Juris Hartmanis (1928-2022):[1] Understanding Time, Space, and Human Creativity

Lane A. Hemaspaandra

Juris Hartmanis's science was artistry. A great sculptor can look at a block of marble and see in it, and bring forth from it, the finished sculpture. Juris could approach a new or unformed area or a young topic and intuit the framework, structure, and goals (and typically many beautiful results) that would make clear the area's lasting importance. He did that for computational complexity, for structural complexity theory, for the potential isomorphism of NP-complete sets, for sparse sets, for upward separation, for independence theory as it might apply to questions like P versus NP, and for so many other topics. The only reason not to call him a sculptor would be if one rather would call him a magician, since his abilities seemed essentially magic—one-of-a-kind in their strength and in his depth of insight. After all, focusing on the first two items in the earlier list, jointly with Dick Stearns he is the parent of computational complexity (for which they won the Turing Award), and he is also a parent of the structural approach to computational complexity. I asked Giovanni Pighizzini, a descriptional complexity expert, about Juris and that area, and he replied, "the techniques he proposed in Structural Complexity have been applied in Descriptional Complexity and they are fundamental in the study of the phenomenon of nonrecursive trade-offs."

Juris was a tremendous raconteur, and he had a wonderful sense of dramatic line. This was reflected in his approach to research. I had the tremendous treat and great fortune to be one of his thesis advisees, and he always told me that a paper should "tell a good story." To Juris, a paper should be about, or even itself sometimes be the initial framing of, a truly interesting and important question, issue, and challenge. Then the paper should (if a paper of first impression on the issue) frame the challenge not in a handwaving way, but with a simple formalism that captures the challenge unambiguously. And then the paper should establish results (typically, in theoretical computer science, should prove theorems) that give interesting insights into the issue. He loved

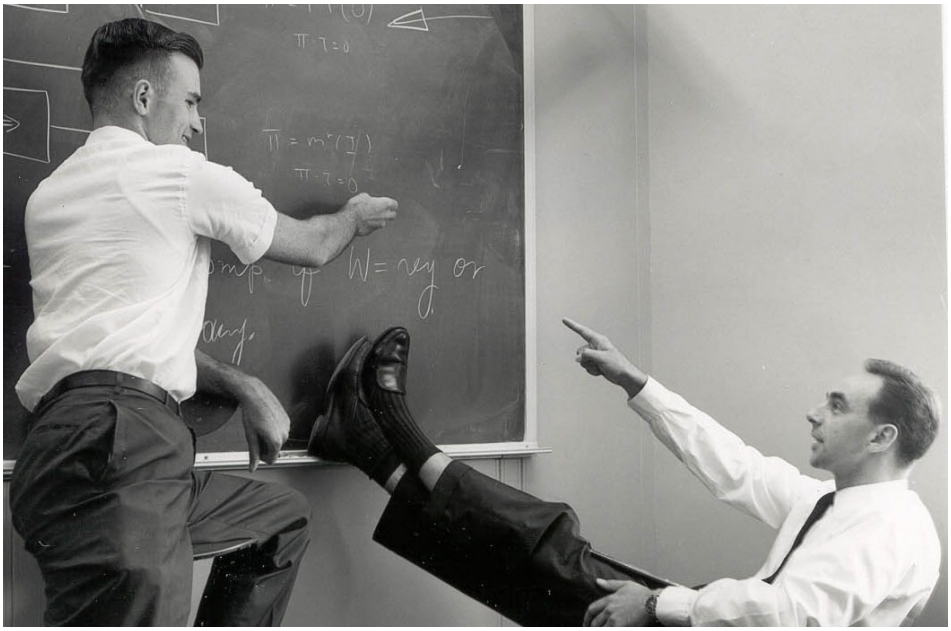
[1] Here are two places to learn about Juris's life, career, and views directly from the best source, Juris himself:

- The 2009 interview with William Aspray, a part of the ACM oral history project: `https://dl.acm.org/doi/10.1145/1141880.1775727`. A condensed version of the interview was reprinted in CACM in 2015: `https://cacm.acm.org/magazines/2015/4/184690-an-interview-with-juris-hartmanis/fulltext`. The following quotation from the interview crisply conveys the ongoing importance, to every computer scientist, of computational complexity theory: "Almost all computer science problems involve algorithms and therefore computational complexity problems. For example, computational complexity theory has put cryptography on a different base. I think every computer scientist should have some intuitive grasp of computational complexity."
- The 2018 video-interview with David Gries: `https://youtu.be/pprmQ-2Ai2Q`. See also Professor Gries's fond, informative web presentation that captures his talk at Juris's 2001 retirement ceremony: `https://www.cs.cornell.edu/gries/banquets/JHretireparty/gries.html`.

the hunt for results (and his intuition as to what theorems might hold, and what proof approaches might land them, was inhumanly good). But he also loved it when his papers captured the interest and imagination of others, and led them to join in the hunt; he knew that as a field, we are all on the same team, and the goal is deepened understanding. As to proving the theorems, these days some people almost seem to value technical difficulty of proofs for its own sake. Juris had enormous technical strength to bring to bear when needed. But for a given result that he was trying to prove, he'd take more joy in a simple, clear proof than a long, difficult one, and he would be particularly thrilled if a proof itself discovered and exploited some new structural insight. His paths and quality of thought were simply *beyond*, along many dimensions; and yet I think those of us who were his students learned tremendously much from watching him work, working with him, and trying within our own limitations to internalize his approach to research. And thanks to what one saw from and learned from him, thanks to his remarkably individualized support, feedback, challenges, and guidance, and thanks to his wildly generous confidence in and encouragement of those he worked with, one found that one could go beyond what one had believed one's limitations were. Juris truly transformed and elevated the lives and careers of those he worked with.

Juris's service contributions to the infrastructure of computational complexity, theoretical computer science, and computer science were giant, such as helping found the yearly complexity conference, heading NSF's CISE directorate, being one of the founding editors of the influential Springer LNCS series, chairing a key report on funding of theoretical computer science research, serving as a BEATCS complexity columnist, and being the founding department chair (selected in part by Professor Nerode, of the Myhill–Nerode Theorem!) of Cornell's Computer Science Department.

I'm writing this a few days after Juris's passing, and already there are many tributes to his work, as well there should be, and surely many more will be available before you read this. Most of those I've seen focus, naturally and deservedly, on Juris's most well-known contributions, including his Turing-Award-winning work with Dick Stearns that framed the meaning (within computation) of time and space. So let me here give one example of how Juris's artistry lit up even his less-known work, and then one example of how his insight framed the P vs. NP problem in a fascinating way, namely, as a window into the issue of human creativity vs. machine abilities):



Dick Stearns (left) and Juris Hartmanis (right).

- Let us consider Juris's 1983/1985 STOC/I&C paper with (two of his thesis advisees!) Immerman and Sewelson (yes, Juris's work was so well-known that one can point to this wonderful STOC/I&C paper and call it, relative to some of his other work, less known), whose main title is "Sparse Sets in NP − P." To briefly set the stage, recall that shortly before that, Juris's student Steve Mahaney, in his FOCS/JCSS paper "Sparse Complete Sets for NP: Solution of a Conjecture of Berman and Hartmanis," had proven that there are sparse NP-complete sets if and only if P = NP. However, that left open the question of what characterized whether NP − P contains *any* sparse sets. (From earlier work, it was known that E ≠ NE characterizes whether there are tally sets in NP − P. But that proof only seems to work for types, such as the tally sets, where the sets are so-called P-capturable: For each set in the class, there is some "P-printable set" (a lovely, important notion that is due to a 1984 Hartmanis and Yesha TCS paper, but I won't define the notion here) of which the set is a subset. Sparse sets don't seem to have that property; they have few strings per length, but those could potentially be almost anywhere among the strings the length. There seemed to be no tool to attack this issue. So Juris and his coauthors created a tool! In particular...) Juris and his coauthors show how sparse NP sets can have their strings' information reencoded into friendly bite-sized morsels, namely $\mathcal{O}(\log n)$-sized strings, uniformly forming part of an NE set. And using that NE set, a certain type of machine can find and use the morsels to recover all the strings at a given length of the original set. Beyond that, it turns out that since there are only a polynomial number of log-length strings, one can if one wishes even do the reconstruction after making a single *parallel* round of $\mathcal{O}(\log n)$-length queries to the set of morsels. The details aren't important for our purpose here. The point is that the Hartmanis–Immerman–Sewelson paper thought out of the box in a truly creative, novel way. The proof isn't hard (though that does not mean it was not amazing for the authors to discover; no one else saw this approach); but the proof is really, really lovely.

  And there is an additional creative turn, alluded to above, within their approach. The authors in effect are tackling the task, which seems unclear how to do in the situation, of testing whether a given string x is in a certain set, by showing, using the morsels approach, how one can find *all* strings of length $|x|$ that are in the set. That is, they solve a harder task, and by doing so get their "easier" goal as a side effect.

- I'm not sure if I heard this from Juris as part of his legendary graduate complexity course (that back when I was a CS Ph.D. student at Cornell was taken by basically all CS Ph.D. students) or in conversation with Juris. (The closest text source I can find is Juris's abovementioned ICALP/TCS paper with Yacob Yesha, "Computation Times of NP Sets of Different Densities," a paper that is mostly known for introducing the notion of P-printability. But that paper's Section 3, entitled "The Computational Complexity of Mathematics," and its comments (along with some easy observations one can make about what it is discussing) basically implicitly cover what I'm about to mention. The actual core of the section is frying different, more difficult fish. But the "human creativity" vs. machine-automated-proof view of P versus NP, which I learned from Juris, is something that has stayed with me for four decades, and that I often mention to students in motivating the P vs. NP issue.)

  One way of (loosely—I'm in fact playing a bit fast and loose as to proof frameworks) interpreting/using the structures Juris and Yacob set up in the paper-section mentioned above, along with that paper's NP-completeness comment about its language "L_1," is this: P versus

> NP controls the issue of whether human creativity, as to proving things, is such a joke that we might as well just prove things by computers, since they are (if P = NP) provably perfect at such proof tasks (give or take a polynomial). In particular, if P = NP, then (within an appropriately axiomatized formal system, which let us take to be fixed) there clearly is a program that, when given as input any theorem that can be proven in the system, will find the shortest possible proof in time polynomial in the length of that shortest proof within the system[2]... so, departments of mathematics, pack your bags... or at least focus your human creativity just on what is left for you to do, namely, framing interesting theorems, and then letting HAL prove them for you. But, on the other hand, if P ≠ NP, there can exist no such program as mentioned above... so, you're back in business, mathematics departments!
>
> As a cherry on top of that, that same Hartmanis–Yesha paper then also discusses not traditional proofs, but proof presentations (basically, proofs on a polynomial-sized "blackboard," but one can choose to use an eraser to remove some proofs once results have been established via them, and one can choose to remove some results once other results have been established by them, and so on; so at the end of the process, some of the proof may no longer be up on the "blackboard," yet the (polynomial-time) watcher/verifier nonetheless is convinced by the process that the theorem is, well, a theorem). And the authors' framework makes implicitly clear that that process is connected to the P versus PSPACE question, and draws an explicit and interesting connection to the NP versus PSPACE question.

Juris's kindness and generosity was amazing. After my oral area-exam at Cornell, on which I suspect I barely survived and which I knew could not possibly have left any of the committee with anything upbeat to say, Juris quietly said to me on his way out of the room, "I saw something there." That one sentence, coming from him, made a tremendous difference. And for decades after I had graduated, when in need of wise career advice in difficult situations, he was the person I went to. He always made time, and he always had—essentially instantly and apparently effortlessly—exactly the right answer, often something I had not even had on my radar; he could think out of the box not just in theoretical research, but also regarding the politics and logistics of academia. His generosity and influence went far beyond his own students, and in the days since his passing, so many people I have been speaking with have mentioned how much he did for them or how much his work influenced them. He sincerely cared about helping the next generations of theoretical computer scientists.

I'd say it is hard to imagine the field of computational complexity without Juris. But in fact, more to the point, the field would not *exist* in its current form and strength without Juris. The depth of loss in his passing cannot be captured in words, but his frameworks and contributions will continue to guide the field's way for as long as complexity theory lives: forever.

Juris said each paper should be about telling a good story. His life was and his papers told great stories. He left his mark on the world; and it is glorious.

[2] If the input is a claim that cannot be proven within the system, the machine would run forever. That is why in the paper they avoid the issue by inputting both a claim and a natural number, and ask if there is a proof within the system whose length is at most that natural number.