The Secret History of System Stabilization

Mohamed G Gouda
University of Texas at Austin
gouda@cs.utexas.edu

Tutorial SSS October 2012

Happy Birthday System Stabilization!

- The area of system stabilization is about 40 years old. (Dijkstra's original paper was published in CACM in 1973.)

- In those 40 years, researchers have introduced some 40 variant definitions of system stabilization. Some of these variant definitions are weaker and some are stronger than Dijkstra's original definition

- Unfortunately, many young (and some old) researchers in the area of system stabilization are not fully aware of this rich history of the area

In This Tutorial

- We provide a mature and sober look at seven variant definitions of system stabilization:
  - Fault-tolerance
  - Silent Stabilization
  - Fault Containing Stabilization
  - Fault Masking Stabilization
  - Weak Stabilization
  - Multiphase Stabilization
  - Security
- We focus on why these variant definitions were introduced

- Confession: I am coauthor of six of these definitions. But this speaks to my laziness not vanity

System States

- A system M is defined by a nonempty set of variables and a set of actions
- A state s of M is defined by one value for each var of M
- A state predicate P of M is a function that assigns a Boolean value, false or true, to each state of M
- Let T denote the state predicate that assigns true to each state of M
- Any state of M, where P is true, is called P-state of M

System Computations

- A transition of M is a pair (s1, s2) of M states where an M action is enabled for execution at s1 and executing this action when M is at s1 yields M at s2

- A computation c of M is a maximal sequence of M states (s1, s2, ..), where each pair of consecutive states in c is a transition of M

Closure

- A state predicate P of system M is said to be closed in M iff for every M transition (s1, s2), if s1 is a P-state, then s2 is a P-state

Theorems of Closure

- Theorem 1:

Let T be the state predicate whose value is true at each state of M. Then, T is closed in M

- Theorem 2:

If P1 is closed in M and
P2 is closed in M
then P1 v P2 is closed in M and
P1 ^ P2 is closed in M

Convergence

- Let P and Q be closed predicates in system M. P converges to Q in M iff for every P-state s of M and every M computation c that starts at s, computation c has a Q-state

Theorems of Convergence

- Theorem 3:

  T converges to T in M

- Theorem 4:

If P1 converges to Q1 in M and

P2 converges to Q2 in M

then P1 v P2 converges to Q1 v Q2 in M and

P1 ^ P2 converges to Q1 ^ Q2 in M

- Theorem 5:

If P converges to Q in M and

Q converges to R in M

then P converges to R in M

Stabilization

- Let M be a system
- Let Q be a state predicate in M
- M stabilizes to Q iff Q is closed in M and T converges to Q in M,
- where T is the state predicate whose value is true at each state of M

Theorems of Stabilization

- Theorem 6:

  M stabilizes to T

- Theorem 7:

If M stabilizes to Q1 and

Q2 is closed in M

then M stabilizes to Q1 v Q2

- Theorem 8:

If M stabilizes to Q1 and

M stabilizes to Q2

then M stabilizes to Q1 ^ Q2

System Composition

- Let M1 and M2 be systems with the same vars and different actions

- Let M1UM2 denote the system whose set of vars is the same as that of M1 (or of M2) and whose set of actions is the union of M1 actions and M2 actions

Theorems of System Composition

- Theorem 9:

If P is closed in M1 and
P is closed in M2
then P is closed in M1UM2

- Theorem 10:

If P converges to Q in M1 and
P converges to Q in M2
then P may NOT converge to Q in M1UM2

- Theorem 11:

If M1 stabilizes to Q and
M2 stabilizes to Q
then M1UM2 may NOT stabilize to Q

Hierarchical Composition

- Let M1, M2, and M1UM2 be systems as defined before ..
- Let Q1 be a closed predicate in M1
- System M1UM2 is said to be Q1-hierarchy iff M1 and M2 satisfy the following two conditions:

  Any var, that is written or read by any M1 action, is not written by any M2 action

  Any var, that is written by any M1 action and read by any M2 action, does not change its value in any M1 transition (s1, s2), where s1 is a Q1-state

Theorem of Hierarchical Composition

- Theorem 12:

If M1 stabilizes to Q1 and
M2 stabilizes to Q2 and
M1UM2 is Q1-hierarchy
then M1UM2 stabilizes to Q1 ^ Q2

Concept of Stabilization Is Attractive ..

- It is simple to define .. no need to define faults or adversaries
- It is based on a very abstract definition of systems .. no distinction between distributed and centralized systems
- It supports a high degree of tolerance of a rich class of faults

Concept of Stabilization is Problematic ..

- Does not support tolerance of different classes of faults
- Does not support fault detection
- Does not support fast recovery from special classes of faults
- Complicates system composition
- Complicates system implementation
- Complicates system verification
- Does not support specification of security (which is strengthening of fault tolerance)

Tolerance of Different Fault Classes

- Stabilization, which is based on the concepts of closure and convergence, defines tolerance of one class of fault

- Can we use these concepts (of closure and convergence) directly to define tolerance of many classes of faults (e.g. stuck-at faults)?

- The answer is YES!

Arora and Gouda, in IEEE Transactions on Software Engineering in 1993.

Associated Fault Systems

- To state that a system M tolerates a new class of fault, associate with system M another system F called the associated fault system of M

- The set of vars of system F is the same as that of system M

- Each action of system F is different from all actions of system M

General Fault-Tolerance

- Let M be a system
  F be a fault system associated with M
  Q be a state predicate of M (or F)

M is F-tolerant to Q iff there is a state predicate P of M (or F) that satisfies the following three conditions

Every Q-state is a P-state

P is closed in M and in F and Q is closed in M

P converges to Q in M

Explaining Fault-Tolerance

- Before the occurrence of any fault in system F, system M goes through its Q-states , which are also P states (since Q is closed in M and each Q-state is a P-state)

- When faults in system F start occurring, system M may leave its Q-states but remains in its P-states (since P is closed in both M and F)

- After faults in system F stop occurring for sometime, system M converges from its P-states to its Q-states and remains in Q-states (since P converges to Q in M)

Special Types of Fault Tolerance

- M is F-tolerant to Q iff there is a state predicate P of M (or F) that satisfies ..
- If P is the state predicate T whose value is true at each state of M, then the fault tolerance is called stabilizing
- If P is the same state predicate Q, then the fault tolerance is called masking

Fault Detection in Stabilizing Systems

- A stabilizing system may recover from many faults without detecting that any fault has occurred

- Can we design stabilizing systems that can detect fault occurrences?

- The answer is YES, sometimes, by adopting two special styles of stabilization:

  Stabilization to fixed points

  Stabilization to silent points (Dolev, Gouda, Schneider, in Acta Informatica 1999)

Fixed points

- Let Q be a closed state predicate of a system M

- Predicate Q is a fixed point in M iff for every M transition (s1, s2), if s1 is a Q-state, then the value of every M var at s1 equals the value of the same var at s2

Detecting All Faults Using Fixed Points

- Let M be a system that stabilizes to a fixed point Q
- Any change in the value of some M var indicates that some fault has occurred shortly before the change
- If the values of all M vars remain unchanged for a long time period, then no fault has occurred during most of this period

Silent points

- Let M be a system that consists of processes which communicate by writing and reading shared vars
- Thus each var in M is either local (i.e. written and read by only one process) or shared (i.e. written by one process and read by another)
- Let Q be a closed state predicate of system M
- Predicate Q is a silent point in M iff for every M transition (s1, s2), if s1 is a Q-state, then the value of every shared M var at s1 equals the value of the same var at s2

Detecting Global Faults Using Silent Points

- Let M be a system that stabilizes to a silent point Q
- Any change in the value of some shared M var indicates that some global fault has occurred shortly before the chnage
- If the values of all shared M vars remain unchanged for a long time period, then no global fault has occurred during most of this period

Cost of Silent Stabilization

- Let M be a system that performs one of the following functions: finds the center of a graph, elects a leader, or constructs a spanning tree in a graph
- Assume that M stabilizes to a silent point Q
- Then each shared var in M consists of at least Omega(log n) bits, where n is the number of processes in M

Fast Recovery

- A stabilizing system is guaranteed to recover from a large class of faults
- But its recovery time from any subclass of these faults is not guaranteed to be small
- Need to design stabilizing systems whose recovery time from some specified subclass of faults is guaranteed to be small

Fast Recovery of Stabilizing Systems

- The time for a stabilizing system M to recover from a subclass of faults is small iff one of the following two conditions holds:

- Fault Containing Stabilization:

The recovery time is O(1) steps independent of the number of processes in system M

- Fault Masking Stabilization:

The recovery time is zero steps when system M masks the effects of faults

Fault Containing Stabilization

- Ghosh, Gupta, Herman, Pemmaraju, in Distributed Computing 2007

- Present transformer that can transform any system M, that stabilizes to a fixed point Q, into system M' that has the same number of processes as M

- M' stabilizes to Q. Also M' is guaranteed to recover from any fault that changes the values of vars in only one process in O(1) steps

- Space overhead of M' is O(number of processes in M)

Fault Masking Stabilization

- Gouda, Cobb, Huang, in SSS 2006

- Present transformer that can transform any system M, that stabilizes to Q, into tri-redundant version M', where each var x in M is replaced by three vars (x, x', x")

- M' stabilizes to Q ^ (for every var x, x=x' ^ x'=x"). Also M' is guaranteed to mask any fault that changes the value of only one var in every three vars (x, x', x")

- Space overhead of M' is O(2*number of vars in M)

Tri-Redundant Version M’ of M

- Replace each var x in M by three vars (x, x’, x”) in M’
- Add the conjunct (x=x’ ^ x’=x”) to the guard of each action that reads var x in M’
- Replace each assignment (x := Expression) by the assignment ((x, x’, x”) := Expression) in each action in M’
- For each var x in M, add an action whose guard is (x=!x’ v x’=!x”) and whose assignment is ((x, x’, x”) := MJR(x, x’, x”))

Can Stabilizing Systems Be Implemented?

- Any implementation of a stabilizing system M may introduce delays to M which may cause the implementation to become non-stabilizing

- What to do in face of this negative observation?

- Answer: Be content with a weak notion of stabilization.

Gouda, in SSS 2001

- But first, we need to define weak convergence

Weak Convergence

- Let P and Q be two closed predicates in a system M
- P weakly converges to Q in M iff for every P-state s, there exists an M computation that starts at s and has a Q-state

Theorems of Weak Convergence

- Theorem 13:

  T weakly converges to T in M

- Theorem 14:

If P1 weakly converges to Q1 in M and
P2 weakly converges to Q2 in M
then P1 v P2 weakly converges to Q1 v Q2 in M and
P1 ^ P2 weakly converges to Q1 ^ Q2 in M

- Theorem 15:

If P weakly converges to Q in M and
Q weakly converges to R in M
then P weakly converges to R in M

Weak Stabilization

- Let M be a system
- Let Q be a state predicate of M
- M weakly stabilizes to Q iff Q is closed in M and T weakly converges to Q in M,
- where T is the state predicate whose value is true at each state of M

Theorems of Weak Stabilization

- Theorem 16:

  M weakly stabilizes to T

- Theorem 17:

If M weakly stabilizes to Q1 and

Q2 is closed in M

then M weakly stabilizes to Q1 v Q2

- Theorem 18:

If M weakly stabilizes to Q1 and

M weakly stabilizes to Q2

then M weakly stabilizes to Q1 ^ Q2

System Composition .. As Before

- Let M1 and M2 be systems with the same vars and different actions

- Let M1UM2 denote the system whose set of vars is the same as that of M1 (or of M2) and whose set of actions is the union of M1 actions and M2 actions

Theorems of System Composition

- Theorem 19:

If P weakly converges to Q in M1 and
Q is closed in M2
then P weakly converges to Q in M1UM2

- Theorem 20:

If M1 weakly stabilizes to Q and
Q is closed in M2
then M1UM2 weakly stabilizes to Q

Theorem of Hierarchical Composition

- Theorem 21:

Let M1, M2, and M1UM2 be defined as before

If M1 stabilizes to Q1 and
M2 stabilizes to Q2 and
Q1 is closed in M2
then M1UM2 weakly stabilizes to Q1 ^ Q2

Adding Delays in System Implementations

- Implementing a system usually involves adding delays to one or more vars in the system
- Adding delay to var v in M yields another system denoted M<v>
- Show, by example, that some M stabilizes to Q but M<v> does not stabilize to Q
- We hoped to show that if M weakly stabilizes to Q then M<v> weakly stabilizes to Q. (But not quite!)

Adding Delays

- Adding delay to var v in a system M consists of three steps:

  Add a new var dv to M

  Make each M action that reads var v, read var dv instead

  Add a new action dv := v to M

- The resulting system after adding delay to var v in system M is denoted M<v>

v-Exclusivity

- Let M be a system where each var v is written by at most one action called the v-action and
  Q be a state predicate of M and
  v be a var in M

- Predicate Q is v-exclusive in M iff for every Q-state s, if the v-action in M, if any, is enabled at s, then no other action in M is enabled at s

Adding Delays Preserves Weak Stabilization

- Theorem 22:

Let M be a system where each var is written by at most one action and where each computation is infinite

Q be a closed predicate in M

v be a var in M

If M weakly stabilizes to Q and

Q is v-exclusive in M

then M<v> weakly stabilizes to (Q and (dv=v or G.v))

where G.v is the negation of the disjunction of the guards of all actions, other than v-action and dv-action, in M<v>

Weak Stabilization Approximates Stabilization

- A computation c of a system M is strongly fair iff for every M transition (s1, s2), if state s1 occurs infinitely many times in c, then transition (s1, s2) occurs infinitely many times in c

- System M stabilizes to Q under strong fairness iff Q is closed in M and for every M state s, every strongly fair computation that starts at s, has a Q-state

- Theorem 23:

If M weakly stabilizes to Q and

M has a finite number of states

then M stabilizes to Q under strong fairness

Proving Stabilization and Weak Stabilization

- To prove stabilization,

prove closure and convergence

- To prove weak stabilization,

prove closure and weak convergence

- Proving closure is usually easy

- Next, we show that proving convergence is much harder than proving weak convergence

Proving Convergence

- To prove that P converges to Q in M, do ..
- exhibit a function F that assigns to each M state s a positive integer, denoted F.s, s then
- show that for every P-state s1 and for every M transition (s1, s2),

  F.s1 > F.s2 or s2 is a Q-state

Proving Weak Convergence

- To prove that P weakly converges to Q in M, do ..
- exhibit a function F that assigns to each M state s a positive integer, denoted F.s, then
- show that for every P-state s1, there exists an M transition (s1, s2) where

  F.s1 > F.s2 or s2 is a Q-state

- Because proving for every is harder than proving there exists, proving stabilization is harder than proving weak stabilization

Multiphase Stabilization

- Gouda, in IEEE Transactions on Software Engineering 2002
- Multiphase stabilization is another weakening of stabilization intended to simplify proving of stabilization
- Indeed, proving stabilization is harder than proving multiphase stabilization

Security

- Security is a strengthening of fault tolerance, which can specified (as discussed above) in terms of two concepts: closure and convergence

- Can we strengthen these two concepts, by adding a third concept of safety, in order to specify security?

- The answer is Yes!

Gouda, in Information Processing Letters 2001

Safety

- Let M be a system
  V be a subset of vars of M
  P, Q be two state predicates of M

M is V-safe from P to Q iff following four conditions hold:

Every Q-state is a P-state

Both P and Q are closed in M

P converges to Q in M

Safety:

No var in V is written in a transition (s1, s2) where s1 is a P-state but not a Q-state

Explaining Safety

- Before an adversary attacks, system M goes through its Q-states , which are also P states (since Q is closed in M and each Q-state is a P-state)
- When an adversary D attacks, system M may leave its Q-states but remains in its P-states (since P is closed in both M and D)
- After D stops its attack, system M converges from its P-states to its Q-states and remains in Q-states (since P converges to Q in M)
- None of the critical V vars is written until system M is at a Q-state (because of the safety condition)

Associated Adversaries

- To state that a system M can defend its security, i.e. provide safety for its critical vars, against certain adversary, associate with system M another system D called the associated adversary of M

- The set of vars of system D is the same as that of system M

- Each action of system D is different from all actions of system M

Security Against Adversaries

- Let M be a system
  V be a subset of vars of M
  P, Q be two state predicates of M
  D be an adversary associated with M

M is V-secure from P to Q against D iff following three conditions hold:

M is V-safe from P to Q

P is closed in D

No var in V is written in any D transition (s1, s2) where s1 is a P-state

Theorems of Security

- Theorem 24:

If Q is closed in M and
V* is the set of all vars in M and
E is the adversary associated with M and has no actions

then M is V*-secure from Q to Q against E

- Theorem 25:

If M is V-secure from P to Q against D1 and
M is V-secure from P to Q against D2

then M is V-secure from P to Q against D1UD2

More Theorems of Security

- Theorem 26:

If M is V-secure from P1 to Q1 against D and
M is V-secure from P2 to Q2 against D

then M is V-secure from P1 ^ P2 to Q1 ^ Q2 against D

- Theorem 27:

If M is V-secure from P to Q against D and
M is V-secure from Q to R against D

then M is V-secure from P to R against D

Conclusions

- The original definition of system stabilization is too strong (and so it is not easy to verify, compose, design, or implement) to be practically useful
- Strengthening this definition even further may be intellectually interesting, but it does not address the issue of practical use
- To address the issue of practical use, we need to look for weaker variants of system stabilization
- My contribution in this regard consists of introducing Weak and Multiphase Stabilization. I still believe in these variants