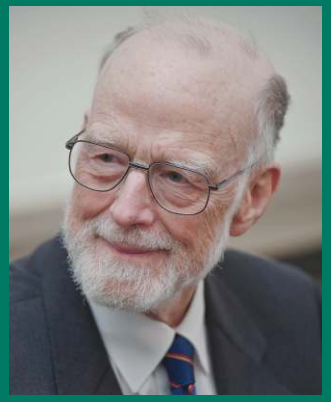


Sir Charles Antony Richard Hoare (Tony Hoare)

wybitny, światowej sławy informatyk, emerytowany profesor Oksfordu.

Urodzony 11 stycznia 1934 r. w Colombo (Sri Lanka), ukończył statystykę matematyczną na Uniwersytecie w Oksfordzie. Po odbyciu służby w Królewskiej Marynarce Wojennej (1956-1958) wyjechał na staż do Uniwersytetu Moskiewskiego, gdzie pod kierunkiem Andreja Kołmogorowa kontynuował studia statystyczne i wykorzystywał je w automatycznym przekładzie języków naturalnych. W latach 1960-1968 był zatrudniony w Elliott Brothers, firmie produkującej komputery, gdzie opracował kompilator języka Algol 60 i wiele algorytmów obliczeniowych. W 1968 r. został profesorem informatyki na Queen's University w Belfaście, a dziewięć lat później powrócił na Oksford, gdzie utworzył Wydział Informatyki.

Uczony jest twórcą m.in. popularnego algorytmu Quicksort, logiki Hoare'a do weryfikacji poprawności programów, języka CSP (Communicating Sequential Processes) służącego do specyfikacji współbieżnych procesów, a także wielu konstrukcji i rozwiązań stosowanych w językach programowania. Jest autorem ponad 200 publikacji naukowych.

Za swoje osiągnięcia uczony otrzymał wszystkie najważniejsze nagrody, jakie można przyznać w dziedzinie informatyki, w tym: Turing Award, Kyoto Prize oraz Medal von Neumanna. Jest członkiem kilku prestiżowych instytucji naukowych, w tym Royal Society oraz Royal Academy of Engineering. W 2000 r. nadano mu brytyjski tytuł szlachecki.

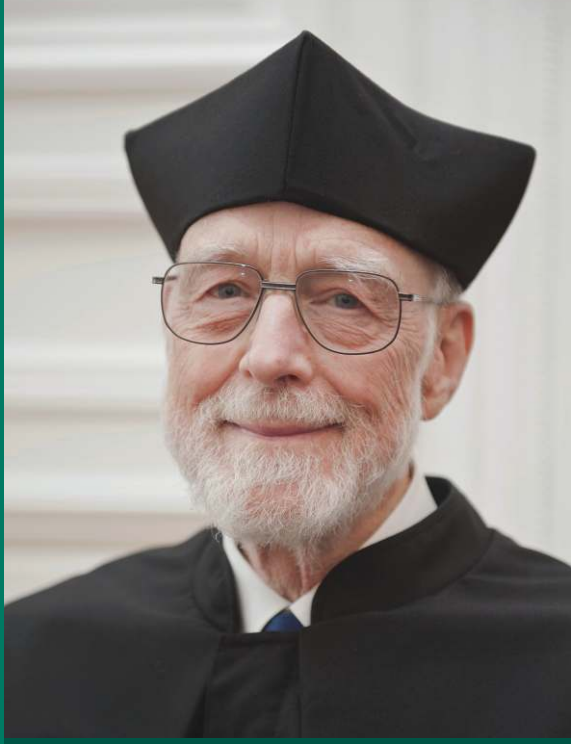
Sir Charles Antony Richard Hoare (Tony Hoare)

an eminent, world-famous computer scientist, retired professor of the University of Oxford.

Born in Colombo (Sri Lanka) on January 11th, 1934, graduated from the University of Oxford (mathematical statistics). After serving military service in the Royal Navy (1956-1958), he had an internship in Moscow State University, where under the supervision of Andrei Kolmogorov continued studies on statistics, which he used in automated translation between natural languages. Over his employment in Elliott Brothers a computer company (1960-1968) he implemented Algol 60 and developed a number of calculation algorithms. In 1968 he became the professor of computer science at Queens University in Belfast, and after nine years he returned to University of Oxford, where he was a founder of the Faculty of Computer Sciences.

Sir Charles Antony Richard Hoare developed, among others, a commonly used Quicksort algorithm, Hoare logic for verifying program correctness, CSP (Communicating Sequential Processes) language to specify the interactions of concurrent processes, as well as numerous constructions and solutions applied in programming languages. He is the author of over 200 publications.

In recognition of his achievements the scientist was granted most prestigious awards in the field of computer sciences, including Turing Award, Kyoto Prize and von Neumann Medal. He also holds the membership of several prestigious scientific institutions such as the Royal Society and Royal Academy of Engineering. In 2000 Professor Hoare was granted British Knighthood.



Sir Charles Antony Richard HOARE
doktorem *honoris causa*
Uniwersytetu Warszawskiego
Uroczystość wręczenia dyplomu

Sir Charles Antony Richard HOARE
Doctor *Honoris Causa*
of the University of Warsaw
The diploma awarding ceremony

28 listopada 2012 roku • November 28th, 2012

Sir Charles Antony Richard Hoare

doktorem *honoris causa* Uniwersytetu Warszawskiego

Uroczystość wręczenia dyplomu

Warszawa, 28 listopada 2012 roku

Opracowanie / Elaboration: Biuro Promocji UW
Redakcja i korekta / Editing and proof reading: Marzena Burczyńska
Współpraca / Cooperation:
Prof. Jan Madey, Prof. Władysław M. Turski

Zdjęcia / Photographs:
Mirosław Kaźmierczak

Tłumaczenie / Translation:
Katarzyna Kopij

ISBN 978-83-62844-27-2

Osoby zainteresowane otrzymaniem publikacji
z uroczystości wręczenia dyplomu doktora *honoris causa* UW
lub odnowienia doktoratu
prosimy o kontakt: tel. 22 55 24 032, e-mail: promocja@uw.edu.pl

To receive the publicatio on the ceremony
of handing the diploma of doctor *honoris causa* of the University of Warsaw
or of renewal of the doctorate, please contact us
phone: +48 22 55 24 032, e-mail: promocja@uw.edu.pl

Skład i druk / Design and print:
WEMA Wydawnictwo-Poligrafia Sp. z o.o.

Program uroczystości

- *Hymn państwowy*
- Otwarcie uroczystości przez JM Rektora – prof. Marcin Pałys
- Laudacja promotora – prof. Władysław M. Turski
- Wystąpienie Dziekana Wydziału Matematyki, Informatyki i Mechaniki – prof. Andrzej Tarlecki
- Wręczenie dyplomu doktora *honoris causa*
- *Gaude Mater Polonia*
- Wystąpienie Doktora *Honoris Causa* – Sir Charles Antony Richard Hoare
- Zamknięcie uroczystości przez JM Rektora
- *Gaudeamus igitur*

UCHWAŁA NR 11
SENATU UNIWERSYTETU WARSZAWSKIEGO

z dnia 17 października 2012 r.

w sprawie nadania Sir Charlesowi Antony'emu Richardowi Hoare'owi tytułu doktora *honoris causa* Uniwersytetu Warszawskiego

Na podstawie art. 62 ust. 1 pkt 9 ustawy z dnia 27 lipca 2005 r. – Prawo o szkolnictwie wyższym (Dz. U. Nr 164, poz. 1365 z późn. zm.) oraz §8 Statutu Uniwersytetu Warszawskiego (tekst jednolity: Monitor UW z 2006 r. Nr 6B, poz. 140), po zapoznaniu się z recenzjami: prof. Pawła Idziaka – Uniwersytet Jagielloński, prof. Henryka Krawczyka – Politechnika Gdańska, prof. Antoniego Mazurkiewicza – Polska Akademia Nauk, dotyczącymi dorobku Sir Charlesa Antony'ego Richarda Hoare'a – kandydata do tytułu doktora *honoris causa* Uniwersytetu Warszawskiego, Senat Uniwersytetu Warszawskiego postanawia, co następuje:

§1

Nadaje się Sir Charlesowi Antony'emu Richardowi Hoare'owi tytuł doktora *honoris causa* Uniwersytetu Warszawskiego.

§2

Uchwała wchodzi w życie z dniem podjęcia.

Przewodniczący Senatu
Uniwersytetu Warszawskiego
Rektor
dr hab. Marcin Pałys, prof. UW

Prof. dr hab. Paweł Idziak
Uniwersytet Jagielloński

Opinia dla Senatu Uniwersytetu Warszawskiego w związku z nadaniem tytułu doktora *honoris causa* Sir Charlesowi Antony'emu Richardowi Hoare'owi

Sir Charles Antony Richard Hoare jest jednym z najwybitniejszych twórców informatyki. Tworzył ją od samych podstaw. Zarówno w czasie jej burzliwych początków, jak i później, kreując idee, które legły u podstaw wytyczania wielu aktualnych do dziś kierunków jej rozwoju.

Odebrał bardzo gruntowne i wyjątkowo wszechstronne wykształcenie. Urodził się w 1934 roku w Colombo na Cejlonie (dziś Sri Lanka), lecz już czasy licealne spędził w Dragon School w Oksfordzie oraz w King's School w Canterbury. Również w Oksfordzie spędził lata studenckie głównie nad łaciną, greką i filozofią. Ta ostatnia wprowadziła Go w świat współczesnej logiki, której idee, język i metodologię wykorzystywał w swoich badaniach naukowych. Tuż po ukończeniu swych studiów w 1956 roku został powołany na dwa lata do Royal Navy. Tam opanował m.in. język rosyjski. Nim miał jednak okazję wykorzystać te nowe umiejętności, powrócił w 1958 roku do Merton College w Oksfordzie, gdzie studiował statystykę matematyczną oraz wszedł w świat programowania w języku Mercury Autocode. Jako doktorant spędził rok w Uniwersytecie Łomonosowa w Moskwie, gdzie pod kierunkiem Andrieja Kołmogorowa kontynuował swe studia statystyczne i wykorzystywał je w automatycznym przekładzie języków naturalnych. To właśnie w czasie tych badań uświadomił sobie jak dużą rolę odgrywa szybkie sortowanie i tam zaczęły kiełkować idee, które później doprowadziły Go do algorytmu *Quicksort*.

Po powrocie w 1959 roku do Wielkiej Brytanii związał się na 8 lat z firmą Elliott Brothers. Jeden z projektów, nad którym pracował zespół pod jego kierownictwem, zakładał stworzenie kompilatora, rewolucyjnego wtedy języka programowania ALGOL 60, na komputery Elliott 803.

W odróżnieniu od swoich poprzedników ALGOL 60 umożliwiał implementację rekurencji. Widząc jej siłę Hoare w pełni dopracował idee *Quicksort*. Do dziś jest to najczęściej używany algorytm sortowania. Doczekał się wielu wariantów, mutacji i ulepszeń. Trudno przecenić więc jego wagę. Samo to osiągnięcie bez trudu zapewniłoby Hoare'owi najwyższą reputację tak w świecie naukowym, jak i wśród programistów. Przykrył je jednak znacznie większymi dokonaniami. Doświadczenie jakie zdobył pracując w Elliott Brothers nad kompilatorami oraz systemami operacyjnymi, uczuliło Go na kluczową rolę języków programowania. Coś co początkowo mogło wydawać się porażką przy tworzeniu systemu operacyjnego Mark II Elliott 503 było inspiracją dla Jego późniejszych głębokich badań nad współbieżnością w wykonywaniu programów i procesów. Również w czasie swojej pracy dla Elliott Brothers powstawały idee, które legły u podstaw matematycznego i praktycznego podejścia do języków programowania i tworzenia oprogramowania wysokiej jakości.

Rola wkładu Hoare'a w metodykę projektowania języków programowania staje się widoczna po uświadomieniu sobie ogromnych kosztów ponoszonych na tworzenie oprogramowania. W praktyce okazuje się, że zbyt często oprogramowanie jest złej jakości, a część winy ponosiły (i nadal ponoszą) języki programowania i ich kompilatory. Dość powiedzieć, że roczne straty z powodu błędów oprogramowania idą w dziesiątki miliardów dolarów. Hoare wielokrotnie wskazywał na konieczność prostoty i elegancji niezbędnej do unikania tych błędów i do kontroli nad nimi. Doskonale ujął to w pracy *Hints on programming language design* z 1973 roku.

Wcześniej jednak, w roku 1968, objął katedrę Computer Science w Queen's University w Belfaście, gdzie – wolny od obowiązków programistycznych i rozczarowany powszechnie akceptowaną praktyką dokumentowania programów – rozpoczął prace nad semantyką języków programowania. Inspirowany pracami Roberta Floyda – pokazującymi jak dodanie warunków i niezmienników w opisie blokowym programu pozwala na precyzyjne matematyczne wykazanie, że program ten jest zgodny ze specyfikacją – jeszcze bardziej zmatematyzował badania semantyczne. W swej pracy *An axiomatic basis for computer programming* odszedł od schematów blokowych na korzyść systemu logicznego, znanego teraz jako logika Hoare'a, umożliwiającego czysto formalne rozumowania o programach i ich zachowaniu. To bardzo znaczący krok –

poprawność jest wyrażana i weryfikowana na etapie budowy programu, a nie jedynie poprzez kontrolę po jego ukończeniu.

W roku 1977 Hoare został profesorem w University of Oxford, gdzie powrócił do swych badań nad współbieżnością. Ich rezultatem było opracowanie zasad współpracy komunikujących się ze sobą procesów współbieżnych. Zasady te zostały ujęte w stworzonym przez Niego rachunku procesów i opisanym najpierw w pracy *Communicating Sequential Processes,* a następnie rozwinięte w książce o tym samym tytule. Związany z tym rachunkiem język dopuszcza interakcje między procesami jedynie we wcześniej precyzyjnie zaplanowanych sytuacjach. Daje to pełną kontrolę nad budową strukturalnych programów współbieżnych. Badania nad tym językiem rozwijały się bardzo szeroko. Szybko doczekały się zastosowań w praktyce inżynierskiej. Język ten, będąc początkowo narzędziem programistycznym, doczekał się też innego wcielenia – poprzez swoje rozwinięcie w język *occam* wpłynął na architekturę transputerów.

Nie sposób wyliczyć wszystkich dokonań Sir Charlesa Antony'ego Richarda Hoare'a. Nawet po przejściu na emeryturę w 1999 roku pozostaje On bardzo aktywny. Dość wspomnieć, że spośród jego około 200 prac naukowych ponad 60 ukazało się po roku 1999.

Za swoje osiągnięcia był honorowany najwyższymi nagrodami, jakie informatyk może otrzymać. Już w 1980 roku otrzymał Medal i Nagrodę Turinga, powszechnie uznawaną za odpowiednik Nagrody Nobla w informatyce. W roku 2005 został uhonorowany równie prestiżową Kyoto Prize, a w roku 2011 otrzymał Medal von Neumanna.

W roku 2000 Królowa nadała mu szlachectwo brytyjskie.

Od roku 1982 jest członkiem Royal Society, a od 2005 Royal Academy of Engineering. Jest doktorem *honoris causa* kilku uniwersytetów.

Wpływ Sir Charlesa Antony'ego Richarda Hoare'a na rozwój informatyki jest olbrzymi. Nie tylko poprzez wspomniane dokonania naukowe – teoretyczne i praktyczne, lecz również poprzez kształcenie kolejnych pokoleń naukowców. Wypromował 23 doktorów, z których większość kształciła kolejnych. Do Jego potomków naukowych zalicza się ponad 230 osób pracujących na 5 kontynentach.

Wpłynął również na kierunki badań w Polsce, w szczególności w Uniwersytecie Warszawskim. Przebywali u Niego na stażach pracowni-

cy Instytutu Informatyki tegoż Uniwersytetu. Dało to tak im, jak i później ich uczniom, bardzo silny impuls do rozwoju badań nad systemami operacyjnymi i procesami współbieżnymi. Również wiele badań prowadzonych w Uniwersytecie Warszawskim w zakresie konstrukcji algorytmów, metod programowania, specyfikacji i weryfikacji programów czy logiki Hoare'a czerpie swe inspiracje z tych kontaktów naukowych.

Niesłychanie wysoka pozycja naukowa Sir Charlesa Antony'ego Richarda Hoare'a i wpływ jaki wywarł na rozwój informatyki w Uniwersytecie Warszawskim w pełni przekonują mnie, że zasługuje On na zaszczyt dołączenia do wybitnego grona doktorów honorowych największego polskiego uniwersytetu – Uniwersytetu Warszawskiego.

Prof. dr hab. inż. Henryk Krawczyk
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

Opinia dla Senatu Uniwersytetu Warszawskiego w związku z nadaniem tytułu doktora *honoris causa* Sir Charlesowi Antony'emu Richardowi Hoare'owi

Sir Charles Antony Richard Hoare urodził się w Colombo (obecnie Sri Lanka) w rodzinie wysokiego urzędnika brytyjskiego. Ukończył Bachelor's degree na uniwersytecie w Oksfordzie (Merton College) w 1956 roku. Tam też przez rok studiował statystykę, a następnie odbył szkolenie wojskowe w Royal Navy. Kolejne dwa lata studiował w ZSRR na uniwersytecie w Moskwie w szkole Kołmogorowa. W roku 1960 rozpoczął pracę w firmie produkującej komputery (Elliott Brothers Ltd.), gdzie zaimplementował translator języka Algol 60 i opracował wiele algorytmów obliczeniowych. W 1968 roku został profesorem informatyki na Queen's University of Belfast, a w 1977 powrócił do Oksfordu, tworząc nowe laboratorium komputerowe, a następnie wydział informatyki. Obecnie jest emerytowanym profesorem na tym uniwersytecie i jednocześnie współpracuje z Microsoft w Cambridge. W swojej karierze naukowej otrzymał wiele istotnych wyróżnień:

- członkostwo kilku instytucji naukowych, w tym Fellow of the Royal Society (1982), Academia Europea (1989) oraz Royal Academy of Engineering (2005),
- doktoraty honorowe: Queen's University of Belfast (1987), University of Bath (1993), Athens University of Economics and Bussiness (2007) oraz szlachectwo brytyjskie nadane przez królową (2000),
- prestiżowe nagrody lub medale: ACM Turing Award (1980), Kyoto Prize (2000), IEEE John von Neuman Medal (2011).

Jest autorem lub współautorem czterech fundamentalnych dla informatyki książek:

1. O.-J. Dahl, E.W. Dijkstra and C.A.R. Hoare. *Structured Programming*. Academic Press, 1972.

2. C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall International, Series in Computer Science, 1985.
3. C.A.R. Hoare, M.J.C. Gordon. *Mechanised Reasoning and Hardware Design*. Prentice Hall 1992.
4. C.A.R. Hoare, He Jifeng. *Unifying Theories of Programming*. Prentice Hall International, Series in Computer Science, 1998.

Jego dorobek naukowy liczy około 200 pozycji, a liczba ich cytowań przekracza 20 tysięcy, co jest wyjątkowym osiągnięciem. Warto podkreślić, że Jego dorobkowi poświęcono kilka publikacji, w tym: *Oral History of Sir Antony Hoare*, autorstwa Jonathana Bowena (2006), czy *An Interview with C.A.R. Hoare*, autorstwa Shustek i Lon. To ostatnie opracowanie ukazało się w *Communications of the ACM* (2009). C.A.R. Hoare jest więc wybitnym naukowcem z dziedziny informatyki, znanym jako twórca tzw. sortowania szybkiego (*QuickSort*), logiki Hoare'a do weryfikacji poprawności programów oraz formalnego języka CSP (*Communicating Sequential Processes*) do specyfikacji współbieżnych procesów (w tym „problemu ucztujących filozofów"), a także jako autor języka OCCAM do programowania równoległego. Każdy student z informatyki w każdym kraju zgłębia wiedzę na ten temat.

Bardzo cenna jest Jego opinia na temat oprogramowania systemów informatycznych. Wskazuje On, że istnieją dwie metody budowy oprogramowania. Jedna polega na przyjęciu pewnych założeń upraszczających, tak by implementacja do systemu komputerowego nie zawierała żadnych błędów, oraz metoda druga uwzględniająca pełną złożoność systemu, ale nie analizująca problemu błędów. C.A.R. Hoare stwierdza, że pierwsza metoda jest znacznie trudniejsza do zrealizowania niż druga i taką tendencję potwierdza również współczesna inżynieria oprogramowania.

Osiągnięcia prof. C.A.R. Hoare'a stanowią prawdziwe kroki milowe w rozwoju informatyki. Jego książki zostały przetłumaczone na wiele języków i stanowią podstawowe podręczniki akademickie w nauczaniu informatyki. Dostarczają one zasad dobrego programowania, w których uwzględnia się problem współdziałania wielu procesów, ich zakleszczenia bądź zagłodzenia. Tego typu problemy pojawiają się również w przypadku złożonych systemów rozproszonych, które wspólnie i zdalnie realizują pewien zestaw zadań obliczeniowych. Tak więc Jego prace mają także ogromne znaczenie dla budowy nowoczesnych systemów informatycznych.

Trzeba zauważyć, że pracownicy Uniwersytetu Warszawskiego już w latach 1968-1970 nawiązali kontakty z C.A.R. Hoare'em dzięki Jego obecności na konferencjach informatycznych w Polsce. Kontakt ten przerodził się następnie we współpracę naukową dotyczącą wykorzystania metodyki CSP Hoare'a do specyfikacji systemów oprogramowania, w tym systemów operacyjnych, ulepszania metod programowania i weryfikacji programów. W konsekwencji ta współpraca zaowocowała interesującymi publikacjami oraz doktoratami, a także wizytą pracowników Uniwersytetu Warszawskiego na uniwersytecie w Oksfordzie i wielokrotnym przyznaniem stypendiów młodym warszawskim informatykom na prestiżową NATO-wską letnią szkołę inżynierii oprogramowania, prowadzoną przez samego Mistrza Hoare'a. Biorąc pod uwagę obecną szkołę informatyki zbudowaną na Uniwersytecie Warszawskim przez prof. Jana Madeya, należy stwierdzić, że nie odnosiłaby ona tak istotnych sukcesów w dziedzinie programowania bez wnikliwego zapoznania się oraz rozwinięcia wyników prac C.A.R. Hoare'a.

W związku z powyższym wniosek o nadanie tytułu i godności doktora *honoris causa* Uniwersytetu Warszawskiego wybitnemu uczonemu z Oksfordu, Sir Charlesowi Antony'emu Richardowi Hoare'owi w pełni popieram i uważam za zasadny. Włączenie tak wybitnej osobowości naukowej do grona wyróżnionych tym prestiżowym tytułem doda także splendoru cenionej i szeroko poważanej uczelni, jaką jest Uniwersytet Warszawski.

Prof. dr hab. Antoni Mazurkiewicz
Instytut Podstaw Informatyki
Polska Akademia Nauk

Opinia dla Senatu Uniwersytetu Warszawskiego w związku z nadaniem tytułu doktora *honoris causa* Sir Charlesowi Antony'emu Richardowi Hoare'owi

Sir Charles Antony Richard Hoare urodził się w 1934 roku w Kolombo na Cejlonie, o obecnej nazwie Sri Lanka, ówczesnej kolonii brytyjskiej na Oceanie Indyjskim. W środowisku informatycznym znany jest jako Tony Hoare. Jest absolwentem Uniwersytetu w Oksfordzie, kończąc tam studia w 1956 roku. Po odbyciu dwuletniej służby w Królewskiej Marynarce Wojennej odbył staż na Uniwersytecie Moskiewskim w grupie badawczej Andrieja Kołmogorowa, gdzie zajmował się zagadnieniami translacji komputerowej (stąd jest jednym z nielicznych, o ile nie jedynym informatykiem brytyjskim władającym językiem rosyjskim). Po powrocie do Wielkiej Brytanii był profesorem Uniwersytetu Królowej w Belfaście, potem Uniwersytetu w Oksfordzie, gdzie obecnie jest profesorem emerytowanym.

Tony Hoare jest osobą reprezentującą całą epokę rozwoju informatyki, od jej początków aż do czasów współczesnych. Swą działalność poświęcił głównie zagadnieniom dotyczącym oprogramowania, odgrywającym kluczową rolę dla powszechnego stosowania techniki komputerowej. Można bez przesady stwierdzić, że dzięki pracom uczonych związanych z informatyką w tej epoce, a wśród nich szczególnie profesora Hoare'a, mamy dziś tak burzliwy rozwój i powszechną stosowalność techniki cyfrowej we współczesnym świecie. Tony Hoare swymi osiągnięciami naukowymi przyczynił się w znacznej mierze do istnienia informatyki w jej współczesnym kształcie.

Za swą wieloletnią działalność Hoare został uhonorowany wieloma prestiżowymi wyróżnieniami. Do najważniejszych należy przyznanie mu w 1980 roku przez Association for Computing Machinery nagrody Alana Turinga za wybitne osiągnięcia naukowe w informatyce

(ang. Computer Science), uważanej niekiedy za odpowiednik nagrody Nobla w tej dziedzinie. Innym wyróżnieniem, o podobnym wydźwięku i randze, jest uhonorowanie go medalem von Neumanna w 2011 roku. Tony Hoare jest ponadto doktorem *honoris causa* uniwersytetów w Belfaście (Płn. Irlandia), Atenach (Grecja) i Bath (Anglia). W 2000 roku otrzymał z rąk brytyjskiej królowej tytuł szlachecki za służbę na polu edukacji i informatyki.

Omawiając dokonania naukowe Tony'ego Hoare'a, należy uznać za szczególną pozycję w Jego dorobku koncepcję komunikujących się procesów sekwencyjnych. Koncepcja ta, odwołująca się bezpośrednio do intuicji związanych z przetwarzaniem rozproszonym, zaowocowała stworzeniem narzędzi formalnych do opisu i badania takich systemów. Jak to często bywa w nauce, kolejne generacje zainicjowanej idei można, z upływem czasu, odnaleźć w pozornie odległych dziedzinach nauki i praktyki, rzecz jasna w postaci zmodyfikowanej i dostosowanej do bieżących, już zaktualizowanych potrzeb. Tak i stało się z koncepcją CSP (*Communicating Sequential Processes*) i ideami tam zawartymi. Kolejne generacje informatyków czerpały z przekazanych tam idei inspirację do swoich badań i dokonań. Można bez przesady stwierdzić, że bez koncepcji CSP dziedzina informatyki związana z przetwarzaniem rozproszonym wyglądałaby dziś całkiem odmiennie i niewątpliwie byłaby znacznie zubożona.

Jest godne podkreślenia, że Tony Hoare – podobnie jak wielu informatyków pokolenia mu współczesnego – przeplatał swą pasję badawczą z techniką programistyczną. Wkład Jego w rozwój tej techniki, łatwy do zignorowania w dorobku naukowym, jest nie do przecenienia w środowisku twórców programów i systemów komputerowych. Korzystają oni z Jego koncepcji, być może nie zdając sobie sprawy komu zawdzięczają ułatwienie tworzenia i zwiększanie efektywności wykorzystania narzędzi informatycznych. Jego algorytm „QuickSort", usprawniający jeden z podstawowych elementów konstrukcji programistycznych, wykracza już poza udogodnienie techniczne, a jest osiągnięciem współczesnej algorytmiki. W tej konstrukcji jako jeden z pierwszych zastosował technikę „dziel i rządź" (*divide et impera*), która wkrótce stała się jednym z kanonów współczesnej algorytmiki. Także i w samej technice programowania takie udogodnienia programistyczne jak konstrukcja „Case of", wprowadzona przez Hoare'a, która w znakomitym stopniu upraszcza

i ułatwia tworzenie programów komputerowych, jest Jego zasługą. Wielu programistów działających w latach 70. ubiegłego stulecia, w tym piszący te słowa, jest Mu wdzięcznych za stworzone ułatwienie ich pracy dzięki wynalezionej przez niego konstrukcji.

W ubiegłym stuleciu szczególnie intensywny rozwój dotyczył językowych narzędzi komunikacji „człowiek – komputer". Wraz z rozwojem środków technicznych informatyki narzędzia te rozwinęły się i zmodyfikowały, jednak wiele z idei zrealizowanych we współczesnych językach i systemach programowania, bądź ukrytych w ich kodach, zawdzięczamy Tony'emu Hoare'owi. Jego udział w dyskusjach dotyczących następcy Algolu – pierwowzoru języka Pascal – jest nie do przecenienia jako wkład do koncepcji nowoczesnych języków programowania.

Podstawowym problemem w informatyce, do dziś aktualnym, jest uzasadnianie poprawności tworzonych konstrukcji. W szczególności jest wielce istotne uzasadnienie poprawności spodziewanych rezultatów konstruowanego (lub już skonstruowanego) programu drogą zdyscyplinowanego i formalnego rozumowania. Wymaga to objęcia wspólnym formalizmem i jednolitą metodą dwóch różnych systemów: systemu statycznego, dobrze ugruntowanego w matematyce, korzystającego z aparatu logiki formalnej, i systemu dynamicznego, operującego działaniami (algorytmami) i ich składaniem, jakim jest formalny opis funkcjonowania programów. Wśród wielu propozycji i dokonań na tym polu, począwszy od prac Floyda i Dijkstry, poprzez logiki algorytmiczną i dynamiczną aż do współczesnych systemów dowodzenia poprawności programów, logika Hoare'a zajmuje poczesne miejsce, jako jeden z pierwszych systemów formalnych, pozwalających wykazać poprawność warunku spełnionego przez wynik programu (a więc opisu jego rezultatów) na podstawie ściśle określonych reguł działania programu i jego warunku wstępnego (a więc opisu danych początkowych programu). Logika ta, poza listą reguł wynikających z konstrukcji programu, ma reguły wnioskowania pozwalające uzasadniać poprawność konstrukcji bardziej złożonych przez składanie ich z prostszych fragmentów. Dokonania w tej dziedzinie przyniosły Hoare'owi, autorowi tego systemu logicznego, nazwanego Jego nazwiskiem jako „Hoare Logic", uznanie światowego środowiska informatycznego.

Miałem kilkakrotnie sposobność osobistego kontaktowania się z Tony'm Hoare'em. Z tych spotkań wynoszę przeświadczenie o osobistej

skromności, życzliwości, otwartości, przychylnego krytycyzmu, także w stosunku do własnych osiągnięć. Spotkania z Nim zawsze były okazją do pojawienia się nowych idei i pomysłów oraz przedyskutowania istniejących.

W podsumowaniu pragnę stwierdzić, że Uniwersytet Warszawski, choć ma już ugruntowaną renomę wśród uniwersytetów światowych, jeszcze zyska na swym prestiżu posiadając w gronie swych doktorów honorowych osobę taką, jak Tony Hoare. Jest on reprezentantem nowoczesnej nauki, wysoko cenionym w międzynarodowym środowisku naukowym, czego dowodami są fakty przytoczone powyżej, i tym, który poszerza naszą wiedzę i umiejętności, oraz przyczynia się do rozwoju informatyki jako narzędzia współczesnej cywilizacji. Uważam, że Tony Hoare w pełni zasługuje na proponowane wyróżnienie i propozycję nadania mu stopnia doktora *honoris causa* Uniwersytetu Warszawskiego popieram z pełnym przekonaniem i bez najmniejszych zastrzeżeń.

Uroczystość wręczenia dyplomu doktora *honoris causa* Uniwersytetu Warszawskiego Sir Charlesowi Antony'emu Richardowi Hoare'owi

Prof. dr hab. Władysław M. Turski

LAUDACJA

Mało brakowało bym dzisiejszego dostojnego Doktoranta poznał osobiście w 1960 roku. Kończyłem wtedy studia na Mechaniczno-Matematycznym Wydziale Uniwersytetu Moskiewskiego, który – w ramach brytyjsko-radzieckiej wymiany naukowej – właśnie gościł aspiranta statystki, C.A.R. Hoare'a, pracującego pod opieką A.N. Kołmogorowa. Gdybyśmy się wówczas rzeczywiście spotkali, być może dowiedziałbym się kilka lat wcześniej, niż nastąpiło to w rzeczywistości, o Jego genialnym pomyśle: algorytmie *Quicksort* porządkowania zbiorów, od pół wieku stanowiącym niezbędny element wykształcenia każdego informatyka, niezastąpiony wzorzec i nieugięty probierz jakości tej części każdego oprogramowania, która dotyczy sortowania.

Droga, którą przebył dwudziestosześcioletni Tony Hoare do odkrycia *Quicksortu* biegnie meandrami, pozornie wcale nie prowadzi w kierunku Jego przyszłej wielkiej kariery naukowej.

Urodzony na Cejlonie, ze szczęśliwego dzieciństwa zachował kiplingowskie prawie wspomnienia szkolnych wypadów do dżungli, dzikich słoni i tygrysów. Mały Tony chciał wtedy zostać pisarzem. Po wojnie rodzina Hoare'ów przeniosła się do Anglii i Tony'ego zapisano do King's School w Canterbury, najstarszej szkoły Królestwa, z tradycją sięgającą końca VI wieku. Tu nastoletni chłopiec znalazł sobie sanktuarium w bibliotece, rozsmakował się w filozofii i zjadliwościach G.B. Shawa. Precyzja i elegancja języka cechują Go do dziś.

W odpowiednim czasie, zgodnie z rodzinną tradycją, C.A.R. Hoare wstąpił do Merton College w Oksfordzie, wybierając kierunek studiów „Lit Hum", obejmujący łacinę i grekę, historię starożytną, językoznawstwo i literaturę oraz klasyczną i współczesną filozofię. Zaraz po studiach został powołany do służby wojskowej, którą odbył w Marynarce Wojennej, intensywnie ucząc się języka rosyjskiego. W 1958 r. wrócił do Oksfordu, gdzie podjął zawodowe studia statystyczne, kontynuacją których był wyjazd do Moskwy. W ramach tych studiów zdobył praktyczną umiejętność

programowania w języku Mercury Autocode na komputer Ferranti-Mercury i napisał swoje pierwsze programy.

Przedmiotem wielkiego zainteresowania w owym czasie był maszynowy (komputerowy) przekład z jednego języka naturalnego na inny, np. z angielskiego na rosyjski. Tym właśnie zagadnieniem zajmował się Hoare w Moskwie.

W ówczesnych komputerach wielkie zbiory danych, takie jak np. słowniki, gromadzono na taśmach magnetycznych, które można było odczytywać (i zapisywać) w jednym tylko kierunku. Kiedy zdarzyło się, że potrzebny był element poprzedzający właśnie odczytany, taśmę trzeba było zwinąć do początku a następnie znowu rozwinąć do szukanego elementu, co zajmowało masę czasu. Nic więc dziwnego, że intensywnie poszukiwano sprawnych metod takiego sortowania zbiorów (np. wyrazów tłumaczonego tekstu), by jak najbardziej ograniczyć liczbę niezbędnych, czasochłonnych przewinięć taśmy (np. ze słownikiem). Wymyślony przez Hoare'a algorytm *Quicksort* jest pod tym względem optymalny; ma on jednak inną jeszcze cechę: jest algorytmem rekurencyjnym, co – z grubsza powiedziawszy – oznacza, iż jego wykonanie odwołuje się znowu do tego samego algorytmu; poprawny algorytm rekurencyjny, oczywiście, nie wymaga powtarzania tego *ad infinitum*.

Programiści owych lat bardzo nie lubili rekurencyjnych algorytmów. Po pierwsze dlatego, że z dostępnymi wtedy narzędziami intelektualnymi trudno było wykazać, że dany algorytm jest poprawny, a po drugie dlatego, że w dostępnych językach programowania nie można było w sposób dostatecznie prosty wyrazić rekurencji. Hoare podjął obydwa te wyzwania.

W czasie gdy przebywał w Moskwie, brytyjska firma elektroniczna Elliott Brothers urządziła tam wystawę swych wyrobów, m.in. komputerów, jak wtedy mówiono: „do obliczeń naukowych", na której to wystawie zatrudniła Tony'ego w charakterze tłumacza, a po jego powrocie do Anglii z miejsca zaoferowała Mu stałą pracę programisty, płacąc specjalny dodatek (100 funtów rocznie!) ze względu na znajomość języka rosyjskiego. Komputery Elliotta cieszyły się sporą popularnością w całej Europie, kilka z nich z powodzeniem pracowało także w Polsce. Kiedy firma podjęła decyzję o produkcji mocniejszego komputera, postanowiła wyposażyć go w nowy, sprawniejszy język programowania. Zamiast wymyślać swój własny, co w owych latach było dość powszechnym sportem,

Hoare zaproponował, by nowe komputery wyposażyć w translator ALGOLu-60, języka opracowanego przez międzynarodowy zespół uczonych (IFIP WG 2.1), uznawanych dziś za koryfeuszy metodologii programowania. ALGOL-60 w pełni uwzględniał rekurencję. Kierownictwo Elliott Brothers przyjęło propozycję, powstał czteroosobowy zespół pod kierownictwem Hoare'a. W skład tego zespołu weszła jedna osoba z uprzednim praktycznym doświadczeniem budowy translatorów, przyszła Żona Tony'ego – Jill Pym. Projekt zakończył się powodzeniem, a jego kierownik został dokooptowany do WG 2.1.

U zarania swej zawodowej kariery, C.A.R. Hoare przepracował osiem lat w przemysłowej firmie komputerowej, poznał wady i zalety pracy w komercyjnym środowisku. Na całe życie zachował zrozumienie wymagań, jakie taka praca stawia przed informatykami, a w szczególności niezłomne przekonanie, że solidna praca przemysłowego informatyka wymaga jego twórczego zaangażowania. W świecie, gdzie panują animozje między informatyką teoretyczną i praktyczną, akademicką i przemysłową, taka postawa i jej stałe poświadczanie pracą własną i pracą kierowanych przez siebie zespołów zasługuje na najwyższy szacunek.

Hoare jest już uczonym o międzynarodowej reputacji. Intensywnie działa w WG 2.1, we współpracy z N. Wirthem przygotowuje projekt następcy ALGOLu-60, zawierający wiele rewolucyjnych ulepszeń. Niestety, projekt nie zyskuje uznania większości zespołu; Wirth kontynuuje pracę na własną rękę, powstaje ALGOL-W a następnie Pascal, który na dwie dekady zdominował światową praktykę i dydaktykę programowania, mimo narastającej w przemyśle i niektórych kręgach akademickich tendencji tworzenia monstrualnych języków programowania (np. Ada czy ALGOL-68), przeznaczonych do konstruowania tzw. wielkich programów (m.in. systemów operacyjnych).

Hoare aktywnie przeciwstawia się tej tendencji: „...in spite of all the work that has been expanded on the design of general and special purpose programming languages, the role they play in the design and implementation of a large system is at best minimal" – to cytat z Jego wystąpienia na czterodniowym seminarium w sierpniu 1970 r. w Jabłonnej, w całości poświęconym zagadnieniom wytwarzania wielkich programów.

ALGOL-60 i jego ideowi następcy (Pascal, Simula) pod względem składni były językami w pełni formalnymi, co pozwalało na w pełni automatyczny rozbiór logiczny napisanych w nich programów. Semantyka

tych języków była jednak opisana nieformalnie, aczkolwiek – w wielu przypadkach – bardzo precyzyjnie. Wobec wielkiej rozmaitości struktury i organizacji używanych wówczas komputerów możliwe więc było, że jeden i ten sam program wykonywany na różnych maszynach dawał wyniki *w zasadzie* takie same, ale różniące się w takich szczegółach, które wymykały się nieformalnej definicji semantyki języka programowania. Zauważmy, że mniej lub bardziej udane próby formalizacji semantyki języków programowania są podejmowane do dziś i wielu znawców uważa, że problem ten niekoniecznie znajdzie powszechnie satysfakcjonujące rozwiązanie.

Jeśli program może dawać różne wyniki, to który z nich należy uznać za właściwy? Albo inaczej, jeśli wiemy jakie wyniki są właściwe, to jaki program należy uznać za poprawny?

W 1968 r. Hoare obejmuje katedrę informatyki na Queen's University w Belfaście i zwraca swoje zainteresowania ku zagadnieniu poprawności programów. Zagadnieniem tym zajmowało się wielu wybitnych uczonych, od twórcy koncepcji programowanego komputera, J. von Neumana, przez takich pionierów informatyki jak J. McCarthy i R. Floyd.

Pomysł Hoare'a sprowadzał się do tego, by treść instrukcji programu traktować jako ograniczenia dowolności wyniku ich wykonania: zamiast dążyć do absolutnie precyzyjnego określenia wyniku danej operacji (wykonania instrukcji) – ustalić granice, w których *każdy* wynik będzie uznawany za poprawny. Formalnie wyraża się to tzw. *trójką Hoare'a*, $P\{Q\}R$, interpretowaną jak następuje: Instrukcja Q wykonana w stanie (obliczenia) spełniającym warunek P ukończy się w stanie spełniającym warunek R. Wystarczy teraz przyjąć za aksjomaty trójki definiujące elementarne instrukcje języka programowania i ustalić reguły przekształcania trójek przy składaniu instrukcji (podobne do reguł wnioskowania w klasycznej logice), żeby dla dowolnego programu można było wyliczyć „supertrójkę" obejmującą cały program, tj. ustalić jakie warunki będzie spełniać stan obliczenia po wykonaniu programu rozpoczętym w stanie spełniającym wskazane warunki początkowe.

Opisana tu, w znacznym uproszczeniu, *logika Hoare'a* jest kamieniem węgielnym współczesnej metodologii programowania. Doczekała się wielu modyfikacji, rozszerzeń i wersji, ale nie ma podręcznika ani monografii dotyczących poprawności programów, w których nie odgrywałaby pierwszoplanowej roli. Opublikowana w 1969 r. praca „An axiomatic

basis for computer programming", w której Hoare wyłożył jej podstawy, doczekała się prawie 20 tysięcy cytowań.

W roku 1977 Prof. Hoare przenosi się do Oksfordu, obejmując najpierw kierownictwo Programming Research Group, quasi-instytutu badawczego, a później – także pierwszą w Oksfordzie katedrę informatyki. Kontynuacja dociekań dotyczących poprawności programów owocuje znakomitą monografią „Structured Programming" napisaną wspólnie z E.W. Dijkstrą i O.-J. Dahlem, a także wieloma osiągnięciami współpracowników i uczniów, których Hoare po mistrzowsku dobiera. Tytułem przykładu, Jean-Raymond Abrial rozwinął w Oksfordzie język Z do opisu i analizy sprzętu komputerowego, m.in. wykorzystany przez ówczesnego giganta komputerowego, koncern IBM, do udoskonalenia CICS (Customer Information Control System), jednego z najbardziej wówczas dochodowych produktów informatycznych. Nawiasem powiedziawszy, prace te zostały wyróżnione Nagrodą Królowej Elżbiety II w dziedzinie technologii.

W polu zainteresowania Prof. Hoare'a wnet pojawia się kolejny ważny temat: współdziałanie niezależnych procesorów. Impuls wyszedł ze strony przemysłu. Coraz bardziej złożone komputery i ich systemy zawierały wiele odrębnych procesorów, współdziałających w realizacji jednego zadania. Najprostszym przykładem jest współdziałanie procesora wykonującego obliczenia i procesora sterującego urządzeniem (lub urządzeniami) wyjściowym: pierwszy z nich co pewien czas generuje porcje informacji, które drugi, po ewentualnym przekształceniu do eleganckiej postaci, drukuje lub wyświetla na ekranie. Instrumentarium intelektualne stworzone do badania poprawności oprogramowania pojedynczych procesorów nie pasowało do oprogramowania systemów wieloprocesorowych, w tym – co krytycznie ważne z praktycznego punktu widzenia – do badania i poprawnego konstruowania systemów operacyjnych, stanowiących podstawę użytkowania komputerów.

Pierwszym rozwiązaniem zaproponowanym przez Prof. Hoare'a (w dużej mierze pod wpływem i we współpracy z P. Brinch-Hansenem, twórcą systemu operacyjnego RC-4000 zainstalowanego m.in. w puławskich „Azotach") był monitor – konstrukcja programistyczna w zdyscyplinowany i dający się sformalizować sposób udostępniająca usługi o aksjomatycznie określonych własnościach. Użycie monitorów, ładnie współgrających z koncepcjami Simuli, języka programowania stworzonego przez O.-J. Dahla, pozwalało zastosować rachunek logiczny do analizy

poprawności szerokiej klasy oprogramowania systemów wieloprocesorowych; bywało to jednak w praktyce dość trudne, a same monitory były zbyt „gruboziarniste": jako konstrukcje programistyczne dość odległe od rozwiązań sprzętowych same wymagały niebanalnych dowodów poprawności realizacji.

Kolejne rozwiązanie zaproponowane przez C.A.R. Hoare'a okazało się tyleż proste, co fundamentalnie skuteczne: współdziałanie procesów ograniczył On do wymiany komunikatów. Do repertuaru instrukcji wykonywanych przez procesy wprowadza się dwie, służące odpowiednio do nadawania i odbierania komunikatów; faktyczna wymiana następuje wtedy, gdy proces „odbiorca" dochodzi do wykonania instrukcji odbioru, a proces „nadawca" jest gotów do wykonania instrukcji nadania. Jeśli jeden z komunikujących się procesów nie doszedł jeszcze do odpowiedniej instrukcji, a drugi jest już gotów nawiązać łączność, ten ostatni czeka na gotowość pierwszego. To rozwiązanie – bardzo bliskie prostemu schematowi dwu procesorów fizycznie połączonych jednym przewodem – ogłoszone w 1978 r. w pracy „Communicating sequential processes" i rozwinięte w monografii (1985) pod tym samym tytułem, stanowi po dziś dzień podstawę analizy i konstrukcji wieloprocesorowych systemów informatycznych, zarówno programistycznych, jak i sprzętowych. Zbudowany w Oksfordzie na podstawie CSP język programowania *occam* posłużył np. firmie INMOS do zweryfikowania poprawności drugiej generacji transputerów (mikroprocesorów), wzbogaconej o arytmetykę zmiennopozycyjną. Zastosowanie metod formalnej weryfikacji zamiast tradycyjnego testowania pozwoliło przyspieszyć wprowadzenie tej wersji transputera na rynek o cały rok. INMOS i Oksfordzkie Laboratorium Komputerowe i za ten sukces otrzymały Nagrodę Królowej w dziedzinie technologii.

W 1999 r. Prof. Hoare przechodzi na emeryturę i obejmuje stanowisko Principal Researcher w brytyjskiej filii Microsoftu w Cambridge, gdzie kontynuuje badania nad problemami poprawności oprogramowania i współbieżności procesorów współdzielących pamięć. Można powiedzieć, że piękne i bogate życie zawodowe zatoczyło krąg: od Elliott Brothers do Microsoftu. Absolwent „Lit-Hum" Oksfordu doskonale wie jednak, że nie sposób wejść dwa razy do tej samej rzeki. Minione pół wieku zmieniło informatykę z interesującej nowalijki w jedną z głównych kariatyd naszej cywilizacji.

Zmiana ta jest dziełem wielu firm przemysłowych i przedsiębiorców, a także wielu wybitnych wynalazców i badaczy. Tony Hoare należy do wąskiego grona najwybitniejszych z nich. W dowód uznania tego faktu wybrano Go do obydwu brytyjskich akademii: Royal Society i Royal Academy of Engineering, a brytyjskie Towarzystwo Komputerowe przyznało Mu rzadką rangę Dostojnego Członka (Distinguished Fellow). Wielkość Jego osiągnięć uznano po obu stronach Pacyfiku: otrzymał amerykańską nagrodę i medal Turinga (uważany za informatycznego Nobla) i japońską Nagrodę Kyoto (którą, *nota bene*, wyróżniono także innego doktora *honoris causa* naszego Uniwersytetu – A. Wajdę). Kilka uniwersytetów nadało Mu godność doktora *honoris causa*, zaś Królowa Elżbieta II – tytuł szlachecki za zasługi dla edukacji. Nie tylko bowiem stworzył w Oksfordzie jeden z najlepszych na świecie ośrodek dydaktyczno-badawczy w dziedzinie informatyki, który m.in. gościł niejednego informatyka z Warszawy, ale przez ponad 20 lat był dyrektorem Letniej Szkoły w Marktoberdorfie, gdzie na dwutygodniowych sesjach czołowi informatycy dzielili się swą wiedzą i pomysłami badawczymi z setką wybitnie uzdolnionych doktorantów z wszystkich zakątków świata. Warto dodać, że choć Szkoła była finansowana przez NATO, nawet w latach zimnej wojny przyjmowała i subsydiowała słuchaczy z Uniwersytetu Warszawskiego.

Nadając godność doktora *honoris causa* C.A.R. Hoare'owi wyrażamy głębokie uznanie wobec Jego życiowych dokonań, czcimy wielkiego uczonego i organizatora badań naukowych, składamy podziękowania za dziesięciolecia życzliwości i przyjaźni.

Nowemu Doktorowi Uniwersytetu Warszawskiego życzymy długich lat owocnej pracy, a sobie – dalszej z Nim owocnej współpracy.

Przemówienie dziekana Wydziału Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego Prof. dr. hab. Andrzeja Tarleckiego

Szanowni Państwo,

Wystąpienie w imieniu Wydziału Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego na tej wspaniałej uroczystości nadania doktoratu *honoris causa* dla Profesora Tony'ego Hoare'a jest dla mnie zaszczytem i przywilejem jednocześnie. Zebraliśmy się tu z końcem roku ogłoszonego Rokiem Informatyki na Uniwersytecie Warszawskim i z końcem miesiąca, w którym świętujemy rocznicę założenia naszego Uniwersytetu. Dla mnie jednak jest to też bardzo szczególne wydarzenie dlatego, że jest to pierwszy doktorat *honoris causa* nadawany przez Uniwersytet Warszawski informatykowi. Nie ma w tym nic zaskakującego: informatyka to wciąż bardzo młoda dyscyplina. Dzięki temu jednak możemy spotykać i nagradzać osoby, które informatykę kształtowały od jej początków – i właśnie profesor Hoare jest bez wątpienia jedną z nich.

Byłoby teraz właściwe, bym przedstawił długą listę licznych osiągnięć naukowych profesora Hoare'a, wyników i idei, które stanowią Jego wkład w rozwój informatyki od ponad pół wieku. W żaden sposób nie mógłbym jednak dorównać wygłoszonemu przed chwilą *laudatio* profesora Turskiego, który nie tylko doświadczał, ale przede wszystkim sam brał aktywny udział w tym rozwoju informatyki niemal od jej początków. Pozwólcie Państwo, że w tym momencie serdecznie podziękuję profesorowi Turskiemu za kluczową rolę promotora w procesie nadania tego doktoratu honorowego i za świetne *laudatio*. Chciałbym też podziękować recenzentom, profesorowi Mazurkiewiczowi, profesorowi Krawczykowi i profesorowi Idziakowi za wyczerpujące, wyśmienite opinie o osiągnięciach profesora Hoare'a. Jak powiedziałem, nie udałoby

mi się im dorównać, więc pozwólcie Państwo, że powiem kilka słów o... sobie.

Naukę programowania i studia informatyczne rozpocząłem na Uniwersytecie Warszawskim w 1975 roku. Od początku kładziono tu wielki nacisk na nauczanie *programowania strukturalnego*, zgodnie z pomysłami wyłożonymi w sławnej książce „*Structured Programming*", której współautorem był profesor Hoare. Uczono nas projektowania i konstruowania programów wraz z formułowaniem niezbędnych wymagań, asercji, niezmienników pętli – i na ile to możliwe, dowodami poprawności programów. Te ostatnie wkrótce przybrały i dla nas formalną postać logiki Hoare'a. Jej badanie to jedna z moich wczesnych fascynacji naukowych; wciąż pamiętam gorące dyskusje dotyczące (nie)pełności tej logiki. Kilka lat później mój doktorat przedstawiał *język programów specyfikowanych* – z pewnego punktu widzenia po prostu wprowadzając rozszerzenie logiki Hoare'a na dodatkowe konstrukcje programistyczne. A przykładem wykorzystywanym w całej rozprawie była wersja opracowanego przez profesora Hoare'a algorytmu *quicksort*. Nieco wcześniej, moja praca magisterska dotyczyła *algebr dla semantyk wejścia/wyjścia* – abstrakcyjnej algebraicznej wersji aksjomatycznej semantyki programów, wprowadzonej dla Pascala w artykule, którego współautorem był profesor Hoare. Po doktoracie spędziłem jakiś czas w Edynburgu, skąd jeden z pierwszych moich wyjazdów zawiódł mnie do Oksfordu. Większość tej wizyty poświęciłem na dyskusję z grupą Z, pracującą w kierowanym przez profesora Hoare'a PRG. Pamiętam też jednak miłą rozmowę z nim samym. Mówił głównie o CSP, co przyznaję, nie było wówczas w centrum moich zainteresowań. Niemniej jakiś czas później poświęciłem sporo wysiłku próbom wprowadzenia do powstającego standardu VDM dodatkowych konstrukcji, które pozwoliłyby na wyrażenie oczekiwanej semantyki CSP. W późnych latach osiemdziesiątych był to dla mnie raczej poboczny temat badań. Jedną z moich najważniejszych fascynacji naukowych były wówczas, i pozostają do dziś, problemy tzw. behawioralnej interpretacji specyfikacji oprogramowania – wyrastające wprost z prac i idei o reprezentacji danych przedstawionych przez profesora Hoare'a na początku lat siedemdziesiątych. W latach dziewięćdziesiątych zaszczycony byłem zaproszeniem na wykłady w znanej szkole letniej w Marktoberdorf, której współdyrektorem był profesor Hoare. Zaskoczył mnie wówczas zupełnie zadając całą serię ciekawych

pytań z zakresu teorii kategorii, na które niestety nie umiałem udzielić satysfakcjonujących odpowiedzi. Dowiedziałem się, że było to związane z jego nowymi pracami dotyczącymi teorii programowania. Jeszcze później, jednym z najważniejszych wydarzeń w mojej działalności organizacyjnej był ETAPS 2003, kongres naukowy, który zorganizowaliśmy w Warszawie. Do najważniejszych wydarzeń kongresu z kolei należał plenarny wykład profesora Hoare'a o wielkim wyzwaniu dzisiejszej informatyki, tzw. *weryfikującym kompilatorze* – wskazał on niektóre kierunki ostatnich badań związanych z weryfikacją oprogramowania, prowadzonych przez moją grupę na UW.

Nie, nie musiałem w tej historii niczego naciągać. Jestem przekonany, że jest ona typowa dla informatyków mego pokolenia. Wyniki i idee profesora Hoare'a tworzą podstawy tego, czego my, informatycy z Uniwersytetu Warszawskiego i z całego świata, uczyliśmy się jako studenci i nad czym pracowaliśmy jako naukowcy. Pozwolę sobie dodać, że bezcenną wartością Jego prac jest poważne, odpowiedzialne podejście do problemów konstrukcji oprogramowania, gdzie poprawność i jakość pozostają niezbywalnym najważniejszym celem.

Nadawany dziś doktorat *honoris causa* Uniwersytetu Warszawskiego, najwyższe akademickie wyróżnienie jakie możemy ofiarować, to świadectwo naszego uznania i naszej wdzięczności za wielki wkład profesora Hoare'a w rozwój naszej dyscypliny, od algorytmu *quicksort* sprzed ponad pół wieku po weryfikujący kompilator i inne osiągnięcia w przyszłości.

Dziękujemy!

Wystąpienie Doktora *Honoris Causa* Sir Charlesa Antony'ego Richarda Hoare'a

Z ogromną przyjemnością przyjmuję to najwyższe wyróżnienie w postaci przyznania mi tytułu Doktora *Honoris Causa* Uniwersytetu Warszawskiego. Pragnę złożyć wyrazy serdecznej wdzięczności zarówno całemu Uniwersytetowi, jak i wszystkim tu obecnym osobom, które towarzyszą mi w czasie tej wspaniałej uroczystości.

Bardzo się cieszę, iż mam okazję po raz kolejny odwiedzić Polskę – kraj, w którym miałem szczęście wielokrotnie bywać w przeszłości. Po raz pierwszy byłem tu w roku 1960. Pod koniec ostatniego roku moich studiów na Państwowym Uniwersytecie Moskiewskim pracowałem przez dwa tygodnie jako tłumacz w czasie wystawy, na której prezentowano w Moskwie brytyjskie przyrządy naukowe. Jednym z eksponatów był komputer 803, skonstruowany przez firmę Elliott Brothers z Londynu; firma ta przywiozła go na wystawę w niewielkiej furgonetce. Po zakończeniu wystawy komputer sprzedano radzieckiemu instytutowi (rzecz jasna, tajnemu), natomiast mnie zaoferowano darmowy przejazd w pustej już furgonetce powracającej do Anglii. Po powrocie przyjąłem propozycję od firmy Elliott Brothers i rozpocząłem tam swoją pierwszą pracę jako programista.

Z czasów tamtej podróży nadal mam żywo w pamięci moment przekraczania granicy między ZSRR a Polską. Kontrast był naprawdę niesamowity: natychmiast zauważyłem ogromną różnicę – zaniedbane, leżące odłogiem i nieprzyjazne tereny zastąpił widok dobrze utrzymanych obszarów rolniczych. Mieszkańcy również byli zupełnie inni – widząc ich otwartość i pogodę niemal skłonny byłem uwierzyć, iż są ludźmi wolnymi. Przez kolejne pięć lat kilkakrotnie miałem okazję odbywać podróże przez Polskę furgonetką wiozącą komputery mojej firmy do Moskwy na coroczną letnią wystawę, po której następowała ich sprzedaż. Często zdarzało się, że zatrzymywaliśmy się na nocleg w Warszawie. Pewnego razu nocowaliśmy również w hotelu Bristol, w którym mieszkam podczas tej wizyty.

Wiele moich miłych wspomnień wiąże się z późniejszymi spotkaniami, warsztatami i konferencjami organizowanymi w Polsce. W październiku 1966 roku przyleciałem do Warszawy na pamiętne posiedzenie międzynarodowej Grupy Roboczej (Working Group – WG 2.1), pracującej nad projektem języka programowania, który rozpowszechnił się następnie pod nazwą ALGOL 68. Miałem wówczas przyjemność poznać polskich członków tej grupy – Antoniego Mazurkiewicza i Władysława Turskiego oraz obserwatorów – Jana Madeya i Andrzeja Salwickiego. Ogromnie się cieszę, że i dzisiaj mogę ich zobaczyć w tej sali. Z innych jednak względów nasze dawne spotkanie dotyczące ALGOL-u nie było dla mnie najprzyjemniejsze. Na tym posiedzeniu bowiem ostatecznie zrezygnowano z języka, nad którym pracował wraz ze mną Niklaus Wirth, i którego opracowanie zleciła nam wcześniej sama Grupa Robocza. Język ten został opublikowany w czerwcu 1966 roku jako „Przyczynek do rozwoju języka ALGOL". Opracowany przez nas język znany był później pod nazwą ALGOL W.

Wizyta w Polsce, która najbardziej zapadła mi w pamięć, była znacznie przyjemniejsza. Brałem wówczas udział w konferencji, która odbywała się w Jabłonnie, było to w roku 1970, a tematem konferencji było „Efficient Production of Large Programs". Władysław Turski, mój promotor na dzisiejszej uroczystości, przewodniczył wówczas konferencji. Językiem oficjalnym był angielski, jednak był on językiem ojczystym zaledwie dla trzech jej uczestników. Większość obecnych władała językiem rosyjskim w stopniu znacznie lepszym niż angielskim. W tej sytuacji Władysław wziął na siebie zadanie dokonywania niezbędnego przekładu angielsko-rosyjskiego i rosyjsko-angielskiego. Robił do wspaniale – moja znajomość rosyjskiego pozwalała mi to ocenić. Tłumaczył bezbłędnie, słuchając fragmentu prezentacji w jednym języku i przedstawiając jego przekład na drugi. Rzecz jasna, ani angielski, ani rosyjski nie był jego językiem ojczystym.

W pewnej chwili wysłuchał długiego fragmentu przedstawionego po rosyjsku i, zwróciwszy się do grupy anglojęzycznej, przetłumaczył go doskonale na… rosyjski. Nie zdawał sobie sprawy, że w myślach dokonał dwukrotnego tłumaczenia. Najpierw, gdy słuchał rosyjskiego prelegenta, tłumaczył z rosyjskiego na angielski, a następnie, gdy mówił do grupy anglojęzycznej, dokonywał ponownego przekładu na rosyjski. Robił to zupełnie nieświadomie i był całym zdarzeniem równie rozbawiony,

jak jego słuchacze. Wreszcie (już po raz trzeci) świetnie przetłumaczył cały akapit na angielski.

To właśnie podczas tamtej konferencji powstał najbardziej znany z moich epigramatów. Wyraziłem w nim pogląd, iż wewnątrz każdego dużego programu jest mniejszy, który chce się z niego wydostać. Stanowiło to adaptację znanego wówczas powiedzenia dotyczącego osób otyłych, z których ciała próbuje się uwolnić inny, szczupły człowiek. O tym, że popularność mojego epigramatu trwała później wiele lat dowiedziałem się, gdy zobaczyłem jego kopię zdobiącą kubek do kawy. Dla naukowca publikacja na kubku do kawy jest prawdziwą nobilitacją i wytycza mu drogę do największej sławy.

Cieszę się z kolejnego spotkania z Władysławem Turskim, z prawdziwą przyjemnością wysłuchałem tak pochlebnej dla mnie Laudacji, opisującej moją działalność. Szczęśliwie i wielce chwalebnie Władysław wykazał się tu zdolnością do formułowania pochlebstw najwyższego gatunku. Z mojego skromnego dorobku wybrał niektóre dokonania w sposób bardzo przemyślany. Była tu mianowicie mowa o osiągnięciach, których przytaczanie sprawia mi największą radość. Wśród wymienionych w Laudacji dokonań znalazły się te wczesne, jeszcze z czasów studiów na Państwowym Uniwersytecie Moskiewskim, kiedy to w roku 1960 udało mi się opracować *Quicksort*, jak również moja późniejsza działalność, czyli 40 lat pracy (do roku 2010) na stanowisku wykładowcy i dyrektora cyklu edycji letniej szkoły w Marktoberdorfie koło Monachium.

Władysław wykazał się również ogromnym taktem, gdyż przez grzeczność **nie** opowiedział paru rzeczy o mnie. Aby zatem wykreować w Państwa oczach swój należycie wyważony wizerunek, muszę uzupełnić Laudację, dodając do niej moje szczere przyznanie się do kilku spośród popełnionych przeze mnie błędów, a także wspomnieć o niepowodzeniach, które spotkały mnie w moim ciekawym życiu. Jak już wcześniej mówiłem, język programowania ALGOL W nie otrzymał stosownej akceptacji. Jednak moje najbardziej znane niepowodzenie miało miejsce w połowie lat sześćdziesiątych ubiegłego wieku, kiedy pełniłem funkcję głównego inżyniera projektu w zakresie wdrożenia systemu operacyjnego dla komputera Elliott 503, który został opracowany jako kompatybilny następca Elliotta 803. Po ponad trzydziestu osobolatach prac projekt nie został zakończony w terminie (który zmieniał się kilkakrotnie). W końcu okazało się, że w ogóle nie udało się nam dotrzymać

żadnego terminu! Po prostu z powodu przeciążenia pamięci program był stanowczo zbyt wolny.

Jednak była i jaśniejsza strona tego wydarzenia – to niepowodzenie było impulsem do rozwoju mojego wieloletniego zainteresowania zjawiskiem współbieżności, które odnosi się do jednoczesnego działania różnych części tego samego programu. Punktem kulminacyjnym moich badań nad współbieżnością było opracowanie teorii Komunikujących się Procesów Sekwencyjnych (CSP) oraz jej zastosowanie w firmie INMOS przy projektowaniu architektury ich transputera. Od tamtej pory aż do chwili obecnej pracuję nad dalszym rozwojem tej teorii oraz przeniesieniem jej na poziom bardziej ogólny.

Władysław, życzliwie wobec mnie, pominął także moją inną, nawet bardziej dotkliwą i kosztowną porażkę. Określam ją mianem mojego błędu za miliard dolarów. Na początku lat sześćdziesiątych spotkałem się z językiem programowania Simula, doskonale zaprojektowanym i wdrożonym w Oslo przez Ole-Johana Dahla. Elementem składowym języka Simula była jedna z początkowych wersji programowania znanego obecnie pod nazwą programowania obiektowego. Stało się to inspiracją do włączenia obiektowości do koncepcji języka ALGOL W, o którym była już wcześniej mowa. Użyteczną, a zarazem nową cechą języka stało się moje odkrycie, iż prosta weryfikacja typu, przeprowadzona w całości przed uruchomieniem działania programu, może zagwarantować bezpieczeństwo i strukturalną spójność programu obiektowego. Rozwiązanie to znalazło zastosowanie w powszechnie podziwianej wersji 1967 języka Simula, w języku ALGOL 68 oraz w popularnych obecnie językach C++ i Java.

Niestety, miałem nieszczęście wszystko zniweczyć poprzez utrzymanie w moim projekcie cechy znanej pod nazwą wskaźnika pustego. Wskaźnik pusty jest bardzo użyteczny i oznacza brak informacji – na przykład informacji dotyczącej żony w przypadku mężczyzny stanu wolnego. Jednak, co jest znacznie gorsze, wskaźnik pusty często bywa bardzo groźny. Bywa przyczyną niezliczonych błędów w programowaniu, mających nieprzewidywalne konsekwencje, co powoduje, iż ich wykrywanie i korygowanie jest niezwykle trudne.

Liczne błędy tego typu pozostają w oprogramowaniu dostarczanym klientom, a wówczas to oni ponoszą konsekwencje tych błędów. Niektóre spośród nich mogą narazić komputer na wrogie i przestępcze ataki, w tym również takie, które w skali gospodarki światowej generują

koszty rzędu miliardów dolarów rocznie. Dotyczy to nie tylko szkód bezpośrednich, lecz również kosztów opracowania środków zaradczych, ochrony, zapobiegawczych i naprawczych. W ciągu pięćdziesięciu lat od chwili wystąpienia tego błędu ogólne koszty z pewnością można szacować na kwotę miliardów dolarów.

Lista moich osiągnięć wymienionych przez Władysława kończy się w latach osiemdziesiątych ubiegłego stulecia, i to jest z jego strony bardzo uprzejme. Dobrze się składa, mam bowiem sposobność opowiedzenia Państwu o mojej dalszej działalności. W końcu lat osiemdziesiątych współpracowałem z Robinem Milnerem z Edynburga i Janem Bergstrą z Amsterdamu nad projektem o nazwie CONCUR, finansowanym ze środków Unii Europejskiej w ramach Akcji Badań Podstawowych. Celem naszych badań było ujednolicenie trzech różnych teorii współbieżności, zaproponowanych przez trzech głównych badaczy. Z przykrością stwierdzam, że nie udało się nam tego dokonać. Trzej główni badacze poczynili ogromne postępy w zakresie zaawansowania i zastosowań swoich teorii, lecz nigdy nie osiągnęliśmy porozumienia co do jednej, jednolitej teorii współbieżności.

I znów porażka stała się inspiracją i ukierunkowała moje dalsze badania. Moja uwaga skoncentrowała się na technikach matematycznych i logicznych, które pozwalałyby na skuteczniejsze ujednolicenie jeszcze szerszej gamy teorii programowania. Chciałem włączyć tu znane programy sekwencyjne, jak również te odnoszące się do działania współbieżnego. W tamtym okresie publikacje wręcz roiły się od samowystarczalnych teorii programowania, ich liczba nie ograniczała się jedynie do trzech teorii włączonych do projektu CONCUR. Ta mnogość teorii jest naturalna, gdy mamy do czynienia z nową i jeszcze niedojrzałą dziedziną wiedzy. Jednak jednym z długoterminowych celów jest unifikacja różnych teorii, początkowo powstałych z myślą o zastosowaniu do różnych zjawisk naturalnych. Powszechnie znanym przykładem jest tu teoria grawitacji Newtona, która łączy zjawiska ruchu planet na niebie ze spadającym na ziemię jabłkiem.

Pomyślne połączenie teorii wykazuje, że kilka istniejących teorii można matematycznie wyprowadzić z jednej ujednoliconej teorii. Te istniejące teorie stają się silniejsze w procesie unifikacji, ponieważ każda z nich uzyskuje dodatkowe potwierdzenie w postaci naukowych świadectw, zgromadzonych w ramach pozostałych teorii. Funkcjonujące do momentu unifikacji teorie zwykle pozostają równie użyteczne jak przed

procesem unifikacji, ponieważ każda z nich w sposób przemyślany bazuje na szczególnych właściwościach i służy konkretnym celom w nieco węższym spektrum zastosowań. Unifikacja odgrywa istotną rolę dla dalszego postępu inżynierii. Dopóki występują kontrowersje natury naukowej pomiędzy badaczami, inżynierowie i ich pracodawcy słusznie wstrzymują się od wyboru i zastosowania jednej z konkurujących teorii. Unifikacja rozwiązuje problem kontrowersji naukowych. Daje ona inżynierom możliwość wyboru najbardziej stosownej spośród ujednolicony teorii, a nawet jej dalszego rozwinięcia poprzez wyspecjalizowanie w kierunku konkretnych warunków danego projektu.

Ten właśnie sposób rozumowania przywiódł mnie i współpracującego ze mną na polu naukowym He Jifenga do opracowania programu badań, co z kolei doprowadziło do opublikowania w 1998 roku książki pod tytułem „Jednolite teorie programowania" („Unifying Theories of Programming"). Nasze prace badawcze prowadzone były ściśle w oparciu o rachunek relacyjny, skodyfikowany i wyjaśniony przez wielkiego polskiego logika Alfreda Tarskiego. Tarski rozpoczął naukę tutaj, na Uniwersytecie Warszawskim w roku 1918 na Wydziale Biologii, lecz szczęśliwie dla logiki, dla filozofii i dla informatyki przeniósł się na studia w zakresie logiki w słynnej szkole Stanisława Leśniewskiego, Jana Łukasiewicza i Wacława Sierpińskiego. Alfred Tarski został zaproszony do wygłoszenia wykładu na Uniwersytecie Harvarda we wrześniu 1939 roku, szczęśliwie zatem uniknął okupacji niemieckiej w Polsce. W 1942 roku przeniósł się do Berkeley, gdzie pracował do końca życia. W 1941 roku opublikowano jego pracę „On the Calculus of Relations" („O rachunku relacyjnym") w International Journal of Symbolic Logic. Prawdziwym zbiegiem okoliczności był fakt, iż zarówno He Jifeng jak ja przeczytaliśmy ten artykuł zanim się poznaliśmy, obaj podziwialiśmy jego elegancję, żałując jednak, że nie ma on zastosowania w informatyce. Potrzebowaliśmy dziesięciu lat, aby jednak znaleźć sposób jego zastosowania, a kolejne dziesięć lat zajęły nam intensywne badania nad opracowaniem szczegółów.

W roku 1999 osiągnąłem wiek emerytalny na Uniwersytecie Oksfordzkim. Na szczęście Roger Needham zaoferował mi zatrudnienie w laboratorium badawczym Microsoft, niedawno utworzonym w Cambridge w Anglii. Po dwudziestu latach pracy na stanowisku profesora informatyki na Wydziale Matematyki Uniwersytetu Oksfordzkiego przeprowadzka do Cambridge była dla mnie i dla mojej żony naprawdę

trudna. Wszystko jednak dobrze się ułożyło, jesteśmy zadowoleni z naszego nowego miejsca zamieszkania. Od strony zawodowej ta zmiana okazała się wspaniała. Naukowcy w Microsoft mają pełną swobodę i wsparcie, gdy chodzi o wybór przedmiotu swoich badań, a przy tym mogą zapomnieć o takich trudach życia akademickiego jak uczestnictwo w posiedzeniach komitetów, wystawianie ocen z egzaminów lub składanie wniosków o granty na badania.

Pozwolę sobie jeszcze na chwilę powrócić do roku 1968, kiedy zamieniłem pracę w przemyśle, czyli w firmie Elliott Brothers, na działalność akademicką w Belfaście. Moim celem było prowadzenie badań ukierunkowanych na weryfikację programów. Uświadomiłem sobie, że praktyczne zastosowanie tych badań w przemyśle raczej nie nastąpi przed moim przejściem na emeryturę za jakieś 30 lat. Wniosek ten skłonił mnie do przejścia ze sfery produkcyjnej do akademickiej. Natomiast w momencie, gdy rozpocząłem współpracę z Microsoftem w 1999 roku rzeczywiście przekonałem się, że mój pogląd wyrażony w 1968 roku był słuszny. Żadne z wyników mojej trzydziestoletniej pracy badawczej nad weryfikacją nie znalazły zastosowania w praktyce. Zauważyłem co prawda, że niektóre twierdzenia dotyczące programów były sporadycznie stosowane przez niektóre zespoły programistów Microsoft jako pomoc przy testowaniu programu, lecz prawie nigdy nie w celu zamierzonym, czyli w celu weryfikacji programu.

Jednak od czasu, gdy rozpocząłem współpracę z Microsoft, stan rzeczy znacznie się zmienił. Miałem wyjątkowe szczęście być naocznym świadkiem nadzwyczajnego rozpowszechnienia technologii weryfikacji programów (znanej również jako metody formalne) w codziennej praktyce opracowywania oprogramowania w siedzibie Microsoft w Redmond. Jest to bezpośredni efekt wyliczeń kosztów generowanych przez błędy w oprogramowaniu, które w pewnym okresie szacowane były nawet na miliardy dolarów rocznie. Niezależny raport sporządzony przez Amerykański Departament Handlu zawiera dane, zgodnie z którymi koszty, których można uniknąć w samych Stanach Zjednoczonych, szacowane są na 25 do 60 miliardów dolarów.

Dlatego też Dział Badań Microsoft włączył się do programu, mającego na celu opracowanie i zastosowanie algorytmów logicznych i matematycznych, wykrywających wiele rodzajów błędów, które pojawiają się w programach komputerowych, w tym błędu wskaźnika pustego, o którym wcześniej wspomniałem. Te zaawansowane algorytmy

zostały włączone do Narzędzi Automatyzacji Projektowania i Konstrukcji Oprogramowania, przypominających narzędzia już istniejące w innych gałęziach inżynierii. Znalazły one powszechne zastosowanie w procesie testowania pierwszych wersji nowo opracowywanego oprogramowania. Wykorzystano je do skanowania i sprawdzania wielu milionów wierszy w kodach Microsoft przed ich dostarczeniem; wpłynęło to na znaczne podniesienie poziomu bezpieczeństwa programów, które potem trafiają na rynek. Obecnie w całej firmie Microsoft pracują setki inżynierów i naukowców, których specjalnością jest weryfikacja.

Wciąż jednak wiele ciekawych problemów czeka na rozwiązanie. Każdy projekt dotyczący weryfikacji wiąże się z działaniami specjalistów, przeanalizowaniem specyficznych właściwości programu poddawanego analizie wraz z jego wymaganym rozwiązaniem. Konieczny jest tu zintegrowany zestaw narzędzi, tworzących zbiór Narzędzi Automatyzacji Projektowania, które mogą być stosowane rutynowo przez znaczną większość profesjonalnych programistów w celu minimalizacji kosztów błędu. W przypadkach, które wymagają weryfikacji na wyższych poziomach, profesjonalny inżynier do spraw weryfikacji powinien bez trudu dostosować parametry narzędzia do konkretnych warunków. Rzecz jasna, budowa zestawu narzędzi o tak szerokim zastosowaniu jest ogromnym projektem, zapewne o skali porównywalnej do budowy akceleratora cząstek przez fizyków lub pozaziemskiego teleskopu przez astronomów. Nie ma wątpliwości co do tego, że jego znaczenie będzie równie doniosłe, jak opracowanie standardowego kompilatora dla C lub C++, który obecnie analizuje miliony wierszy kodu – odpowiada to wielu tysiącom osobolat pracy. Tak czy inaczej, marzyłem o tym, że długoletni program fundamentalnych badań akademickich mógłby dostarczyć najważniejszych koncepcji, narzędzi oraz badań przypadków, które przyczyniłyby się do urzeczywistnienia się tych idei.

Po raz pierwszy przedstawiłem ten pomysł w swoim zaproszonym wystąpieniu na międzynarodowej konferencji (ETAPS), która odbyła się w Warszawie w 2003 roku. Wyraziłem wówczas nadzieję, że tak ogromny projekt mógłby być uznany za Wielkie Wyzwanie Informatyczne dwudziestego pierwszego wieku. Podobnie jak nieco wcześniej zakończony Projekt Poznania Ludzkiego Genomu, wymagałby on długoletniej ścisłej współpracy wiodących badaczy z całego świata, reprezentujących tę dziedzinę. W składzie zespołu badawczego tego projektu powinni się znaleźć przedstawiciele trzech grup badaczy: teoretycy, zapewniający

solidne podstawy i uzasadnienie teoretyczne dla prototypowych narzędzi, konstruktorzy narzędzi, wdrażający teorię poprzez udoskonalone algorytmy oraz eksperymentatorzy, testujący narzędzia w sposób niezależny w rzeczywistych lub realistycznych sytuacjach, dotyczących weryfikacji programów. W okresie ostatnich pięciu lat ponownie skoncentrowałem się na kwestiach unifikacji teorii programowania. Mam nadzieję, że ujednolicenie będzie pomocne w procesie projektowania zintegrowanego zestawu narzędzi, gdzie każde narzędzie może być bezpiecznie opracowane w oparciu o dowolnie wybraną z ujednoliconych teorii, najlepiej odpowiadającą obranym celom.

Mam wrażenie, że ostatnio w moich pracach nastąpił przełom. Wykorzystałem mianowicie potencjał, zdolność do adaptacji oraz elegancję dziedziny matematyki zwanej algebrą. Sformułowałem zbiór paru tuzinów algebraicznych Praw Programowania, których złożoność nie przekracza stopnia złożoności praw arytmetycznych nauczanych od pokoleń w szkołach. Dotyczą one zarówno programów sekwencyjnych, jak i współbieżnych. Wykazałem, że algebra z powodzeniem unifikuje Logikę Hoare'a (na której opiera się weryfikacja programów sekwencyjnych) z semantyką operacyjną Robina Milnera (która określa prawidłowe wykonanie procesów współbieżnych). Jestem przekonany, że wreszcie udało mi się osiągnąć cele, które zakładał projekt CONCUR sprzed dwudziestu pięciu lat. Ponadto udało mi się znaleźć pociechę dla siebie i moich współpracowników na wypadek opóźniającego się zakończenia (lub nawet rozpoczęcia) projektu, który tu opisałem. Pocieszam się mianowicie, kontemplując piękno, będące nieodłączną częścią Praw oraz prostotę opracowanych w oparciu o nie dowodów.

Na zakończenie swojego wystąpienia chciałbym oddać hołd pamięci dwóch wspaniałych naukowców z dziedziny informatyki, Robina Milnera oraz Edsgera Dijkstry. Obaj wnieśli długotrwały wkład w rozwój nauki. Byli również moimi bliskimi przyjaciółmi, a przyjaźń ta obejmowała również całe nasze rodziny. Od chwili ich poznania, w mojej działalności badawczej z podziwem wzorowałem się na Edsgerze, a rywalizowałem i jednocześnie podziwiałem Robina. Ponieważ nigdy nie mieliśmy wspólnych publikacji, a jedynie wybiórczo powoływaliśmy się nawzajem na swoje prace, myślę, że teraz mam najlepszą okazję, aby złożyć im obu moje najgłębsze wyrazy wdzięczności, zarówno w sferze naukowej, jak i osobistej.

Przed uroczystością. Od lewej: prof. Władysław M. Turski, prof. Jan Madey, Sir Charles Antony Richard Hoare, prof. Marcin Pałys, prof. Andrzej Tarlecki

Before the ceremony. From left to right: Prof. Władysław M. Turski, Prof. Jan Madey, Sir Charles Antony Richard Hoare, Prof. Marcin Pałys, Prof. Andrzej Tarlecki

Otwarcie uroczystości. Od lewej: prof. Andrzej Tarlecki, Sir Charles Antony Richard Hoare, prof. Marcin Pałys – rektor UW, prof. Władysław M. Turski

Opening of the ceremony. From left to right: Prof. Andrzej Tarlecki, Sir Charles Antony Richard Hoare, Prof. Marcin Pałys – Rector of the University of Warsaw, Prof. Władysław M. Turski

Laudację wygłasza
prof. Władysław M. Turski

Professor Władysław M. Turski
delivers the Laudatio

Przemawia dziekan Wydziału
Matematyki, Informatyki i Mechaniki,
prof. Andrzej Tarlecki

Dean of the Faculty of Mathematics,
Informatics and Mechanics,
Professor Andrzej Tarlecki,
delivers his address

Wręczenie dyplomu doktora *honoris causa* Uniwersytetu
Warszawskiego Sir Charlesowi Antony'emu Richardowi Hoare'owi

Sir Charles Antony Richard Hoare is handed the diploma of doctor
honoris causa of the University of Warsaw

Wykład wygłasza
Sir Charles Antony Richard Hoare,
doktor *honoris causa* Uniwersytetu
Warszawskiego

Sir Charles Antony Richard Hoare,
doctor *honoris causa*, delivers his address

Prof. Marcin Pałys, rektor UW, wręcza Medal Uniwersytetu Warszawskiego Sir Charlesowi Antony'emu Richardowi Hoare'owi

Professor Marcin Pałys, Rector of the University of Warsaw,
hands Sir Charles Antony Richard Hoare a Medal
of the University of Warsaw

Sir Charles Antony Richard Hoare

Doctor *Honoris Causa* of the University of Warsaw

The Ceremony of Awarding

Warsaw, November 28th, 2012

Programme of the Ceremony

- *National anthem*
- Opening of the ceremony by Rector of the University of Warsaw, Professor Marcin Pałys
- Laudatio delivered by Professor Władysław M. Turski
- Speech by the Dean of the Faculty of Mathematics, Informatics and Mechanics, Professor Andrzej Tarlecki
- Presenting the diploma
- *Gaude Mater Polonia*
- Speech by Sir Charles Antony Richard Hoare
- Closing the ceremony by Rector of the University of Warsaw, Professor Marcin Pałys
- *Gaudeamus igitur*

RESOLUTION № 11
OF THE SENATE OF THE UNIVERSITY OF WARSAW

of October 17th, 2012

on approving the reviews and awarding a degree
of doctor *honoris causa* of the University of Warsaw

The Senate of the University of Warsaw in accordance with the Higher Education Act of 27 July 2005 (Dz. U. 2012, item 572 and 742), in connection with the Statute of the University of Warsaw, §8, having accepted the reviews submitted by Professor Paweł Idziak from Jagiellonian University in Krakow, Professor Henryk Krawczyk from Gdansk University of Technology, Professor Antoni Mazurkiewicz from Polish Academy of Sciences, presenting achievements of Sir Charles Antony Richard Hoare, the nominee for the degree of doctor *honoris causa* of the University of Warsaw. The Senate of the University of Warsaw has adopted this resolution:

§1

Sir Charles Antony Richard Hoare is hereby granted the degree of doctor *honoris causa* of the University of Warsaw.

§2

The Resolution shall enter into force as of the date of its adoption.

The President of the Senate
of the University of Warsaw,
Rector
Professor Marcin Pałys

Professor Paweł Idziak
Jagiellonian University

Opinion to the Senate of the University of Warsaw on conferring Honorary Doctorate to Sir Charles Antony Richard Hoare

Sir Charles Antony Richard Hoare is one of the most distinguished creators of computer science. He was constructing the very bases of the science both in the times of its turbulent beginnings, as well as throughout its further development, through outlining the ideas which indicated directions in the development of contemporary computer science.

He received thorough and exceptionally comprehensive education. He was born in 1934 in Colombo (on former Ceylon), Sri Lanka, later receiving secondary level education at Dragon School in Oxford and King's School in Canterbury. In Oxford he completed his studies, majoring in Latin, Greek and Philosophy. The latter subject served as the gateway to modern logics, whose ideas, language and methodology he applied in his scientific research. After graduation in 1956 he was called up to the Royal Navy, where he learnt Russian. Prior to applying the new skills, in 1958 he returned to Merton College in Oxford, where he studied mathematical statistics and familiarised himself with programming in Mercury Autocode language. He spent one year as a doctoral student at the Lomonosow University in Moscow, where under the supervision of Andrei Kolmogorov he continued his studies on statistics, which he used in automated translation between natural languages. This research contributed to raising his awareness of the crucial role of quick sorting, at that time he came up with the first ideas, which were turned into his later concept of *Quicksort* algorithm.

On his return to Great Britain in 1959 he started his 8-year-long experience of employment at Elliott Brothers company. One of the projects carried out by the team under his supervision assumed the development of a compiler for the revolutionary at that time ALGOL 60

programming language, for the Elliott 803 computers. Unlike his predecessors, ALGOL 60 allowed for the usage of recursion. In view of its capacity, Hoare fully completed the ideas of *Quicksort,* which has become the most frequently applied sorting algorithm. There have been numerous versions, mutations and improvements of the algorithm, whose role turned out to be absolutely invaluable. This achievement alone could have contributed to Hoare's reputation of most renowned researcher both in scientific community and among programmers. However, it was overshadowed by his further accomplishments of crucial significance. The experience gained at Elliott Brothers of elaborating compilers and operating systems made him sensitive to the key role played by programming languages. What initially seemed to be a failure over the process of developing the Mark II Elliott 503 operating system, inspired his subsequent thorough research on concurrency in developing programs and processes. The employment at Elliott Brothers also contributed to developing new ideas within the underlying mathematical and practical approach to programming languages and constructing high quality software.

The role of Hoare's contribution into the methodology of designing programming languages becomes apparent, when we realize the scale of costs incurred for software development. In practice, too often we have seen the cases when software has proved to be of bad quality, for which partially the programming languages and their compilers are to be blamed. Suffice it to say that annual losses generated by errors in software amount to billions of dollars. Hoare repeatedly raised the issue of simplicity and elegance, indispensable to avoid those mistakes and monitoring them. He put in an excellent way in his paper entitled *Hints on programming language design* of 1973.

Prior to this, however, in 1968 he was appointed the Head of the Chair of Computer Science at Queen's University of Belfast, where – free from the responsibility of programming and disappointed by commonly approved practice of documenting programs – he commenced efforts on semantics of programming languages. Inspired by Robert Floyd's findings, which indicate the way in which extending circumstances and invariables in a block description of a program allow for demonstrating in a strict mathematical way that the program meets its specification, Hoare extended the incorporation of mathematics in semantic research. In his

paper entitled *An axiomatic basis for computer programming* he proposed abandoning block schemes in favour of a logical system, also known as Hoare's Logic, which allows for purely formal reasoning related to programs and their functioning. This step was of crucial importance – correctness is expressed and verified during the process of program design rather than exclusively by means of inspection upon its completion.

In 1977 Hoare became the Professor at the University of Oxford, where he resumed his research on concurrency. As a result, the principles of cooperation between concurrent processes communicating with each other were formulated. These principles were specified in the calculus of processes proposed by him, which was first described in the paper *Communicating Sequential Processes,* and further developed in a book of the same title. The language related to the calculus mentioned allows for interactions between the processes exclusively in precisely pre-defined situations. This provides full control over the construction of structural concurrent programs. The research on this language developed extensively, very soon have been applied in engineering practice. The language, which initially served as a programming tool, was later developed into its new version – as *occam* programming language it shaped the architecture of transputers.

It is virtually impossible to enumerate all the accomplishments of Sir Charles Antony Richard Hoare. Also after his retirement in1999 he has been very active. Let us only mention that out of his 200 scientific papers, over 60 have been published after 1999.

For his achievements he has been granted the most prestigious awards in his discipline of computer science. As early as in 1980 he was granted the Medal and Turing Award, commonly regarded as equivalent to the Nobel Prize in computer science. In 2005 Hoare was awarded equally prestigious Kyoto Prize, and in 2011 he was presented the John von Neumann Medal.

In 2000 he was conferred British Knighthood by the Queen.

Since 1982 he has held the membership of the Royal Society, since 2005 of the Royal Academy of Engineering. He has been conferred the title of honorary doctor by several universities.

Sir Charles Antony Richard Hoare has had an immense impact on the development of computer science. In addition to his record in

theoretical and applied science, his merits include educating new generations of researchers. He has supervised 23 doctors, most of whom have been educating their own doctoral students. The group of his scientific "descendants" includes over 230 people working on 5 continents.

He has also shaped research direction in Poland, in particular, at the University of Warsaw. He offered internships to the employees of the Institute of Informatics of the University. That opportunity resulted in receiving a strong stimulus towards the development of research on operating systems and concurrent processes. Also, many studies carried out at the University of Warsaw on algorithm construction, programming methods, program specification and verification or Hoare Logic have been inspired by that scientific cooperation.

Exceptionally high scientific merits of Sir Charles Antony Richard Hoare, as well as his impact on the development of computer science at the University of Warsaw contribute to my conviction that he fully deserves being invited to the exquisite selection of honorary doctors of the largest university in Poland – the University of Warsaw.

Professor Henryk Krawczyk
Faculty of Electronics, Telecommunication and Informatics
Gdańsk University of Technology

Opinion to the Senate of the University of Warsaw on conferring Honorary Doctorate to Sir Charles Antony Richard Hoare

Sir Charles Antony Richard Hoare was born in Colombo (now Sri Lanka) in the family of a British civil servant. He was granted his Bachelor Degree at the University of Oxford (Merton College) in1956. For another year he read the course in statistics at the same college, upon its completion he served in the Royal Navy. Over the following two years he studied in USSR at the State University in Moscow in Kolmogorov school. In 1960 he was employed at Elliott Brothers Ltd., a computer manufacturing company, where he implemented the translator of Algol 60 language and developed a number of computational algorithms. In 1968 he became the professor of computer science at Queen's University of Belfast, and in 1977 he returned to the University of Oxford, setting up a new computer laboratory and later the Faculty of Computer Science. Currently he is a retired professor at that university; he also cooperates with Microsoft in Cambridge. Over his scientific career he has been granted a number of significant awards and distinctions:

- membership of several scientific institutions, including Fellow of the Royal Society (1982), Academia Europea (1989) and Royal Academy of Engineering (2005),
- honorary doctorates: Queen's University of Belfast (1987), University of Bath (1993), Athens University of Economics and Business (2007) and British Knighthood conferred by the Queen (2000),
- prestigious awards and medals: ACM Turing Award (1980), Kyoto Prize (2000), IEEE John von Neuman Medal (2011).

He is the author or co-author of four books of fundamental meaning in computer science:

1. O.-J. Dahl, E.W. Dijkstra and C.A.R. Hoare. *Structured Programming*. Academic Press, 1972.

2. C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall International, Series in Computer Science, 1985.
3. C.A.R. Hoare, M.J.C. Gordon. *Mechanised Reasoning and Hardware Design*. Prentice Hall 1992.
4. C.A.R. Hoare, He Jifeng. *Unifying Theories of Programming*. Prentice Hall International, Series in Computer Science, 1998.

His scientific achievements include approximately 200 publications, with the number of citations exceeding 20,000, which is an outstanding achievement. It is worth mentioning that his scientific record has become the subject of several publications, including *Oral History of Sir Antony Hoare* by Jonathan Bowen (2006), or *An Interview with C.A.R. Hoare* by Shustek and Lon. The latter work was published by the *Communications of the ACM* (2009). Indeed, C.A.R. Hoare is a renowned computer scientist, famous for inventing *QuickSort*, *Hoare Logic* applied in the verification of program correctness and for a formal language CSP (*Communicating Sequential Processes*), applied in specification of concurrent processes (including "the problem of dining philosophers"). He has also gained popularity as the author of OCCAM language for parallel programming. For any student of computer science in any country this knowledge is indispensable and fundamental.

His view on software in computer systems has played a crucial role. He indicates the existence of two methods of program construction. One is based on adopting certain simplifying assumptions so that the implementation in the computer system is flawless; in the second method all the complexity of the system is taken into account, however, error analysis is not performed. C.A.R. Hoare claims that the first method is significantly more difficult to be applied than the second one; this view seems to be confirmed by the contemporary software engineering.

The achievements of Professor C.A.R. Hoare serve as real milestones in the development of computer science. His books have been translated into many languages and are used as basic academic textbooks in computer science. They provide a set of principles of good and proper programming, where the issues of concurrency of many processes, their deadlock or starvation are addressed. Similar problems also occur in case of complex distributed systems, which jointly and remotely carry out a certain set of computational tasks. Hence, Professor Hoare's research contributes significantly to the process of construction of modern computer systems.

It is worth mentioning that the employees of the University of Warsaw established relations with C.A.R. Hoare as early as in 1968-1970 when he attended computer science conferences held in Poland. Later those relations evolved into scientific cooperation on the application of CSP Hoare methodology in specifications of programming systems, including operating systems, improving programming methods as well as programs verification. This cooperation resulted in interesting publications and doctoral dissertations, as well as a visit paid by the UW associates at the University of Oxford. Also, a number of scholarships were granted to young programmers from Warsaw who attended the prestigious NATO summer school in software engineering, headed by Master Hoare himself. It is worth bearing in mind that contemporary school of computer science, developed at the University of Warsaw by Professor Jan Madey, would not be as successful in programming as it is now without thorough acquisition and subsequent development of the research held by C.A.R. Hoare.

In view of the above statements, I would like to express my support to the motion on conferring the title and distinction of an honorary doctor to a distinguished researcher from Oxford, Sir Charles Antony Richard Hoare. I regard the motion as entirely justified. The presence of such a scientific personality among other holders of this prestigious academic distinction will certainly increase the fame of the University of Warsaw – a widely renowned institution of higher education.

Professor Antoni Mazurkiewicz
Institute of Computer Science
Polish Academy of Sciences

Opinion to the Senate of the University of Warsaw on conferring Honorary Doctorate to Sir Charles Antony Richard Hoare

Sir Charles Antony Richard Hoare was born in 1934 in Colombo, at that time a British colony on the Indian Ocean. In computer science community he is known as Tony Hoare. He graduated from the University of Oxford in 1956. After his two-year service in the Royal Navy he completed an internship at the state University in Moscow, where he joined the research team headed by Andrei Kolmogorov. He focused on the issue of computer translation (therefore, he is one of very few or perhaps even the only British computer scientist with good command of Russian). After his return to Great Britain he became the professor at Queen's University in Belfast, later continuing his professorship at the University of Oxford, where he currently is a retired professor.

Tony Hoare is a person symbolising a whole era of computer science development, from its beginning until present time. His activity has been devoted, in particular, to the issues related to software, which play a key role in every day application of computer technology. We can claim that it is the merit of the scientists working at the beginning of that era, with Professor Hoare playing a special role among them, that today we can see a rapid development and widespread applicability of digital technology in contemporary world. By his accomplishments Tony Hoare has contributed immensely to the existence of computer science in its present form.

Over many years of his activity Hoare has been awarded many prestigious distinctions. The most significant include granting him in 1980 by the Association for Computing Machinery the Alan Turing Award for his crucial scientific achievements in computer science; this Award is often regarded as equivalent to the Nobel Prize in this

discipline. Another distinction of similar rank and significance was von Neumann Medal granted him in 2011. Tony Hoare was also conferred the title of honorary doctor of the University in Belfast (North Ireland), in Athens (Greece) and Bath (England). In 2000 he was conferred British Knighthood by the Queen for his service in education and computer science.

Regarding Tony Hoare's scientific achievements, we should recognize his fundamental role in developing the concept of communicating sequential processes. This concept, which refers directly to the intuitions related to distributed processing, led to developing formal tools applied to describe and study such systems. As it often happens science, next generations and extensions of one idea can be found in seemingly distant disciplines, obviously, in a modified form, adjusted to current and updated needs. This was also the case of CSP (Communicating Sequential Processes) concept and the ideas it comprises. New generations of computer scientists were inspired by the ideas conveyed within CSP in their research and achievements. We can certainly state that the concept of CSP has influenced the discipline of distributed processing, whose shape would otherwise be different and less sophisticated.

It is worth emphasizing that Tony Hoare – like other computer scientists of his generation – has managed to combine his passion for research with programming technique. His contribution to the development of this technique, often underestimated by scientific community, proved to be invaluable among programmers, with its great role played in developing programs and computer systems. The concepts invented by Tony Hoare are widely exploited by programmers, who perhaps are not aware of Hoare's contribution as the person who has facilitated the process of developing and increased the effectiveness of applying programming tools. His "Quicksort" algorithm, which has improved one of the basic elements of programming constructions, exceeds the concept of technical facilities, and is a real achievement of contemporary algorithmics. In this construction he applied the "divide and conquer" technique (divide et impera), which soon became one of the canons of modern algorithmics. Also, in programming technique itself, programming concepts such as "Case of" construction introduced by Hoare, which substantially simplifies and facilitates the development

of computer software, is one of his great achievements. Many programmers, who were professionally active in the 70s, including myself, are sincerely grateful to Tony Hoare for facilitating their work through inventing that construction.

In the last century linguistic tools enabling human-computer communication have developed dynamically. With the advancement of technical means applied in computer science, those tools were developing and modifying; however, many ideas embedded in modern programming languages and systems or hidden in their codes were invented by Tony Hoare. His participation in the discussions on Algol – the prototype of Pascal – played a crucial role as a contribution to the concept of modern programming languages.

One of basic problems in computer science, of universal character, is proving the correctness of the constructions under development. In particular, it is very important to verify the correctness of the results expected of a program under construction (or an existing one), by means of disciplined and formal reasoning. This requires one formal approach and unified methodology towards two different systems; static system, well established in mathematics, using the apparatus of formal logics, and dynamic system, which uses algorithms and combines them, thus creating a formal description of program operations. Among numerous propositions and achievements in this respect, from the works of Floyd and Dijkstra, through algorithmic and dynamic logics, to contemporary systems of proving the correctness of programs, Hoare Logic holds a significant position as one of the formal systems allowing for demonstrating the correctness of fulfilled condition through program outcomes (i.e. the description of its results), according to precisely specified rules of program operation, as well as its initial condition (i.e. a description of a program's initial data). This logic, apart from the list of rules derived from the construction of the program, comprises inference rules, which allow for proving the correctness of more complex constructions by means of putting together simpler fragments. Tony Hoare's achievements in this area, as the author of the logical system known as Hoare Logic, brought him recognition of computer science community.

I have had a pleasure to meet Tony Hoare in person on several occasions. From our encounters I have gained an impression of him

being a modest person, full of friendliness, kindness, openness, positive criticism, directed also towards his own achievements. Our meetings have always offered an opportunity of developing new ideas and concepts, as well as of discussing the existing ones.

In conclusion, I would like to state that the University of Warsaw, despite its stable renown among global universities, will gain more prestige thanks to having Tony Hoare among its Honorary Doctors. He represents modern science, he is highly valued in the international scientific community, which seems to be proved by the facts mentioned above. He is also a scientist who develops our knowledge and skills, contributing as well to the development computer science as a tool of contemporary civilization. I believe that Tony Hoare fully deserves the distinction of Honorary Doctor of the University of Warsaw. I strongly support this idea, with absolutely no reservations.

The ceremony of awarding Sir Charles Antony Richard Hoare with a degree of doctor *honoris causa* of the University of Warsaw

Prof. dr hab. Władysław M. Turski

LAUDATIO

I nearly met our Distinguished Nominee in person in 1960. I was just completing my studies at the Mechanico-Mathematical Faculty of Moscow University which then hosted – under the aegis of British-Soviet scientific cooperation – a young aspiring statistician, C.A.R.Hoare, working under guidance of A.N. Kolmogorov. Had I indeed met Tony then, I could have picked up his brilliant invention some years ahead of actually reading about it in learned journals: the *Quicksort* algorithm for sorting the elements of a set – this has been for over half a century an indispensable part of every computer scientist's education, an irreplaceable practical tool and a firm quality standard for that part of every software system which deals with sorting.

The path that led the twenty-six years old Tony Hoare to the discovery of Q*uicksort* is neither easy, nor straightforward, his curriculum vitae up to this point shows no portents foreshadowing a brilliant scientific career.

Born in Ceylon, from his childhood Tony retains almost Kiplingesque memories of school outings to the jungle, wild elephants and tigers. Young Tony wanted then become a writer. After the War, his family settled in England and Tony went to the King's School in Canterbury, the oldest one in the Realm, with traditions stretching back to the VI century; there he found his sanctuary in the Library, developing a taste for philosophy and the acid humour of G.B.Shaw. Linguistic precision and elegance remain distinguishing features of Hoare's writing.

In due course, following family tradition, Tony enrolled in Merton College, Oxford, where he read "Lit Hum" which included subjects such as Latin, Greek, ancient history, linguistics and literature, as well as classical and contemporary philosophy. Upon graduation he was drafted into National Service which he served in the Royal Navy, learning Russian intensively. In 1958 Tony returned to Oxford where he undertook education to become a professional statistician which he continued in Moscow. It was then that he learned to program using the

Manchester Autocode for the Ferranti Mercury computer and wrote his first programs.

In those years, machine (computer) translation between natural languages, such as from Russian to English, was a subject of great interest. During his visit to Moscow, Hoare devoted most of his time to it.

Large collections of data, such as comprehensive dictionaries, were then stored on magnetic tapes, which could be read (and written) in one direction only. If, having accessed an item, one needed to look up another element, stored ahead of it, it was necessary to rewind the tape to its beginning and only then wind it on to the desired position; needless to say, this was a very time-consuming procedure. Obviously then, it made great sense to sort (rearrange) the collection of items requiring look up in such a way that their sequence "agreed" with the direction in which the tape could be read. There was a huge practical demand for an efficient method (algorithm) of sorting. *Quicksort*, the algorithm invented by Hoare is in this respect optimal but it has another remarkable property, viz. it is recursive. Roughly speaking, execution of a recursive algorithm refers to execution of another copy of the same algorithm; obviously, in a properly designed (correct) recursive algorithm this cannot go ad infinitum: the execution must finally stop.

Programmers did not like recursive algorithms very much. Firstly, with the intellectual tools then at their disposal it was quite difficult to establish correctness of a recursive algorithm. Secondly, the programming languages at their disposal did not support recursion, i.e. it was impossible to express a recursive algorithm in a simple way. Hoare addressed both of these issues.

During his visit to Moscow, a British electronics firm, Elliott Brothers, staged an exhibition of their scientific computers and employed Tony as a translator. Upon his return to the UK, they offered him a permanent position as a programmer with a special yearly bonus of £100 for his competence in Russian. Elliott computers were then quite popular in Europe, several of them were successfully installed in Poland. When the firm decided to enter the market with a stronger machine, they were determined to equip it with a new, more powerful programming language. Instead of inventing his own, which was then a popular sport, Hoare proposed to furnish new computers with compilers for ALGOL-60, a programming language developed by an international group of scientists

(IFIP WG 2.1), today considered as the coryphaei of programming methodology. ALGOL-60 fully supported recursion. Elliott Brothers' management accepted this proposal; a special, four-person strong programming team headed by Tony was established and charged with the task of implementing ALGOL-60 on the new computers. Among members of this team there was one person with previous experience of compiler writing, Jill Pym; she soon became Tony's wife. The ALGOL-60 project for new Elliott Brothers computers was a success, one of very few from dozens then undertaken worldwide, and Hoare was invited to join WG 2.1.

At the beginning of his professional career, C.A.R. Hoare spent eight years in an industrial firm, learning the pluses and minuses of a commercial environment. For his entire life he has retained an understanding of the demands placed on programmers by this kind of work, in particular, an unshakable conviction that the honest work of an industrial informatician requires creative engagement. In the world, where there is plenty of antagonism between informatics "theoretical" and "practical", "industrial" and "academic", such an attitude and its constant application in one's own work and in the work of supervised teams, deserves the highest respect.

Hoare was very active in WG 2.1, and in close collaboration with N. Wirth developed a successor to ALGOL-60 that included a number of important improvements; unfortunately their proposal failed to gain majority approval. Wirth continued this line of research on his own, published first ALGOL-W and then Pascal, the programming language that for nearly two decades dominated the world scene, both in education and in practical projects, despite a growing (in some environments) tendency to construct monstrously large programming languages (such as ALGOL-68 or Ada) intended for programming so-called large systems (such as operating systems).

Hoare strongly opposed this tendency: "...in spite of all the work that has been expanded on the design of general and special purpose programming languages, the role they play in the design and implementation of a large system is at best minimal" he said in his lecture at a four-day conference in Jabłonna (1970) devoted to this problem.

ALGOL-60 and its closely-related successors (Pascal, Simula) had a fully formal syntax which allowed for fully automatic parsing of

programs written in such languages. Their semantics, however, albeit very precise, was not formally defined. Considering the great variety of computer structures then in use, it was possible that a single program executed on different computers would yield results differing in some details, viz. that eluded informal definition of programming language semantics. Note that more or less successful attempts to formalise programming language semantics continue to this day with a slender chance of finding a commonly accepted solution.

If a program yields different results, which one is to be accepted as correct? Or, conversely, if we know which results are correct, which program is to be viewed as correct?

In 1968 Hoare, already an internationally recognised scientist, accepted the computer science chair at the Queen's University, Belfast, and turned his attention to the problem of program correctness. In one form or another, this problem was addressed by the inventor of the stored program computer, J. von Neumann, and pioneers of informatics, such as J. McCarthy and R. Floyd.

Hoare's idea consisted in treating the meaning of a program statement (i.e. its semantics) as a limit on the arbitrariness of the results of its execution: instead of trying to define absolutely precisely a unique result, he fixed the boundaries within which *any* result will be accepted as correct. Formally, this is expressed by the so-called *Hoare's triple,* $P\{Q\}R$, interpreted as follows: statement Q executed in a state (of the computation) satisfying condition P will terminate in a state satisfying condition R. If, for each elementary statement of a programming language, a suitable triple is taken as an axiom, and sound rules are introduced for evaluating triples characterising compound statements (similar to rules of reasoning in classical logic), then – given a program – one can evaluate a "super triple" embracing the entire program, viz. determine the condition that must be satisfied upon termination of a program started in a state satisfying the initial condition of the "super triple".

The preceding (grossly simplified) description of *Hoare's logic* is the basis for contemporary programming methodology. Over the years there have been many additions to its simple concepts and many attempts to enrich the calculus of correctness that it implies, but there is no textbook nor monograph on the subject of program correctness that would neglect

Hoare's contribution; his paper "An axiomatic basis for computer programming" (published in 1969), which announces the principles of his approach, has been referred to nearly 20,000 times.

In 1977 Prof. Hoare moved to Oxford, first as the head of the Programming Research Group, and soon after as the first ever Oxford chair of computing science. He continued investigations of program correctness notions which led to the publication of the very important monograph "Structured Programming", written with E.W. Dijkstra and O.-J. Dahl, and to many significant achievements by his disciples and collaborators, whom he skilfully assembled. For example, Jean-Raymond Abrial developed in Oxford the language Z for description and analysis of computer systems which was subsequently applied by IBM, the computer giant of that epoch, to the improvement of CICS, one of the most profitable (and most complex!) pieces of software on the market; this work earned IBM and the authors' Laboratory the Queen's Award for Technology.

Soon another important subject attracted Prof. Hoare's attention: the cooperation of independent processors. The stimulus came again from the computer industry. More and more complex computing systems contained separate multiple processors cooperating in the execution of a common task. Perhaps the simplest example is that of a processor performing computations with another which controls an output device (a screen or printer). From time to time, the former generates a portion of information, which the latter is to deliver to the outside world, perhaps after converting it into a more readable form. Intellectual tools created for investigating correctness of programs for single processes did not fit easily for software written for multiprocessor systems, including – the ever more important – operating systems which were rapidly becoming fundamental parts of software.

The first solution proposed by Prof. Hoare (to a large extent influenced by and developed with P-B. Hansen, author of the operating system RC-4000, installed, among others, in Azoty, Puławy, Poland) was the *monitor,* a software construct delivering well-defined services to the outside world in a highly disciplined and formalised manner. Using monitors, which agreed quite well with the concepts of Simula, a language developed by O.-J. Dahl, one could apply a logical calculus to a rather large class of multiprocessor software issues. Unfortunately this was quite

a difficult process, and monitors themselves, being rather coarse-grained constructs, required non-trivial proofs of correctness (of implementation).

The next solution proposed by Hoare was a lot simpler and more fundamental: process cooperation was restricted to exchange of messages. The instruction repertoire of a process was enlarged to include two message passing constructs: one construct serves to send a message, another – to accept it. Actual exchange occurs when the receiving process arrives at the accept instruction and the sending process is ready to execute the send instruction. If one of the communicating processes arrives at a message exchange instruction when the other process is not yet ready to perform the matching instruction, the former process waits for the latter. This solution – very much like a simple hardware scheme of two processors linked by a single wire – was published in the paper "Communicating sequential processes" (1978) and fully presented in the monograph bearing the same title (1985).

Up to now the calculi based on this principle had provided the basic tools for investigating and designing multiprocessing systems both in software and in hardware. For example, based on CSP, occam, a language, designed in Oxford, was used by the firm INMOS to verify the design of the second generation transputers (microprocessors) which included floating point arithmetic. It is estimated that application of occam to this task considerably shortened the expected "testing" time of the new product and, consequently, the new generation of transputers could be marketed a full year sooner than planned. For this achievement, INMOS and the Oxford Computing Lab won an ealier Queen's Award for Technology.

Upon retirement from Oxford University in 1999, Prof. Hoare accepted the post of Principal Researcher in the British Division of Microsoft Research in Cambridge, where he continues investigations of program correctness and cooperation of processes with shared memory. One could say that a productive and beautiful professional life has run full circle: from Elliott Brothers to Microsoft. But a graduate of Lit-Hum knows very well it is impossible to enter the same river twice. In the last half-century informatics has changed from a promising sapling to one of the most important caryatids of our civilization.

This change was brought about by many firms and entrepreneurs, many inventors and scientists. Tony Hoare belongs to the narrowest

group of the most outstanding ones. In recognition of this fact, he was elected to both British academies: the Royal Society and the Royal Academy of Engineering; the British Computer Society awarded him the rarest rank of Distinguished Fellow. The greatness of his achievements has been acknowledged on both sides of Pacific: he received the Turing Award and Medal (considered the Nobel Prize for computing) from the American Association for Computing Machinery and the Japanese Kyoto Prize (*nota bene* awarded also to another Doctor h.c. of our University: Andrzej Wajda). Several universities made him an honorary Doctor, and Queen Elizabeth II knighted him for services to education. For he has not only established in Oxford one of the world's leading centres of computing science education and research, which among others hosted several our colleagues from Warsaw, but also for over 20 years directed the famous Marktoberdorf Summer School, where leading computer scientists from all over the world share their ideas with promising young graduates. And, although the School was financially supported by NATO, even during the years of the Cold War, students from University of Warsaw were accepted there and their attendance was subsidised.

By granting C.A.R Hoare the degree of Doctor *Honoris Causa* we express our deep appreciation of his life achievements, honour a great scientist and leader of research, and thank him for decades of goodwill and friendship.

To a new Doctor *Honoris Causa* of the University of Warsaw we wish many more years of fruitful work, and to ourselves – further fruitful cooperation with him.

(Translated from Polish by the author with kind assistance
of Joanne and Cliff B. Jones)

Address by the Dean of the Faculty of Mathematics, Informatics and Mechanics of the University of Warsaw, Professor Andrzej Tarlecki

Ladies and Gentlemen,

It is my honour and privilege to speak here on behalf of the Faculty of Mathematics, Informatics and Mechanics of the University of Warsaw at this wonderful ceremony of awarding a doctorate *honoris causa* to Professor Tony Hoare. We meet here at the end of the year declared as the Year of Informatics at the University of Warsaw and of the month marked by the anniversary of the foundation of our University. For me though this is also a very special occasion because this is the first ever doctorate *honoris causa* awarded by the University of Warsaw to a computer scientist. In a way, this is not a surprise: Computer Science is still a relatively young discipline. One good aspect of this is that we can meet and award those who shaped it from the very beginning – and Professor Hoare is undoubtedly one of them.

It would be proper now to list Professor Hoare's numerous impressive scientific achievements and the ideas he contributed to developments in Computer Science over more than half a century. In no way, however, can I match the *laudatio* by Professor Turski, who himself experienced and took active part in those developments as they were happening. Let me pause here to express my sincere thanks to Professor Turski for his role as the promoter in awarding this degree and for the admirable *laudatio*. Let me also thank the reviewers, Professor Mazurkiewicz, Professor Krawczyk and Professor Idziak, for their exhaustive, excellent opinions about the impressive achievements of Professor Hoare. As I said, I cannot match them, so instead, let me try to say a few words about... myself.

I started learning programming at the University of Warsaw in 1975. From the very beginning the stress was on *structured programming*, in line with the ideas laid out in the famous "*Structured Programming*" book that Professor Hoare co-authored. We were taught to design and develop our programs formulating all the needed requirements, assertions, and loop invariants – and as much as possible, arguments for program correctness. The latter soon took for us the form of Hoare logic. Its study was one of my early scientific fascinations, and I still recall heated discussions of the logic's (in)completeness. A few years later, in my PhD, I presented a *language of specified programs*, from one point of view just proposing an extension of Hoare logic to further programming constructs. And incidentally, a running example in the thesis was a version of Professor Hoare's *quicksort* algorithm. Before that, my MSc thesis was on *algebras for input/output semantics* – an abstract algebraic version of axiomatic semantics for programs, as introduced for Pascal in an article co-authored by Professor Hoare. After my PhD, I spent some time in Edinburgh, with one of the first trips from there taking me to Oxford. I spent most of the visit in discussions with the Z group who worked within PRG directed by Professor Hoare, but I also recall a nice chat with Professor Hoare on that occasion. He spoke mostly about CSP, which I admit was not a focus of my interest at the time. Nevertheless, some time later I devoted quite a bit of effort trying to squeeze additional constructs into the emerging VDM standard to allow the expected semantics of CSP to be expressed. This was a side topic for me in the late eighties; a major fascination then, lasting until today, concerned the so-called behavioural interpretation of program specifications – a close descendant of the ideas on data representation that Professor Hoare put forward in the early seventies. In the early nineties, I was honoured by an invitation to lecture at the Marktoberdorf Summer School, co-directed by Professor Hoare. He totally surprised me there by asking a series of inquisitive questions on category theory, which I failed to answer satisfactorily, I'm afraid. I learnt then that this was to support his new work on theories of programming. Some years later one highlight of my organisational activities was ETAPS 2003, a congress we ran here in Warsaw. In turn, one of its highlights was a plenary lecture by Professor Hoare on the great challenge of the *verifying compiler* – a topic which stimulated some of the recent activities on program verification in my group here.

No, I did not have to bend anything in this story, which I believe is rather typical for computer scientists of my generation. Professor Hoare's results and ideas formed the basis of what we, computer scientists at the University of Warsaw and all over the world, learnt as students and worked on as scientists. Let me add that we especially value his work for its serious, responsible attitude to the issues of program construction, with correctness and quality taken as the indispensable primary goal.

This doctorate *honoris causa* of the University of Warsaw, the highest academic degree our community can offer, is a token of our recognition and gratitude for lessons we learnt from Professor Hoare's work and for numerous contributions to our discipline that Professor Hoare have made over the years, from the *quicksort* algorithm of more than half a century ago to the verifying compiler of the future.

Thank you!

Address
by Doctor *Honoris Causa*
Sir Charles Antony Richard Hoare

It is with immense pleasure that I accept the high distinction which you confer on me by the Award of an Honorary Doctorate of the University of Warsaw. I would like to express my heartfelt gratitude to the University, and to all of you who join me here today as witnesses of this magnificent ceremony.

It is also a great pleasure for me to be again in Poland, where I have enjoyed so many visits in the past. My first visit was in 1960. At the end of my academic year at Moscow State University, I was hired for two weeks as a translator at an exhibition of British Scientific instruments held in Moscow. One of the exhibits was an 803 computer, manufactured by Elliott Brothers of London, who had brought it to the exhibition in a small van. At the end of the exhibition, the computer was sold to a Soviet Institute (a secret one, of course). I was offered a free lift back to England in the empty returning van. On my arrival in England, I accepted an offer for my first job, as a programmer in Elliott Brothers.

On that trip back from Moscow, I vividly remember my crossing of the border from USSR into Poland. The contrast was extraordinary: a change from an uncultivated and untended and unlovely roadside to a prosperous agricultural landscape. And the people were different too: so friendly and so cheerful that I could almost believe that they were free. During the next five years, I made the same trip several times both ways through Poland, in a van carrying my Company's computers to Moscow for a summer exhibition and subsequent sale. We often stopped the night in Warsaw, and once stayed at the Bristol hotel, where I have been staying on this trip.

I also have many fond memories of subsequent scientific meetings, workshops and conferences held in Poland. In October 1966 I flew to Warsaw for a fateful meeting of an international Working Group

(WG 2.1), engaged in the design of a programming language that was later promulgated as ALGOL 68. I had the pleasure of first meeting other Polish members of the group, Antoni Mazurkiewicz and Wlad Turski, and observers Jan Madey and Andrzej Salwicki. I am delighted to see them again today in this audience. But in other respects, I did not enjoy the ALGOL meeting. It was the meeting which finally rejected a language designed by Niklaus Wirth and myself, which had been previously commissioned by the Working Group itself, and which had been published in June 1966 as a 'Contribution to the Development of ALGOL'. Our language was later known as Algol W.

My most memorable trip to Poland was much more enjoyable. It was to a conference held in Jablonna in 1970, on 'Efficient Production of Large Programs.' Wlad Turski, my promoter at today's ceremony, was the chairman. The official language of the conference was English, but there were only three native speakers of English in attendance. Most of the audience spoke and understood Russian much better than English. So Wlad undertook to do all the necessary translation in both directions. He was brilliant – my knowledge of Russian was sufficient to judge that. He would listen to a paragraph of a presentation in one language and then translate it faultlessly into the other language. Of course, neither language was his own native language.

On one occasion he listened to a long paragraph in Russian, and turned to the English- speaking group to translate the whole paragraph into perfect... Russian. He had no idea that he had actually performed two translations in his head. The first was from Russian to English, while he was listening to the Russian speaker. The second was while he was speaking to the English group, and then he translated from English back again to Russian. He had no idea of what he had done. He was surprised when he was told, and he joined the general laughter. Finally, he retranslated the whole paragraph (for the third time) into perfect English.

It was at that conference that I uttered my most famous epigram. I expressed the view that inside every large program there is a small program trying to get out. This was an adaptation of a common saying in those days about fat persons and thin persons trying to get out. I knew that my epigram was famous many years later, when I saw it reprinted on the side of a coffee mug. For a scientist, coffee mug publication is a great distinction, and leads to the widest fame.

Now it is my great pleasure to meet Wlad Turski again, and listen to his most flattering Laudation of my life's work. Of course, he has commendably indulged only in the very best kind of flattery. He has shared with you a well-judged selection of my modest achievements, exactly the ones which I am most happy to be reminded of. His account started with the lucky invention of Quicksort at Moscow State University in 1960, and it ended with my contributions, over a period of forty years (ending in 2010), as a lecturer and Director of nearly twenty summer schools, in the well-known series at Marktoberdorf, near Munich.

Wlad has also been kind in what he has **not** told you about me. To create a balanced picture, I feel that I must complement his laudation by a frank confession of some of the many mistakes and failures of an interesting life. I have already told you of the failure to obtain acceptance of the ALGOL W programming language. But my best known failure was in the mid nineteen-sixties, when I was the Chief Engineer of a project to implement an operating system for the Elliott 503 computer, which had been developed as a compatible successor for the Elliott 803. After more than thirty man-years of implementation effort, the project failed to meet its promised delivery date (several times). In the end, it failed to be deliverable at all! As a result of memory thrashing, it was just ridiculously slow.

Turning to the brighter side, this failure was the impetus for my life-long interest in the phenomenon of concurrency, which calls for simultaneous execution of different parts of the same program. My research on concurrency culminated in the development of the theory of Communicating Sequential Processes, and its application by INMOS in the design of the architecture of their transputer. I have continued since then to develop and generalise the theory, right up to the present day.

Wlad has also been kind to me in forgetting about an even more costly failure. I call it my billion-dollar mistake. In the early 1960's, I encountered the programming language Simula, brilliantly designed and implemented in Oslo by Ole-Johan Dahl. Simula included one of the very first versions of what is now known as object-oriented programming. It inspired the inclusion of object orientation in the language ALGOL W, mentioned earlier. A useful and novel feature of the language was my discovery that simple type-checking, conducted wholly before a program starts execution, could guarantee the security and structural soundness of

an object-oriented program. This solution was also adopted in the widely admired 1967 version of Simula, in ALGOL 68, and in the currently more popular languages C++ and Java.

Unfortunately I then went on to spoil everything, by retention in my design of a feature, known as the null pointer. The null-pointer is very useful to represent the absence of information, for example, information about the wife of a man who is not married. Far worse than this, the null pointer is also notoriously dangerous. It is the cause of innumerable programming errors, leading to unpredictable consequences, which make them difficult to detect and correct.

Many of these errors remain in software that is delivered to customers, and it is they who have to suffer the consequences. Some of the errors can expose a computer to malicious or fraudulent attack, of the kind that still costs the world economy many billions of dollars per year. This includes not only direct damage, but also the cost of precautions, protection, prevention and cure. Over the period of fifty years since the mistake was first made, the sum of costs must surely be measured in billions of dollars.

Wlad's account of the significant events in my life ends in the 1980s. He therefore leaves to me the opportunity to tell you something about what I have been doing since then. In the later 1980s, I worked with Robin Milner from Edinburgh and Jan Bergstra from Amsterdam on a project called CONCUR, funded by the European Community as a Basic Research Action. Its stated goal was to unify three different theories of concurrency, as propounded by the three principal investigators. I'm afraid we failed to meet this goal. The three principal investigators made great progress in the advancement and application of their own theories; but we could never bring ourselves to concur (that is, to agree) on a single unifying theory of concurrency.

Again, the failure inspired the direction of my subsequent research. I turned my attention to mathematical and logical techniques that would lead to a more successful unification of a still wider range of theories of programming. I wanted to include familiar sequential programs as well as programs that called for concurrent execution. At that time, there was in the published literature a proliferation of self-sufficient theories of programming, many more than just the three theories of the CONCUR project. This proliferation of theories is natural when a branch of science

is young and immature. But one of the main longer term goals of science is the unification of diverse theories, originally developed for application to different natural phenomena. A well-known example is Newton's theory of gravitation, which unified the phenomena of planetary motion in the skies with the fall of an apple to the Earth.

Successful unification shows that a number of existing theories are mathematically derivable from a single unifying theory. The existing theories are actually strengthened by the unification, because each of them is now supported by all the scientific evidence collected for all the other theories. The existing theories often remain just as useful as before the unification, because each of them cleverly exploits the particular characteristics, and serves the particular needs, of a narrower range of applications. Unification is also essential to the progress of engineering. As long as there is scientific controversy between scientists, practical engineers and their employers will be rightly reluctant to use any of the competing theories. Unification resolves the scientific controversy. It empowers the engineer to select the most suitable one of the unified theories, and even extend it by further specialisation to the particular circumstances of a current project.

That was the reasoning that led me and my long-term research colleague He Jifeng to devote ourselves to a programme of research, leading to publication in 1998 of a book entitled 'Unifying Theories of Programming'. Our research was firmly based on the relational calculus, as codified and expounded by the great Polish logician Alfred Tarski. Tarski entered this University of Warsaw in 1918 to read biology; but fortunately for logic and for Philosophy and for Computer Science, he was deflected to study logic in the famous school of Stanislaw Lesniewski and Jan Lukasiewicz and Waclaw Sierpinski. He was invited to give a talk in September 1939 at Harvard University, and so was fortunate in escaping the German occupation of Poland. Indeed, he moved in 1942 to Berkeley, where he worked the rest of his life. In 1941 he published his influential article 'On the Calculus of Relations' in the international Journal of Symbolic Logic. Coincidentally, both He Jifeng and I had read this article before we met, and admired its elegance; but regretted that we could think of no way of using it in Computer Science. It took us ten years to realise how it could be used, and now it took another ten years of intensive research to work out the details.

In 1999, I reached retirement age at Oxford University. Fortunately, I was invited by Roger Needham to join the staff of a new Microsoft Research Laboratory, recently established in Cambridge, England. After twenty two years as a Professor of Computing in the Mathematics Faculty of Oxford University, it was a wrench for me and my wife to move to Cambridge. But it turned out well, and we have been very happy in our new location. For me professionally, it has been wonderful. Microsoft researchers have freedom and support in the selection of their own topic of research, without any of the academic chores of attending committee meetings, marking examinations, or applying for research grants.

Let me hark back to 1968, when I moved from industrial employment at Elliott Brothers to a Professorship in Belfast. My purpose was to conduct research leading to the verification of programs. I realised that the practical application of this research in Industry was unlikely to occur until after my retirement date, some thirty years ahead; and this prediction was why I decided to move from industrial employment to academic life. When I joined Microsoft in 1999, I found indeed that my 1968 prediction was correct. None of the results of my thirty years' research in verification were in practical application. True, I found program assertions being used sporadically by some Microsoft programming teams as an aid to program testing, but almost never for their original purpose of program verification.

But since I joined Microsoft, the scene has changed dramatically. I have been exceptionally fortunate in witnessing at first hand an extraordinary spread of the use of program verification technology (otherwise known as formal methods) in the daily practice of software development at Microsoft in Redmond. This was a direct result of a calculation of the cost of programming errors, which was at the time measured in billions of dollars every year. An independent report by the US Department of Commerce came up with an estimate of 25 to 60 billion dollars of avoidable costs for the United States alone.

The Research Division of Microsoft therefore engaged in a programme of development and deployment of logical and mathematical algorithms, capable of detecting many of the kinds of error that are known to occur in computer programs, including the null reference error that I described earlier. These advanced algorithms were incorporated into Design Automation tools for Software Engineering, similar to those that are now

available to other branches of engineering. They have been widely applied in testing early versions of newly developed software. They have scanned and checked many millions of lines of Microsoft code prior to delivery, and significantly raised the level of security of software eventually released to the market. There are now hundreds of specialist verification engineers and scientists employed throughout Microsoft.

But many challenging problems remain. Each verification project requires specialist effort, exploiting the specific characteristic of the program to be analysed, and the particular kind of solution required. What is needed is an integrated suite of tools forming a Design Automation toolset, that can be used routinely by the great majority of professional programmers to save the cost of error. Where higher levels of verification are required, a professional verification engineer should find it easy to adjust the parameters of the tool to specific circumstances. It is clear that the construction of such a broadly applicable toolset is an immense project, perhaps on the same scale as construction of a particle accelerator by physicists, or an extra-terrestrial telescope by astronomers. It will certainly be as great as the construction of a standard modern compiler for C or C++ , which now runs to some ten million lines of code – many thousands of man-years of effort. Nevertheless, I dreamed that a long-term programme of fundamental academic research cold provide the essential concepts and tools and case studies leading to the achievement of this ideals.

I first presented this goal in an invited keynote address that I gave to an international conference (ETAPS) held in Warsaw in 2003. I suggested that such an immense project should be regarded as a Grand Challenge for Computer Science in the twenty-first century. Like the just completed Human Genome Project, it would require long-term close collaboration of international leading researchers in the field. The project team will have to include at least three classes of researcher: theorists to provide a sound underpinning for prototype tools; tool-builders implementing the theory by improved algorithms; and experimentalists testing the tools independently on real or realistic case studies in the verification of programs. In the last five years, I have turned my attention again to the unification of theories of programming. It is my hope that the unification will assist in the design of an integrated toolset, where each tool can safely use whichever one of the unified theories that is most appropriate for its purposes.

Recently, I think that I have made a breakthrough. I have exploited the power and adaptability and elegance of the branch of mathematics known as algebra. I have formulated a collection of a couple of dozen algebraic Laws of Programming, no more complicated than the laws of arithmetic taught to generations of schoolchildren. The laws apply equally to sequential and to concurrent programs. I have shown that the algebra successfully unifies the Hoare Logic (on which verification of sequential programs is based) with Robin Milner's operational semantics (which specifies correct execution of concurrent processes). I believe that this at last fulfils the objectives of the CONCUR project of twenty five years ago. Furthermore, I have found a consolation for myself and others for the delay in the completion (and even the start) of the grand challenge project that I have described. I console myself by contemplation of the inherent beauty of the Laws, and the inherent simplicity of the proofs based upon them.

I would like to end my address with a tribute to the memory of two great Computer Scientists, Robin Milner and Edsger Dijkstra. They have both made enormous long-term contributions to the progress of science. They were my close personal friends, and our friendship was shared by our families. In all my research since I met them, I have been an admiring follower of Edsger, and an admiring rival of Robin. Since we never wrote a joint paper, and only made selective reference to each others' work, this is a good occasion to recall my deep scientific and personal debt to both of them.