOFF_OFFSET);
    thisArray = (ArrayObject*)
        dvmGetFieldObject((Object*) thisStrObj, STRING_FIELDOFF_VALUE);
    compArray = (ArrayObject*)
        dvmGetFieldObject((Object*) compStrObj, STRING_FIELDOFF_VALUE);

    thisStr = dvmCreateCstrFromString(thisStrObj);
    compStr = dvmCreateCstrFromString(compStrObj);

    ALOGE("%s expected %d got %d", compareType, expectResult, newResult);
    ALOGE(" this (o=%d l=%d) '%s'", thisOffset, thisCount, thisStr);
    ALOGE(" comp (o=%d l=%d) '%s'", compOffset, compCount, compStr);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) thisArray->contents) + thisOffset, thisCount*2,
        kHexDumpLocal);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) compArray->contents) + compOffset, compCount*2,
        kHexDumpLocal);
    dvmAbort();
}
#endif
```

Caches

„Small <u>perturbations</u> to the environment or parameters do <u>not change</u> the <u>observable behavior</u> substantially.“



```
static void badMatch(StringObject* thisStrObj, StringOb
    int expectResult, int newResult, const char* compar
{
    ArrayObject* thisArray;
    ArrayObject* compArray;
    const char* thisStr;
    const char* compStr;
    int thisOffset, compOffset, thisCount, compCount;

    thisCount =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIE
    compCount =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIE
    thisOffset =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIE
    compOffset =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIE
    thisArray = (ArrayObject*)
        dvmGetFieldObject((Object*) thisStrObj, STRING_
    compArray = (ArrayObject*)
        dvmGetFieldObject((Object*) compStrObj, STRING_

    thisStr = dvmCreateCstrFromString(thisStrObj);
    compStr = dvmCreateCstrFromString(compStrObj);

    ALOGE("%s expected %d got %d", compareType, expectR
    ALOGE(" this (o=%d l=%d) '%s'", thisOffset, thisCou
    ALOGE(" comp (o=%d l=%d) '%s'", compOffset, compCou
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) thisArray->contents) + thisOffset,
        kHexDumpLocal);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) compArray->contents) + compOffset,
        kHexDumpLocal);
    dvmAbort();
}
#endif
```

continuous

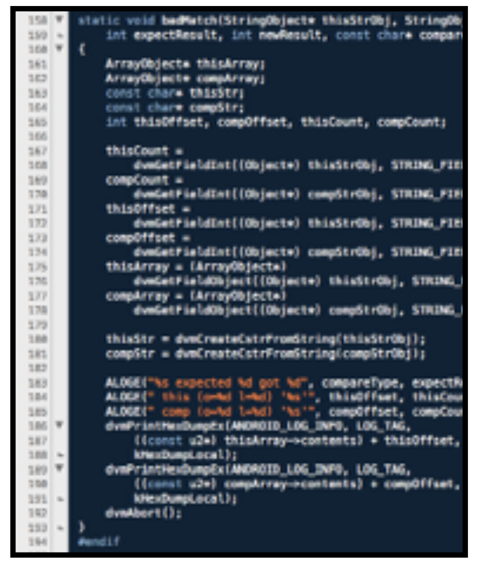
∀ε>0 ∃δ>0: **(**an arbitrarily small change (< δ) to the *input* value x_i
and other inputs identical**)**
must only cause an arbitrarily small change (< ε) to the *output* value x_j

K-Lipschitz continuous

(a change (< ε) to the *input* value x_i
and other inputs identical**)**
can change the *output* value x_j by at most K·ε

Limitations:

- what if only *parts* of a program are continuous
- no divisions
- not applicable to *reactive* and *concurrent systems*



```
static void badMatch(StringObject* thisStrObj, StringOb
    int expectResult, int newResult, const char* compar
{
    ArrayObject* thisArray;
    ArrayObject* compArray;
    const char* thisStr;
    const char* compStr;
    int thisOffset, compOffset, thisCount, compCount;

    thisCount =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIE
    compCount =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIE
    thisOffset =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIE
    compOffset =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIE
    thisArray = (ArrayObject*)
        dvmGetFieldObject((Object*) thisStrObj, STRING_
    compArray = (ArrayObject*)
        dvmGetFieldObject((Object*) compStrObj, STRING_

    thisStr = dvmCreateCstrFromString(thisStrObj);
    compStr = dvmCreateCstrFromString(compStrObj);

    ALOGE("%s expected %d got %d", compareType, expectR
    ALOGE(" this (o=%d l=%d) '%s'", thisOffset, thisCou
    ALOGE(" comp (o=%d l=%d) '%s'", compOffset, compCou
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) thisArray->contents) + thisOffset,
        kHexDumpLocal);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) compArray->contents) + compOffset,
        kHexDumpLocal);
    dvmAbort();
}
#endif
```

(δ,ε)-robustness

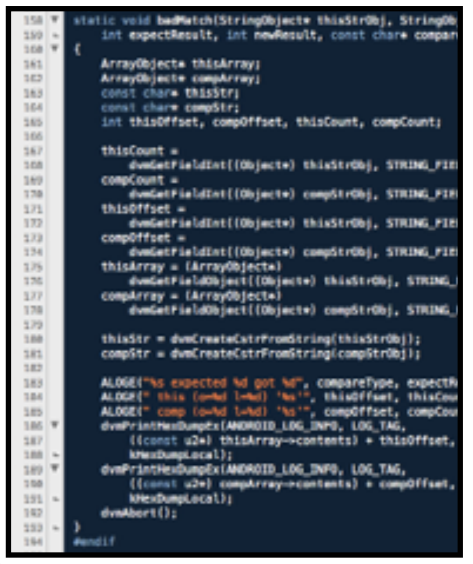
Symbolic Robustness

with respect to the i^{th} input

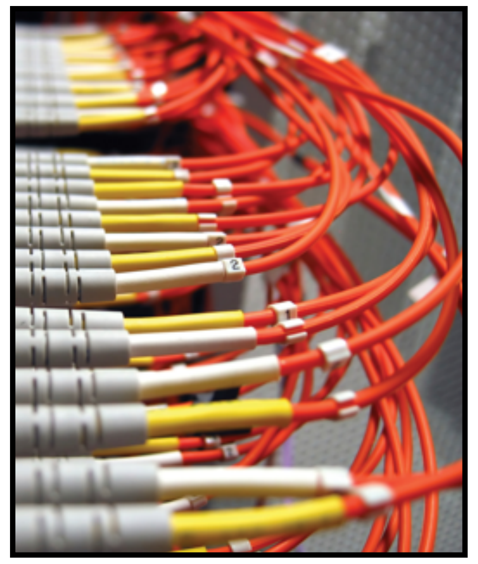
if (difference in the i^{th} input ≤ δ
and other inputs identical**)**
then difference in output ≤ ε

Limitations:

- δ is a constant
- not applicable to *closed loop systems*
- no floating point numbers
- no non-linear arithmetic
- considers only one output



```
static void badMatch(StringObject* thisStrObj, StringOb
    int expectResult, int newResult, const char* compar
{
    ArrayObject* thisArray;
    ArrayObject* compArray;
    const char* thisStr;
    const char* compStr;
    int thisOffset, compOffset, thisCount, compCount;

    thisCount =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIE
    compCount =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIE
    thisOffset =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIE
    compOffset =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIE
    thisArray = (ArrayObject*)
        dvmGetFieldObject((Object*) thisStrObj, STRING_
    compArray = (ArrayObject*)
        dvmGetFieldObject((Object*) compStrObj, STRING_

    thisStr = dvmCreateCstrFromString(thisStrObj);
    compStr = dvmCreateCstrFromString(compStrObj);

    ALOGE("%s expected %d got %d", compareType, expectR
    ALOGE(" this (o=%d l=%d) '%s'", thisOffset, thisCou
    ALOGE(" comp (o=%d l=%d) '%s'", compOffset, compCou
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) thisArray->contents) + thisOffset,
        kHexDumpLocal);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) compArray->contents) + compOffset,
        kHexDumpLocal);
    dvmAbort();
}
#endif
```



Networked System

K-Lipschitz continuous

continuous

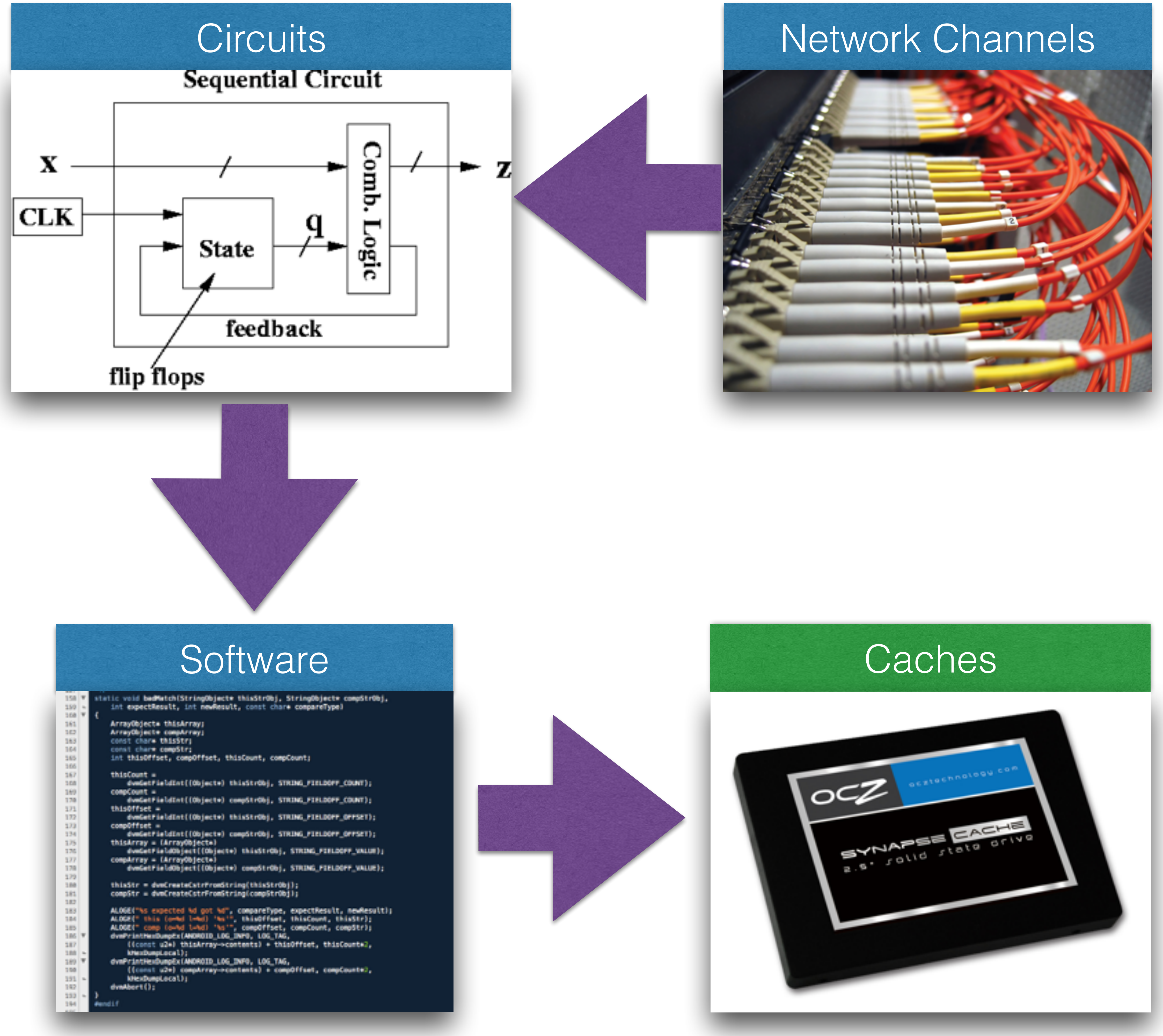
(δ,ε)-robustness
Symbolic Robustness

Distances on sequences of symbols

Distances on datatypes like integers

Not directly applicable in the networked setting!

Interesting Goal:
robustness w.r.t. input/output of networked system



```
static void badMatch(StringObject* thisStrObj, StringObject* compStrObj,
    int expectResult, int newResult, const char* compareType)
{
    ArrayObject* thisArray;
    ArrayObject* compArray;
    const char* thisStr;
    const char* compStr;
    int thisOffset, compOffset, thisCount, compCount;

    thisCount =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIELDOFF_COUNT);
    compCount =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIELDOFF_COUNT);
    thisOffset =
        dvmGetFieldInt((Object*) thisStrObj, STRING_FIELDOFF_OFFSET);
    compOffset =
        dvmGetFieldInt((Object*) compStrObj, STRING_FIELDOFF_OFFSET);
    thisArray = (ArrayObject*)
        dvmGetFieldObject((Object*) thisStrObj, STRING_FIELDOFF_VALUE);
    compArray = (ArrayObject*)
        dvmGetFieldObject((Object*) compStrObj, STRING_FIELDOFF_VALUE);

    thisStr = dvmCreateCstrFromString(thisStrObj);
    compStr = dvmCreateCstrFromString(compStrObj);

    ALOGE("%s expected %d got %d", compareType, expectResult, newResult);
    ALOGE(" this (o=%d l=%d) '%s'", thisOffset, thisCount, thisStr);
    ALOGE(" comp (o=%d l=%d) '%s'", compOffset, compCount, compStr);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) thisArray->contents) + thisOffset, thisCount*2,
        kHexDumpLocal);
    dvmPrintHexDumpEx(ANDROID_LOG_INFO, LOG_TAG,
        ((const u2*) compArray->contents) + compOffset, compCount*2,
        kHexDumpLocal);
    dvmAbort();
}
#endif
```

„Small perturbations to the environment or parameters do not change the observable behavior substantially.“

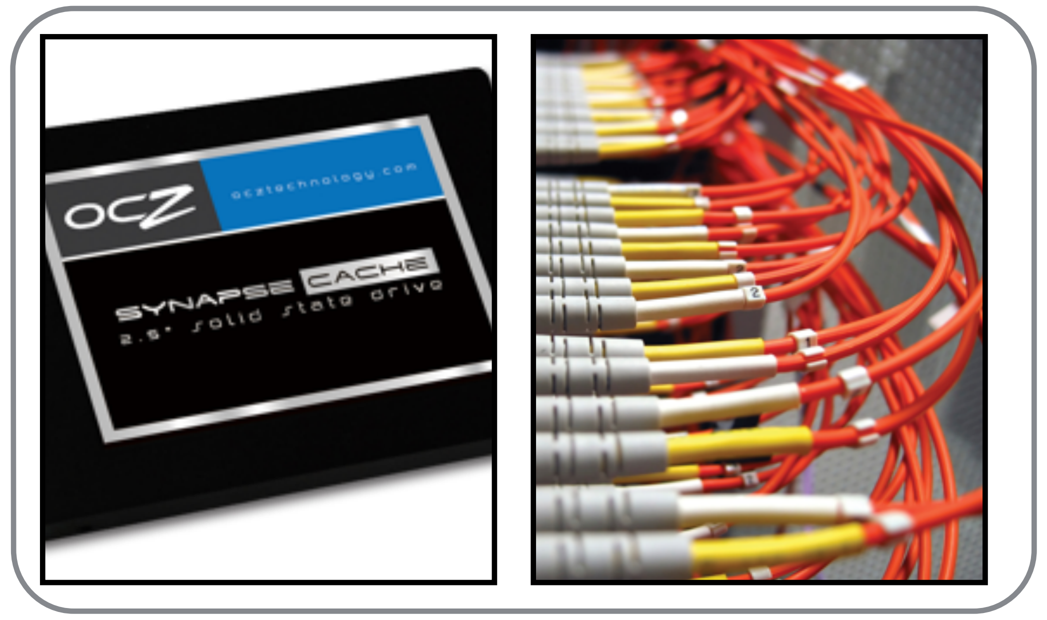
k-miss-sensitivity

same access sequence

$\text{misses}(q,s) \leq k \cdot \text{misses}(q',s) + c$

initial cache state

How does the history influence cache misses?



Multi-level cache models!

(r,c)-robustness

if $d(s,s') < \delta$
then $\text{misses}(s) \leq r(\delta) \cdot \text{misses}(s') + c(\delta)$

How does a changed input sequence influence cache misses?

(r,c)-competitiveness

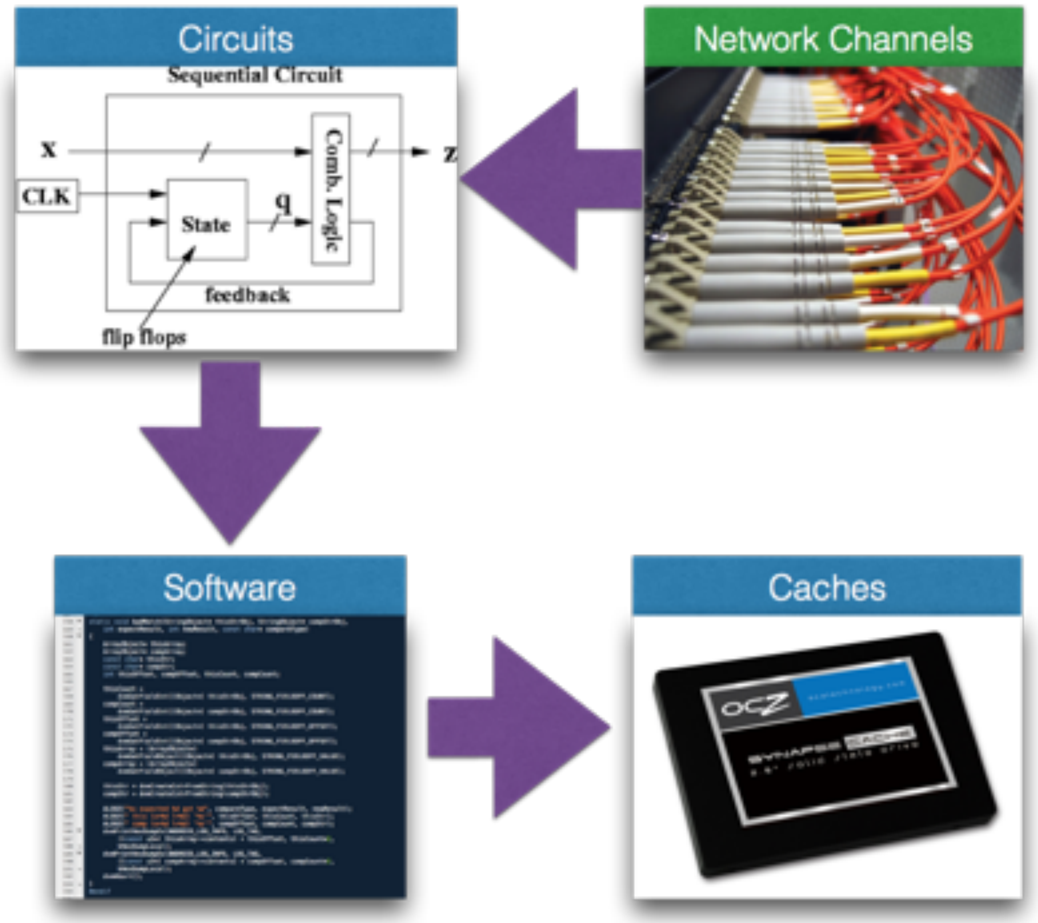
$\text{misses}(s) \leq r \cdot \text{OPT}(s) + c$

misses of optimal offline strategy

Compare to optimal strategy.

Final Conclusion

- Safety critical systems should be robust!
- Many related *robustness* properties,
  - but how to combine them?
- Weaknesses?



„Small perturbations to the environment or parameters do not change the observable behavior substantially.“

Image Sources

- Car (Audi A1) - http://www.extremetech.com/wp-content/uploads/2012/12/Audi-A1.jpg
- Power Plant - http://upload.wikimedia.org/wikipedia/commons/8/8d/Nuclear_Power_Plant_-_Grohnde_-_Germany_-_1-2.JPG
- Aircraft - http://cdns.designmodo.com/wp-content/uploads/2010/09/CivilAircraft_005019.jpg
- Network Cables - http://mms.businesswire.com/media/20130603006748/de/371298/5/Network_Cables_1825894.jpg
- Sequential Circuit - http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Seq/Figs/seq.png
- Sequential Circuit - Robustness of Sequential Circuits, L. Doyen, and T. Henzinger, A. Legay, D. Nickovic, ACSD '10
- Cache - http://portnoy-sw.com/blog/wp-content/uploads/2012/10/synapse_main.jpg