No. 18-1199

In the Supreme Court of the United States

INVESTPIC, LLC,
Petitioner,

v.

SAP AMERICA, INC.,
Respondent.

On Petition for Writ of Certiorari to the United States Court of Appeals for the Federal Circuit

BRIEF OF DR. PHILIP NECHES AS *AMICUS CURIAE* IN SUPPORT OF PETITIONER

ROBERT P. GREENSPOON
COUNSEL OF RECORD
FLACHSBART & GREENSPOON LLC
333 N. MICHIGAN AVE., STE. 2700
CHICAGO, IL 60601-3901
(312) 551-9500
rpg@fg-law.com

APRIL 15, 2019

TABLE OF CONTENTS

TABLE OF AUTHORITIES

N/A

INTEREST OF *AMICUS CURIAE*[1]

Dr. Philip M. Neches, Ph.D., submits this brief as *amicus curiae*.

In 2012, the National Academy of Engineering elected Dr. Philip Neches to its rolls "for the architecture and software of parallel database appliances," recognizing his work as Founder and Chief Technical Officer of Teradata Corporation. The world's largest financial institutions, retailers, airlines, internet application providers, government agencies, and the like, use Teradata systems. The Teradata architecture developed from Dr. Neches' doctoral thesis at the California Institute of Technology, where he now serves on the Board of Trustees. Dr. Neches served as Chief Technical Officer for NCR Corporation and the Multimedia Products and Services Group of AT&T. In the past 20 years, Dr. Neches has advised many IT industry companies, large and small, across a range of technologies and markets. Dr. Neches believes his unique experience in parallel computing and academia can shed light on the question presented beyond what the parties can do.

---

[1] No counsel for a party authored this brief in whole or in part. No person or entity other than *Amicus* or its counsel made a monetary contribution to the preparation or submission of this brief. Consent for filing this *amicus* brief has been obtained from all parties.

Though this brief is authored with the aid of counsel, Dr. Neches wishes to present his argument in his own voice.

SUMMARY OF ARGUMENT

I implore the Supreme Court to grant certiorari in this case, because it will answer the fundamental question of the extent to which, if at all, patent protection will be available for innovations in software, data, and computing. At present, nearly all data and information technology companies, large and small, rely on both hardware and software innovations that demand protection as critical intellectual property. Likewise, industry relies on patenting to supply an updated, constantly refreshed "library" of the state of the art in the field. However, some new interpretations of patent law afford only one portion of those inventions patent protection (*i.e.*, those that are physical). Inventions in the non-physical realm are just as crucial to secure and protect as those in the physical realm. Industry needs full incentives for all industry participants to publish their new results, which until recently the patent system did admirably.

Many, if not most, of the innovations in modern computing occur in the digital realm, rather than involving hardware itself. The future health of the American computing industry may hinge on the degree of intellectual property protections granted for non-physical inventions. Our intellectual

property system must fully credit non-physical inventions, or we risk handicapping innovators, developers, and inventors in global competition. We must incentivize those willing to spend the developmental capital necessary to spur forward. We must also incentivize prompt publication of results, even among for-profit enterprises. Recent narrowing interpretations of patent eligibility should be rolled back, lest they deprive the computing industry of needed tools to attract capital, build new jobs, and protect risk-based enterprises. This will also re-incentivize technological progress in non-physical fields, and maximize dissemination of new knowledge through patenting.

ARGUMENT

Every young coder or aspiring programmer strives to better their skills and create something new, something that brings efficiency and connection to people across the globe. However, without the potential for patent protection, we risk disallowing any but the most massive and established companies from maximizing the utility of innovative creation in the computing field.

To exemplify the type of laborious innovation that must be protected in the computing field, I will lay out the technology behind the application for Patent No. 6,349,291 ("'291") and the developmental track that provided its creation. As discussed below, I believe that if the '291 patent inventors knew at the time that the current interpretation of patent eligibility would have resulted in a worthless patent, it is not likely they would have gone through the

process in the first place. The world would have missed out on full dissemination and publication of those important groundbreaking ideas.

Parallel computing is a very complex and difficult art for many reasons. However, it is not an abstract creation. The number of programmers who can write successful parallel applications is quite small. Programmers must envision many simultaneous computations, and design the program so that the result is correct. This means that different parts of the computation cannot work on the same datum at the same time, just as automobiles on a highway cannot be allowed to collide with each other.

In my experience, two relatively small classes of people write almost all of the successful parallel applications. The first class consists of researchers (graduate students, postdocs, and occasionally extremely bright undergraduates) at the most elite research institutions. The second class consists of systems programmers who work on the internals of operating systems and are versed in re-entrant code. Code is deemed "re-entrant" if it can be interrupted in the middle of its execution and then safely called again before its previous execution completes.

In both re-entrant and parallel coding, the programmer must deal with many "instances" of the same code working on different data. The style of concurrent programming required for parallel applications is taught today in advanced computer science classes, but is still a relatively unexplored part of the curriculum. Even today, few

programmers actually practice these disciplines because of the extraordinary intellectual complexity. Most applications, whether in the commercial or scientific regimes, can be designed and coded for sequential computing environments, and almost all programmers only practice that much simpler discipline.

The designer of a parallel program is thus challenged to manage the flow of data and computation to achieve workload balance. This cannot be done without considering in detail the architecture of the target hardware. What will work well on one target machine will often produce disastrously poor performance on another target machine. Unless the programmer is lucky enough to have a problem that is a near-exact analog for a previous problem, the job of "partitioning" (assigning data and parts of the computation to particular processing elements) must be done anew for each problem. And since the code that orchestrates the movement of data composes the overwhelming bulk of the code of a parallel application, almost every new parallel application begs the designers and coders to start from scratch. People who develop many parallel applications develop their intuition about what may work, but they still have to begin at the beginning to figure out what will work. Needless to say, development in parallel computing is hugely expensive in both time and resources, and a lack of programming expertise and high startup costs curtail the amount of risk-based investment available to the field.

To this day, there is no successful standard language for parallel programming. There are many sets of language extensions available, each of which corresponds to a very particular execution environment, and each of which requires the application designer to thoroughly understand how that particular combination of language, environment, and hardware will deal with the concurrent execution of different parts of the same parallel program.

As just explained, parallel computing orchestrates the movement of data and processing among the hardware resources of the target system: getting the right instructions and the right data to the right place at the right time—all while avoiding collisions, errors, and delays. In practice, very little of the code of a parallel application implements the underlying algorithm. Almost all the design, code, debug, test, validation, and maintenance effort goes into managing the flow of data and execution.

Based on my experience, the '291 patent gives insight into the hardest problems of parallel system design (namely, partitioning of the data and execution flow) such that a practitioner with ordinary skill could carry out the rest of development of such a system. An ordinarily-skilled practitioner would have difficulty, even today, in coming up with the partitioning and execution flow. At the time the '291 patent was filed, that level of insight was even rarer.

I work daily with innovative companies, mostly start-ups, that embody their creative ideas

mostly in software. They find new ways of attacking important problems with major economic and social consequences. Often, they apply techniques known and proven in one field to another field. But frequently, they come up with entirely new techniques and/or significant improvements to existing techniques that are indeed novel, innovative, and non-obvious.

These companies, which Alexander Hamilton dubbed "infant industries," navigate difficult and dangerous waters on the path to commercial success. Larger companies may copy their innovations, whether by blissful ignorance or conscious malice, depriving the innovators of the rewards of their effort and creativity.

This is exactly the circumstance the Framers of the Constitution had in mind when they mandated the infant United States Government to create a patent system. My company benefited from the patent system, and I advise my new companies to avail themselves of the same protection.

I must acknowledge that the Patent Office granted a number of "software" patents since the '291 patent that, to me, fail the "obviousness" test and should not have been allowed. This class of, in my view, wrongly issued patents stymies innovation by limiting the creative options of other programmers and enterprises. I support efforts to clean up this situation. However, I think the Federal Circuit over-reached with the "physical realm" test that throws out the baby with the bath water.

Innovations that create accessibility for legions of new programmers, such as '291 patent, must be afforded patent protection if we are to spur further development in computing generally and in parallel processing specifically. Not only are inventions such as '291 patent costly to create, they represent crucial frameworks for technical advantage and future development. When developments like the '291 patent occur within for-profit enterprises, the patent system is the only reliable spur to publication of new results, and dissemination of new knowledge.

Patenting should be a seed for inquiry and investigation, and this seed should not be arbitrarily withheld from non-physical technologies. Our intellectual property system must reward lynchpin non-physical creations through the exclusivity and financial incentives associated with patent protection, and spur their technological dissemination to the maximum extent for the good of all.

CONCLUSION

For the foregoing reasons, *amicus* requests that this Court grant certiorari and reject the innovation-chilling "physical realm" test of the Federal Circuit.

Respectfully submitted,

ROBERT P. GREENSPOON
COUNSEL OF RECORD
FLACHSBART & GREENSPOON LLC

333 N. MICHIGAN AVE., STE. 2700
CHICAGO, IL 60601-3901
(312) 551-9500
rpg@fg-law.com

April 15, 2019