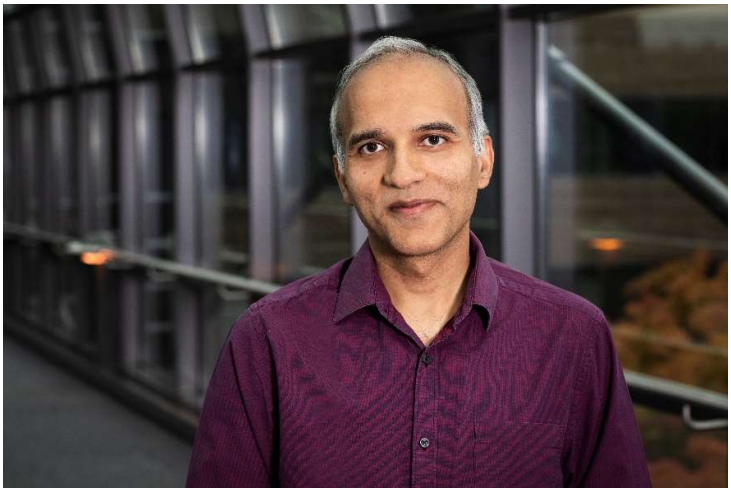
Shaping the future with: Sumit Gulwani

In our series "Shaping the future with" we introduce you to interesting personalities who are working on future technologies. This time with Sumit Gulwani from Microsoft Research. He explains why the computers of the future will understand us humans better.

Sumit Gulwani founded and currently leads the PROSE research and development team at Microsoft Research, which develops techniques for *program synthesis* and incorporates them into various mass-market products. He has made his mark on his field of research with his work. Program synthesis is the ability to automatically generate intended computer programs from natural forms of intent specification such as input-output examples, natural language, or sometimes even predictively from the static or temporal coding context. Among other things, he is the inventor of the popular Flash Fill feature in Microsoft Excel, which automatically inserts data into an Excel column as soon as it recognizes a pattern as to how it relates to other columns.



We spoke with Sumit about his groundbreaking work, which was recently awarded the Max Planck-Humboldt Medal 2021. We also talked about how program synthesis makes life easier for all of us and the impact it can have on education.

You were recently awarded the Max Planck Humboldt Medal 2021 for your achievements in the field of program synthesis. How would you explain your field of research to a child?

Sumit: "The techniques of program synthesis allow users to express a process in the most natural way, for instance, through **examples** or through a **description in natural language**, without having to program with a strict and sometimes difficult to understand syntax.

The other day I was explaining the two techniques to my son Sumay, who needs his own passwords for many digital applications at his primary school. The teacher communicated these passwords by emailing an example of how to derive them from a student name and ID, “if your name is: Alex Zorn and ID is 12345678, then your password is a-ZORN#1234.”

I showed Sumay that the Flash Fill function in Excel, which I developed in 2010, is smart enough to solve such analogy tasks. And it does so by generating a program from very few representative input-output examples, which is then run with new inputs to get the desired outputs. This is called **program synthesis using examples.** This video shows how it works in Excel:

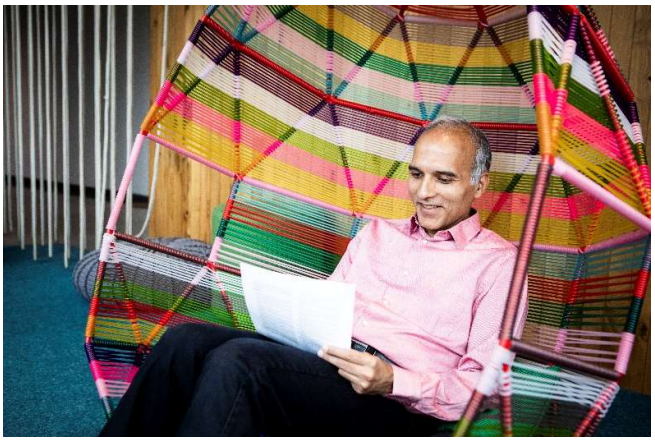
https://www.instagram.com/reel/CLaPESvFnhd/?utm_source=ig_web_copy_link

While examples can sometimes be an easy way to communicate the intent of a computation, other tasks are best described in **natural language.** This can be well explained with a problem from Sumay's math curriculum:

Sumay has a large list with the number of stars that students from various classes have received. Now he has to determine "which class received the most stars". Natural-language program synthesis techniques can translate such literal descriptions into executable programs over tables (such as SQL queries or spreadsheet formulas). This allows Sumay to quickly determine the result without a manual calculation.

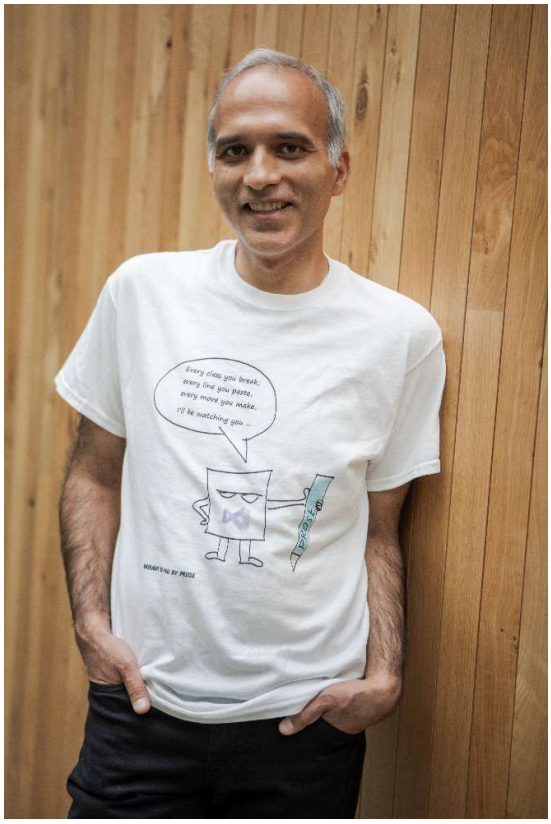
These techniques have shaped automatic programming. They have shown that programs can be efficiently generated from imprecise specifications and that program synthesis can not only help computer professionals with complicated algorithms but also computer users with relatively simple programs or program snippets. Recently, I learned that Flash Fill is part of several middle school computing textbooks in India."

To what extent do you think improvements in program synthesis can empower individuals and organizations to shape our future?



Sumit: "Computer technology has permeated all aspects of our lives. However, 99 percent of computer users cannot program and therefore struggle with tedious, repetitive tasks. Programming as it exists today is an artificial barrier to effective and creative use of computers.

For example, Sumay's teacher could have sent each student their own password if she could have created a program to do so. The techniques of **program synthesis by demonstration** can watch Sumay's teacher send a couple of these personalized emails and then offer to send them to the rest of the children in the class.



Program synthesis can also support computer professionals such as data scientists or developers.

Data scientists spend a lot of time manipulating data, cleaning it, transforming it, and analyzing it. This data manipulation can be accelerated by program synthesis. For instance, the data connectors in Microsoft's PowerQuery engine synthesize parsing scripts from examples to extract tabular data from semi-structured documents like custom text files or webpages. Recently, we were pleased to see them used to create dashboards for various Covid 19 datasets.

Developers spend most of their time writing templatized code or in code maintenance tasks such as refactoring code, merge conflict resolution, and doing repetitive edits to fix bugs. Program synthesis can help with these tasks, acting as an AI-powered pair programmer to make suggestions as you work. Our IntelliCode Suggestions feature in Visual Studio is one such form factor for this use case.

The latest addition to the bunch, GitHub's CoPilot, which is based on the pre-trained model Codex developed by OpenAI, has captured the imagination of the world. It predicts code fragments - using natural language comments and previous code context. We are on the cusp of a disruption that AI-powered pair programming will bring. "

Lately, you've been using program synthesis tools for education. How can program synthesis support programming education, learning feedback, grades, and more? Where do you see possible future applications in education?

Sumit: "A single teacher attends to many students in a classroom, so individuals cannot receive personalized feedback. Online courses with many participants exacerbate this problem.

An example: If one learns programming today, the error messages of the compilers (programs that generate machine-readable code) are often confusing. Moreover, it is not enough to present the students with any sample solution when they fail to solve a problem. Because what they really need to know is how *their* solution can be converted into a correct solution. Program synthesis techniques can be used to convert students' solution into a nearest correct solution (or the next logical step). This can form the basis not only for fairer grading, but also for personalised feedback and guidance to students, providing them with a better learning experience.

I would like to take on a grand challenge: to develop an AI bot that is almost indistinguishable from a human teacher in terms of feedback and cues for learners. By collecting learning journeys of a large number of learners from around the world, we could develop appropriate ML models. These when combined with appropriate program analysis and synthesis techniques can lead to an AI bot that can provide instant personalized feedback to learners. Such technology can even facilitate peer-to-peer learning by connecting learners and enabling pedagogical experiments. This gives instructors more time to interact with learners."

You once said: "Programming is the science of talking to computers". How do you get people excited about learning this language?

Sumit: "The deepest level of learning takes place when you create something that is meaningful to you. With this understanding, I found the right moment last year to introduce programming to my son Sumay.

He was solving a 2nd grade math problem: find all tuples (a,b,c,d) where a*b=c*d and a,b,c,d are distinct digits. After he had painstakingly listed the 40 solutions, I joked with him: 'Do you know that a computer could do this for you? All you have to do is tell it what to do. ' His eyes lit up, and he enthusiastically tried out his first computer program.

In addition to a motivating curriculum, we also need to make computers smarter so that we humans can talk to them more easily.

When Sumay was excited to write his first computer program to solve his math problem, I pulled up an online Python editor. Then we started writing the program for the problem. And Sumay held his breath as he waited on the screen for the solution. Mind you, I had never written a Python program before. But with my 20 years of programming experience, I was confident that I could figure it out using error messages. But no, I failed. I tried many different syntaxes, but the error messages were unhelpful and confusing. Sumay got frustrated and asked, 'Dad, are you sure you know how to

program? ' Eventually, I had to look up the documentation to get it to work. You know, most people would have been able to read my intent from the inaccurately written program. And if computers can understand such sloppy programs or pseudocodes, it may not only change programming education, but it will also seed the next programming revolution that will democratize programming for all and take programming closer to human conversation.”