

# Recoverability Preservation: A Measure of Last Resort

Ali Mili, Frederick Sheldon, Fatma  
Mili, Jules Desharnais

# Reflections on Software/ Program Fault Tolerance

Commonly used techniques of fault tolerance:

- *Trigger Happy*. Fire off as soon as the current state is found to be incorrect.
- *Heavy Artillery*. Geared (unnecessarily) towards producing a correct state.
- *Inefficient*. Involve heavy overhead in terms of space (duplicating states) and time (check-pointing etc).
- *Panic Stricken*. Resort to Emergency Measures too soon, on unnecessarily strong conditions.

# Reflections on Software/ Program Fault Tolerance

We advocate a more measured approach:

- *Triggered only when the state is unmaskable.* No false alarms.
- *Aims only to produce a maskable state.* Minimizes computation, and required data.
- *Uses only forward error recovery.* No time/space overhead.
- *Uses the Panic Button as a Last Resort.* Only when the state is unrecoverable.

# Recoverability Preservation

We know how to characterize maskable, unmaskable states, recovery routines. We need to characterize Recoverable States.

Modeling device: We make recoverability not a property of the state but a property of the function that produces it. We call this property: *Recoverability Preservation*.

# Recoverability Preservation: Illustration

A Program/ System structured as the product  
of two components/ functions

P; L:F.

(P: Past; F: Future; L: Label). Expected  
functions:

- $P(x) = x \bmod 6.$
- $F(x) = x \bmod 9 + 12.$

## Illustration, II

- If Past Function is incorrect, and computes
$$P1 = (x \text{ mod } 6 + 18)$$
then states produced by P1 are not correct but they are *maskable* (the excess 18 will be canceled by taking mod 9 in function F).
- No intervention is required.

## Illustration, III

- If Past Function is incorrect, and computes  $P2 = (x \bmod 12)$   
then states produced by P2 are not maskable, but they are *recoverable*.
- Recovery routine: apply (mod 6) to the current state.

## Illustration, IV

- If Past Function is incorrect, and computes  $P3 = (x \bmod 3)$   
then states produced by P3 are not recoverable, but they are *partially recoverable*.
- *Probabilistic Recovery Routine*: return  $x$  (or  $x+3$ ), with 0.5 probability of success.

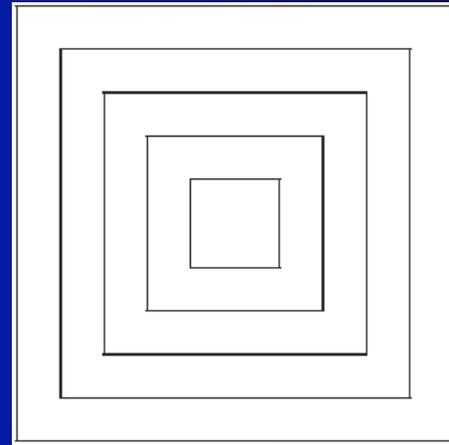
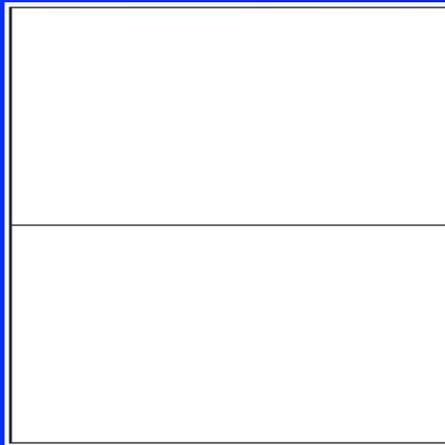
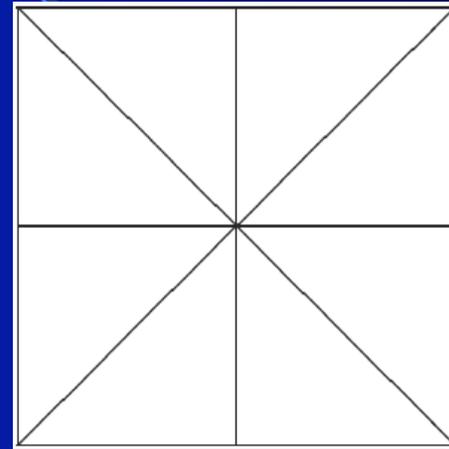
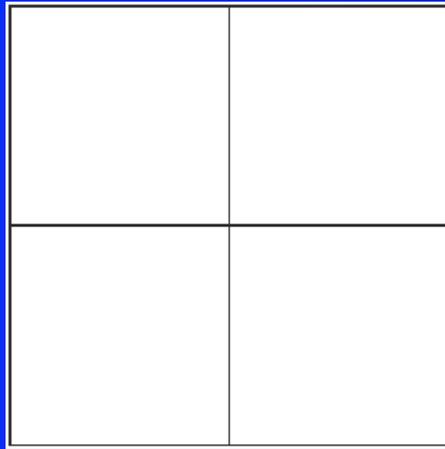
# Illustration, V

- If Past Function is incorrect, and computes  $P4 = (x \bmod 7)$  then states produced by P4 are not recoverable.
- No recovery is possible, for knowing  $(x \bmod 7)$  does not inform us on  $(x \bmod 6)$ .

# Intuitive Analysis

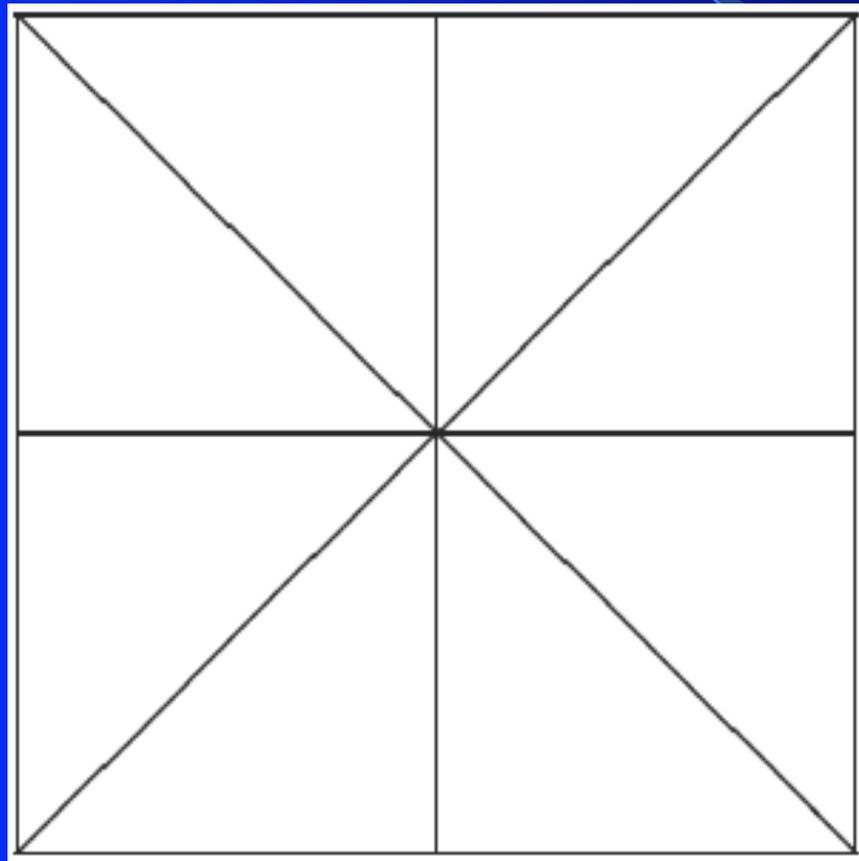
- Q preserves recoverability for P if  $\mu(Q) \subseteq \mu(P)$ , where  $\mu(R) = R R^{\wedge}$  (level sets of R).
- Interestingly: condition involves how Q partitions its domain but does not involve what value Q assigns to each partition.
- If Q assigns the wrong image to a partition, that can be corrected by the recovery routine
- But if Q partitions its domain wrongly (re: mod 7 rather than mod 6) nothing can be done.

# Degrees of Recoverability

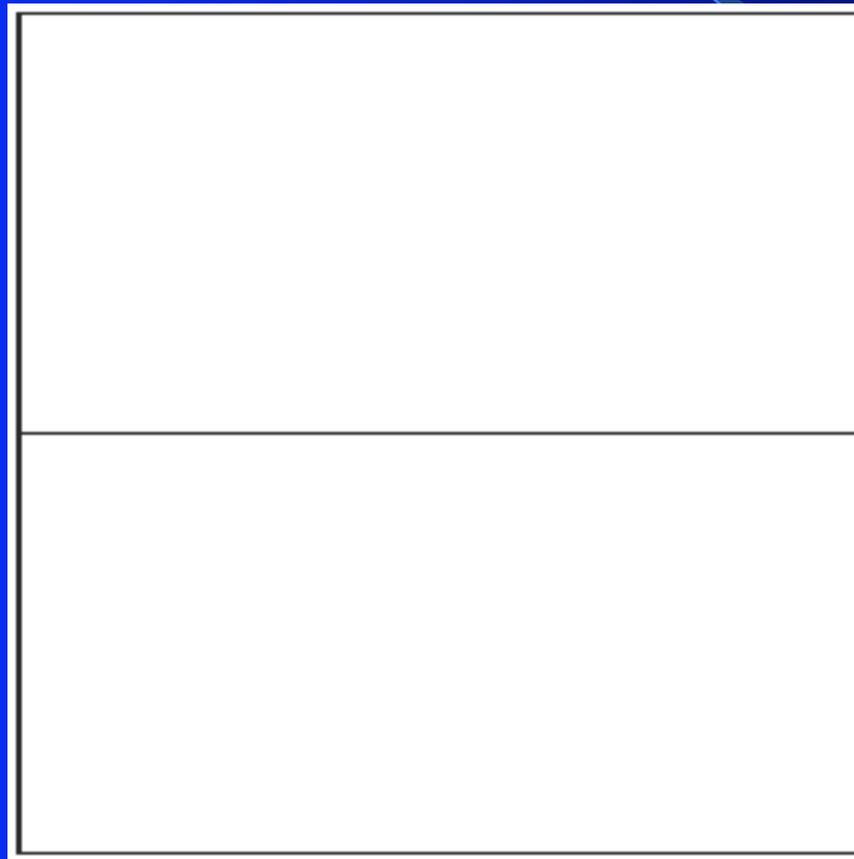


# $P\hat{P}$ for Original $P$

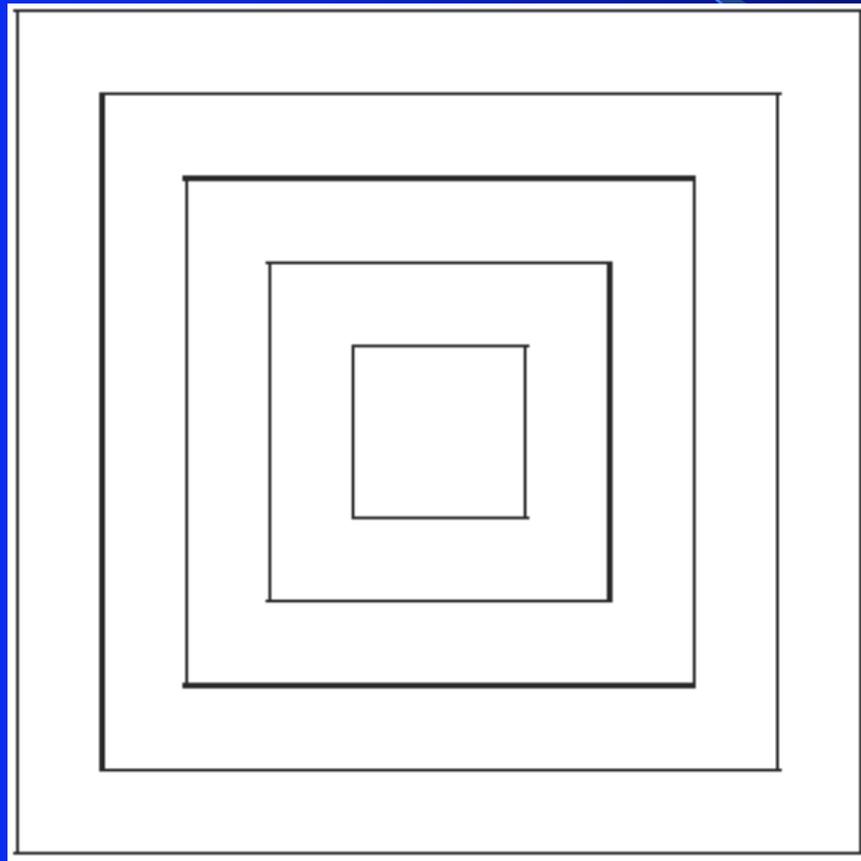

$P_2 \hat{P}_2$ , where  $P_2$   
preserves recoverability



$P_3 \hat{P}_3$ , where  $P_3$   
preserves partial recoverability



$P_4 \hat{P}_4$ , where  $P_4$   
does not preserve recoverability



# Characterizing Recoverability Preservation

- Characterization by  $\mu(Q) \subseteq \mu(P)$  is intuitive, but incomplete.
- For completeness: we must involve the specification  $R$  that the system  $(P; F)$  must refine.
- Because  $R$  is potentially non-deterministic, we get an extra dimension of redundancy (unexplored in the illustrative example).

# Sufficient Conditions

- A past function  $\Pi$  preserves maskability (i.e. produces maskable states) if it refines

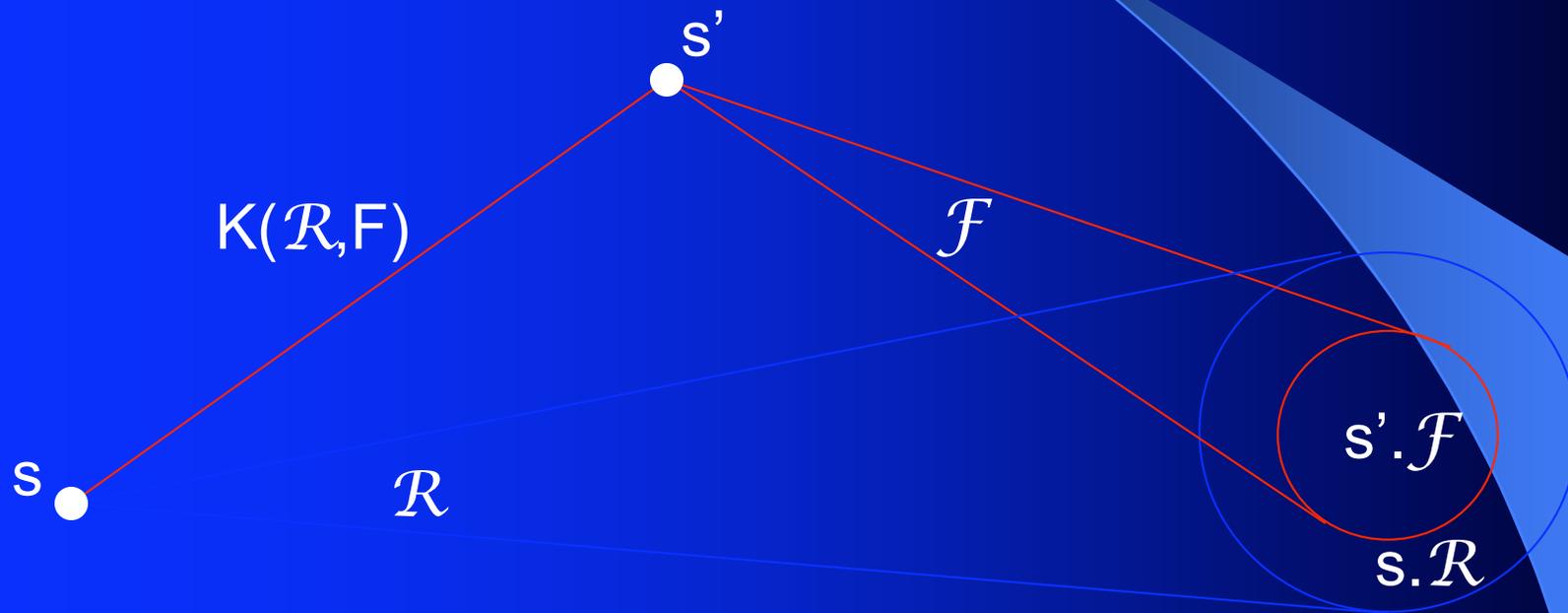
$$\kappa(R, F),$$

where  $\kappa$  is the left quotient operator.

- A past function  $\Pi$  preserves recoverability (i.e. produces recoverable states) if it satisfies the following conditions

$$KL \subseteq \pi L \wedge L \subseteq \overline{(K\hat{L} \cap \pi)} \overline{KL}$$

# Left quotient of $\mathcal{R}$ by $\mathcal{F}$



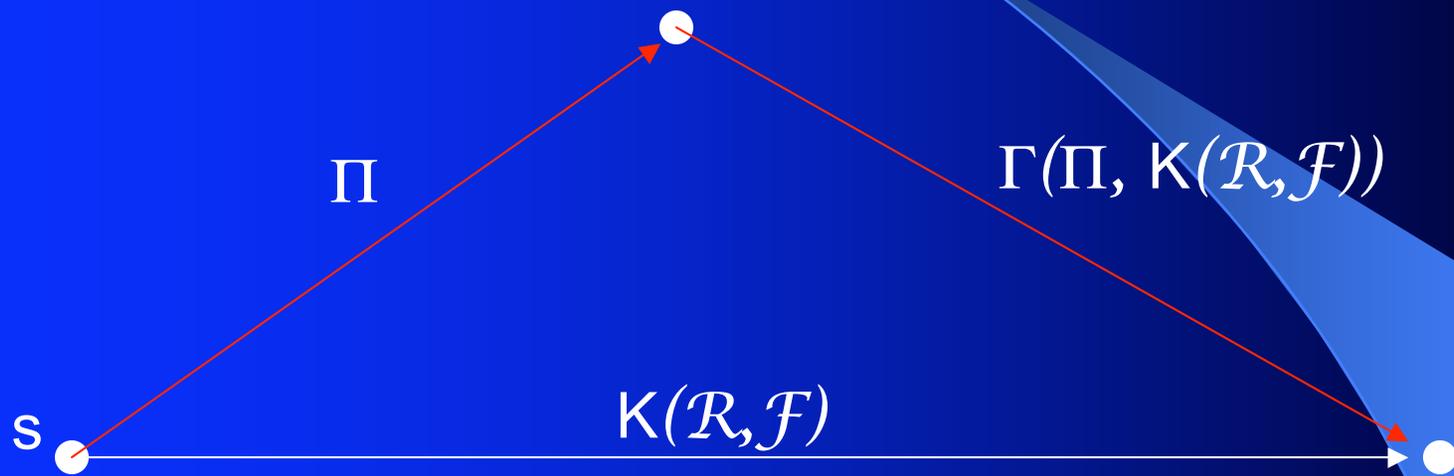
# Specifying the Recovery Routine

- If past function  $\Pi$  preserves recoverability with respect to future function  $F$  and specification  $R$  then

$$r = \Gamma(\Pi, \kappa(R, F))$$

is a specification of the recovery routine, where  $\Gamma$  is the right quotient and  $\kappa$  is the left quotient operator.

- Any routine that refines  $r$  will map recoverable states into maskable states.

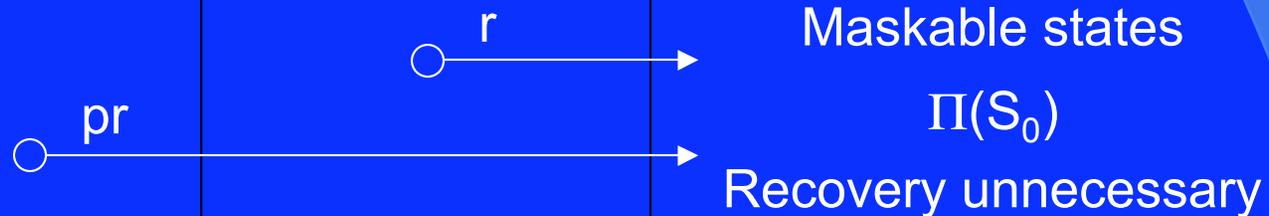


# Hierarchy of Correctness Levels

Unrecoverable states  
→ Recovery insufficient

Partially recoverable states  
→ Probabilistic recovery

Totally recoverable states  
→ Total recovery necessary & sufficient



# Linking to Intuitive Discussion

- If  $R$  is regular ( $R=RR^{\wedge}R$ ) and the following conditions hold

$$RF^{\wedge}L \subseteq \Pi L \wedge \Pi \Pi^{\wedge} \subseteq RR^{\wedge}$$

then  $\Pi$  preserves recoverability.

- Generalizes the condition discussed upon inspecting the sample example.

# Application: Lean fault Tolerance

```
If not maskable(s) then recovery-  
measures(s);
```

```
recovery-measures(s) :
```

```
If recoverable(s) then deterministic-  
recovery(s)
```

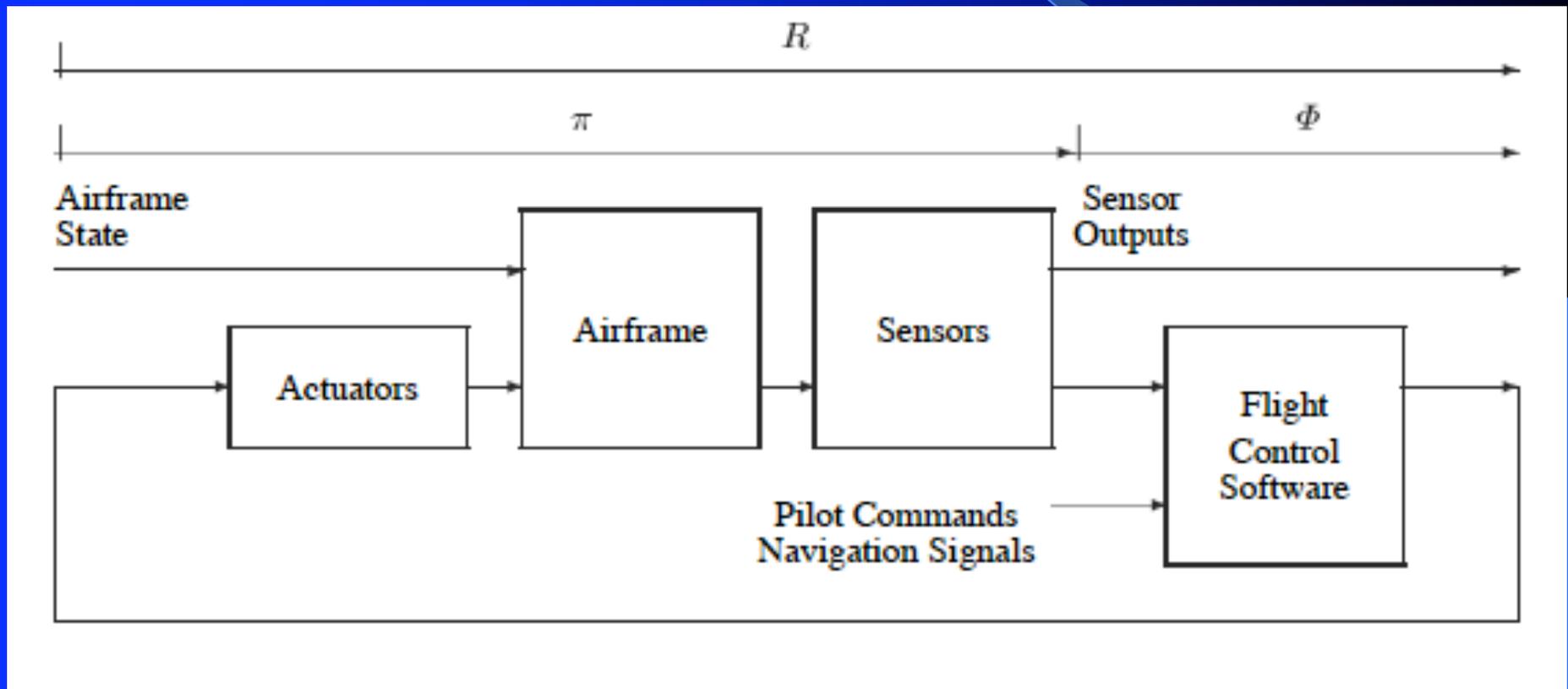
```
else
```

```
  If partially-recoverable(s)  
    then probabilistic-recovery(s)  
    else failure(s);
```

# Recoverability Preservation, a Substitute for Correctness

- Prove recoverability preservation.
- Takes steps to recover.
- Substitutes/ complements correctness proofs.
- Using safety condition for R.

# Flight Control Loop



# Characterizing Fault Modes

- Fault Tolerant Flight Control System: A system that can recover from some types of faults, including loss of sensors, loss of flight surfaces, loss of control of actuators.
- When these faults arise, the system must alter its control law and make up for fault.

# Characterizing Fault Modes

- Question: Which sensor-aircraft-actuator faults can be handled by fault tolerant FCS?
- Those for which the aggregate sensor-aircraft-actuator preserves recoverability.
- A highly speculative answer, we acknowledge; perhaps difficult to model.

# Concluding Remarks

- Introduced idea of recoverability preservation.
- Shown its use for a more measured, more efficient approach to fault tolerance.
- Shown its application for fault modeling.
- Genesis of the idea: analyzing a fault tolerant flight control system (tolerates damage to flight surfaces).