Vinton G. Cerf

DOI:10.1145/2398356.2398358

What's a Robot?

I am a big science fiction fan and robots have played a major role in some of my favorite speculative universes. The prototypical robot story came in the form of a play by

Karel Čapek called “R.U.R.” that stood for “Rossum’s Universal Robots.” Written in the 1920s, it envisaged android-like robots that were sentient and were created to serve humans. “Robot” came from the Russian word “работать” (“rabotat,” which means “work”). Needless to say, the story does not come out well for the humans. In a more benign and very complex scenario, Isaac Asimov created a universe in which robots with “positronic” brains serve humans and are barred by the Three Laws of Robotics from harming humans:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.

3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

A “zeroth” law emerges later:

0. A robot may not harm humanity, or, by inaction, allow humanity to come to harm.

In most formulations, robots have the ability to manipulate and affect the real world. Examples include robots that assemble cars (or at least parts of them). Less facile robots might be devices that fill cans with food or bottles with liquid and then close them up. The most primitive robots might not normally even be considered robots in normal parlance. One example is a temperature control for a home heating system that relies on a piece of bi-metal material that

expands differentially causing a circuit to be closed or opened depending on the ambient temperature.

I would like to posit, however, that the notion of robot could usefully be expanded to include programs that perform functions, ingest input and produce output that has a perceptible effect. A weak notion along these lines might be simulations in which the real world remains unaffected. A more compelling example might be high-frequency stock trading systems whose actions have very real-world consequences in the financial sector. While nothing physical happens, real-world accounts are impacted and, in some cases, serious consequences emerge if the programs go out of control leading to rapid market excursions. Some market meltdowns have been attributed to large numbers of high-frequency trading programs all reacting in similar ways to inputs leading to rapid upward or downward motion of the stock market.

Following this line of reasoning, one might conclude that we should treat as robots any programs that can have real-world, if not physical, effect. I am not quite sure where I am heading with this except to suggest that those of us who live in and participate in creation of software-based “universes” might wisely give thought to the potential impact that our software might have on the real world. Establishing a sense of professional responsibility in the computing community might lead to increased safety and reliability of software products

and services. This is not to suggest that today’s programmers are somehow irresponsible but I suspect that we are not uniformly cognizant of the side effects of great dependence on software products and services that seems to increase daily.

A common theme I hear in many conversations is concern for the fragility or brittleness of our networked- and software-driven world. We rely deeply on software-based infrastructure and when it fails to function, there can be serious side effects. Like most infrastructure, we tend not to think about it at all until it does not work or is not available. Most of us do not lie awake worried that the power will go out (but, we do rely on some people who *do* worry about these things). When the power *does* go out, we suddenly become aware of the finiteness of battery power or the huge role that electricity plays in our daily lives. Mobile phones went out during Hurricane Sandy because the cell towers and base stations ran out of power either because of battery failure or because the back-up generators could not be supplied with fuel or could not run because they were underwater.

I believe it would be a contribution to our society to encourage deeper thinking about what we in the computing world produce, the tools we use to produce them, the resilience and reliability that these products exhibit and the risks that they may introduce. For decades now, Peter Neumann has labored in this space, documenting and researching the nature of risk and how it manifests in the software world. We would all do well to emulate his lead and to think whether it is possible that the three or four laws of robotics might motivate our own aspirations as creators in the endless universe of software and communications.

Vinton G. Cerf, ACM PRESIDENT