On Emergent Misbehavior

John Rushby
With help from Hermann Kopetz

Computer Science Laboratory
SRI International
Menlo Park CA USA

The Basic Idea

- We build systems from components, but systems have properties not possessed by their individual components
- Emergence is the idea that complex systems may possess qualities that are different in kind than those of their components: described by different languages (ontologies)
  - e.g., velocities of atoms vs. temperature of gas
  - e.g., neural activity in the brain vs. thoughts in the mind

  Quality is used as a generic term for the result of emergence: behavior, structure, patterns, etc.
- Systems where macro qualities are straightforward consequences of the micro level are called resultant

Overview

- There's good emergence and bad
- In particular, complex systems can have failures not predicted from their components, interactions, or design
- Call this Emergent Misbehavior
- I'm interested in emergent misbehavior and how to control it
- I suspect "emergence" here is more glitter than substance
- But I'll start by outlining traditional emergence
- Then get on to misbehavior
- And a Crazy Idea

Emergence

Two key ideas

- Downward Causation: interactions at the macro level propagate back to the micro level
  - e.g., flock flowing around an obstruction: individuals respond to actions of neighbors
  - Micro behavior seems stochastic
  - Macro behavior is systemic
- Supervenience: there can be no difference at the macro level without a difference at the micro level
  - If I have a new idea, my neural state must change
  - But different micro states may correspond to the same macro state

  i.e., macro states are a surjective function of micro states

Strong and Weak Emergence

- What I just described is sometimes called strong emergence
  - Not obvious you can compute macro behavior from micro
- In contrast to weak emergence
  - Asserts you can compute macro behavior from micro, but only by simulation
  - i.e., there's no accurate description of the system simpler than the system itself
- Weak emergence is an attempt to eliminate downward causation
  - Because it looks like something from nothing
  - Because it is epiphenomenal (sterile side-effect)
- But then weak emergence just looks like another name for behavior that is unexplained (by our current theories)

Is Emergence Relative?

- Emergence is relative to our models or theories for how macro qualities derive from the micro level
- So weak emergence is just a reflection of ignorance
  - i.e., of the weakness of our current theories and models
- Note that we can have theories for emergent qualities without being able to explain their emergence from the micro level
  - e.g., chemistry prior to quantum mechanics
- Even when we can predict macro qualities from micro models, that's not always the best way to proceed
  - We have statistical thermodynamics, but we still use Boyle's Law

Is Emergence Relative? (ctd.)

- Even strong emergence can be “explained” by adding new details to models of micro behavior
- e.g., traffic jams, which look emergent
  - New rule: in heavy traffic, faster cars cannot overtake slower ones, so they have to brake
    - ⋆ This reflects/encodes downward causation
  - More sophisticated models predict phantom traffic jams (standing waves, or solitons)
- So, qualities are emergent until we learn how to explain then, then they become resultant
- cf. Quantum Mechanics and downfall of British Emergentism
- Emergent qualities are ontologically novel (at least, in this domain), so revision to micro-level theory may be substantial
- So. . . ?

Emergent Misbehavior

- There's good emergence and bad
- In particular, complex systems can have failures not predicted from their components, interactions, or design
- Emergent or just unexpected?
- Probably the latter, but in sufficiently complicated contexts it may be useful to consider these failures as different in kind than the usual ones
- Maybe some are due to downward causation
- In any case, possibly a useful new way to look at failures

Examples

- Jeff Mogul's paper:
  - Mostly OS and network examples concerning performance and fairness degradation rather than outright failure
  - e.g., router synchronization
  - Note that these properties are expressed in the language of the emergent system, not the components
  - Like phantom traffic jams
- Feature interaction in telephone systems
- West/East coast phone and power blackouts
- 1993 shootdown of US helicopters by US planes in Iraq
- Überlingen mid-air collision

Even "Correct" Systems Can Exhibit Emergent Misbehavior

- We have components with verified properties, we put them together in a design for which we require properties P, Q, R, etc. and we verify those, but the system fails in operation. . . how?
- There's a property S we didn't think about
  - Maybe because it is ontologically novel: needs to be expressed in a new language of the emergent system, not in the language of the components
  - If we'd tried to verify it, we'd have found the failure
  - But it's hard to anticipate all the things we care about in a complicated system
- Call these unanticipated requirements
- Note that S could be negated (i.e., a property we don't want)

Even "Correct" Systems Can Exhibit Emergent Misbehavior (ctd.)

- We verified that interactions of components A and B deliver property P and that A and C deliver Q, taking care of failures appropriately: $A||B \vdash P$, $A||C \vdash Q$
- But there's an interaction we didn't think about
  - We didn't anticipate that some behaviors of C (e.g., failures) could affect the interactions of A and B, hence P is violated even though A and B are behaving correctly (and so is C, wrt. the property Q): $A||B||C \not\vdash P$
- That's why FAA certifies only complete airplanes and engines
- Call these unanticipated interactions (or overlooked assumptions)

Causes of Emergent Misbehavior

- I think they all come down to ignorance
  - Or epistemic uncertainty
- There are no accurate descriptions of some complex systems simpler than the system itself (recall weak emergence)
- But all our analysis and verification are with respect to abstractions and simplifications, hence we are ignorant about the full set of system qualities
- More particularly, we may be ignorant about
  - The complete set of requirements we will care about in the composed system
  - The complete set of behaviors of each component
  - The complete set of interactions among the components

How to Eliminate or Control Emergent Misbehavior

- Identify and reduce ignorance
- Eliminate or control unanticipated behaviors and interactions
  - i.e., deal with the manifestations of ignorance
- Engineer resilience
  - i.e., adapt to the consequences of ignorance

Identify and Reduce Ignorance

Vinerbi, Bondavalli, and Lollini propose tracking ignorance as part of requirements engineering

- Quantify it (qualitatively, e.g., low, medium, high)
- Have rules how it propagates though AND and OR etc.
- If it gets too large, consider replacing a source of high ignorance (e.g., COTS, or another system) by a better-understood and more limited component

Identify and Reduce Ignorance (ctd. 1)

- There are other fields where epistemic uncertainty plays a central rôle: particularly, safety
  - Have to try and think of everything
  - And deal with it
- Everything raises epistemic uncertainty
- Hazard analysis is about systematic ways to explore everything
- But I think it can be put on a more formal footing
  - And that automated support is needed and feasible
- There are some promising avenues for doing this
  - e.g., model checking very abstract designs
  - Using SMT solvers for infinite bounded model checking with uninterpreted functions
- Distinguish the (formal) verification and the safety case
  - Safety case addresses epistemic uncertainty in verification

Identify and Reduce Ignorance (ctd. 2)

- Black and Koopman observe that safety goals are often emergent to the system components
- e.g., the concept (no) “collision” might feature in the top-level safety goal for an autonomous automobile
- But “collision” has no meaning for the brake, steering, and acceleration components
- They suggest identifying local goals for each component whose conjunction is equivalent to the system safety goal, recognizing that some unknown additional element X may be needed (because of emergence) to complete the equivalence
- An objective is then to minimize X
- Seems based on an impoverished view of how local goals compose when components interact

Eliminate Unanticipated Behaviors and Interactions

- Behaviors and interactions due to superfluous functionality
  - e.g., use of a COTS component where only a subset of its capabilities is required
  - Or functions with many options where only some required

  These can be eliminated by wrapping or partial evaluation

  Being explored in the previrtualization project

- Interactions that use unintended pathways
  - E.g., A writes into B's memory
  - Or tramples on its bus transmissions
  - Or monopolizes the CPU

  These can be eliminated by strong partitioning of resources

  But we remain vulnerable to pathways through the plant
  (e.g., Concorde's tires and tanks

Control Unanticipated Behaviors and Interactions

- Unanticipated behaviors on intended interaction pathways
    - e.g., unclean failures
    - Local malfunctions

  These can be controlled by strong monitoring
    - Monitor component behavior against system requirements; shutdown on failure
    - Monitor assumptions; treat source component (or self?) as failed when violated
    - Use interface automata to monitor interactions
    - Use inline reference monitors (IRMs) to monitor security

Engineer for Resilience

- Our diagnosis is very similar to Perrow's Normal Accidents
- In his terms, we aim to reduce interactive complexity and tight coupling
- One way to do both is to increase the autonomy of components
  - i.e., they function as goal-directed agents
  - e.g., substitute runtime synthesis for design-time analysis
    (both use formal methods, but in different ways)
- But then may be more difficult to design the overall system
  - Actions of intelligent components frustrate system goals
  - e.g., pilot actions on AF 447
- Overall system should become adaptive or autonomic
  Using AI and machine learning

Summary

- Reductionist approaches to system design and understanding may no longer be appropriate
  - Systems are built from incompletely understood components, and other systems
  - System goals far removed from component functions
- Widespread emergent misbehavior seems inevitable
  - In some cases, can attempt to reduce emergence and restore validity of reductionism
  - In other cases, should embrace emergence and aim for adaptation and resilience
- In no cases will it be business as usual
- Datum: safety critical code size in aircraft and spacecraft doubles every two years (Holzmann)

Crazy Idea

- We'd like to compose system-level properties from local ones
- We actually know how to do this in the small
  - The last 20 years of formal methods
- But it doesn't scale
  - Systems, properties are too big
  - Too much other stuff: harbingers of emergent misbehavior
  - Especially for system-level properties like safety & security
- So build/verify/synthesize and use/assume the defenses I described against emergent misbehavior
- To create an environment in which local properties may safely compose (well, reasonably safely)
- Composability (PPP), Compositionality, Monotonicity
- Then focus on the automated verification/synthesis of local components, their assurance, and their composition
- Assurance case rests on these two verified/synthesized pillars