Pokroky matematiky, fyziky a astronomie

C. W. H. Lam
Jak spolehlivý je počítačový důkaz?

Pokroky matematiky, fyziky a astronomie, Vol. 36 (1991), No. 4, 209--216

Persistent URL: http://dml.cz/dmlcz/138404

A v Anglii jde o skutečně radikální změnu, neboť zejména školy odpovídající svou povahou našemu gymnáziu jsou v posledních čtyřech třídách vysoce specializovány.

K tomu je třeba poznamenat, že také francouzský systém má velmi silnou složku školské výuky zaměřenou na všeobecné vzdělání.

Neměly by proto pro nás být všeobecně vzdělávací průprava a její široké uplatnění otázkami, na něž by se měla teprve hledat odpověď. Řešení, jak se zdá, je nalezeno a připravuje cestu člověku, který se v budoucnosti bude setkávat s rostoucím množstvím problémů, jež bude muset zvládat.

Nyní půjde o to, jak daleko má u nás všeobecné vzdělání jít, tj. do jaké šíře a do jakého věku mladé generace. Pro nás, matematiky a fyziky, bude velmi účelné posoudit, jakou úlohu by měla mít matematika a fyzika ve všeobecném vzdělání a v tomto směru bude nutné vypracovat konkrétní návrhy.

Další materiály ze sjezdů otiskneme v příštím čísle.

Redakce

Jak spolehlivý je počítačový důkaz?

C. W. H. Lam

Clement Lam získal hodnost Ph. D. v oboru matematika na Kalifornském technologickém institutu. Studium na tomto institutu ukončil v roce 1974 jako žák Herberta Rysera. Jeho prvním působištěm byla univerzita v Calgary, odkud po roce odešel na katedru informatiky na montrealské univerzitě Concordia. V současné době je profesorem na Concordii. Jeho vědecký zájem se soustředil na kombinatoriku, zvláště v souvislosti s použitím počítačů při řešení kombinatorických problémů. Spolu s L. H. Thielem a S. Swierczem dokázal v roce 1988, že neexistuje konečná projektivní rovina řádu 10. Tento důkaz si vyžádal několik tisíc hodin strojového času obřího počítače CRAY-1A.

„Je matematický důkaz důkazem, když ho nikdo nemůže zkontrolovat?", tázal se nedávno titulek jednoho článku [2] v *New York Times*. Článek informoval o výsledcích výzkumu, který jsme právě dokončili. Cílem našeho výzkumu bylo nalezení konečné projektivní roviny řádu 10. Výpočty byly provedeny pomocí počítače a byly mimořádně rozsáhlé. Počet prozkoumaných případů byl vyšší než 10^{14}. Toto číslo je tak vysoké, že není přirozeně v lidských silách výsledky zkontrolovat. Stručně jsem se této otázky dotkl v rozhovoru s reportérem M. W. Brownem, který článek připravoval. Neřekl jsem mu však, že u superpočítače CRAY-1A, který prováděl většinu našich výpočtů,

C. W. H. Lam: *How Reliable Is a Computer-Based Proof?* The Mathematical Intelligencer 12, No. 1, pp. 8–12. Přeložila Helena Nešetřilová.

byly hlášeny nezjištěné chyby a že jedna taková chyba připadá zhruba na tisíc hodin chodu počítače. Naše výpočty byly tak rozsáhlé, že spotřebovaly několik tisíc hodin strojového času. Musíme tedy počítat s tím, že je v nich několik chyb. Představte si výše uvedený novinový titulek, který by se ptal: „Je matematický důkaz důkazem, když ho nikdo nemůže zkontrolovat a je v něm několik chyb?"

Snažím se slovu „důkaz" vyhýbat, já osobně dávám přednost výrazu „počítačový výsledek". Je však mnoho matematiků, kteří jsou ochotni přijmout takový výsledek jako důkaz! Abych věci uvedl na pravou míru, zahrnul jsem do článku [6], který referoval o našem výsledku, také odstavec, který se zabýval správností výsledku. Jeho závěrem bylo konstatování, že pravděpodobnost současného výskytu několika náhodných hardwarových chyb, které by způsobily, že počítač hledanou rovinu nenajde, je velice malá. Všimněte si, že tvrzení „výsledek je správný" není *absolutní*, ale pouze *skoro jisté*. To je pro počítačové výsledky typické.

Když tak o tom přemýšlím, možná, že bych se slovu „důkaz" přece jen vyhýbat neměl. Vždyť to není poprvé, kdy počítač sehrál při „důkazu" matematické věty důležitou úlohu. Jeden z důležitých případů, které naší práci předcházely, byl důkaz problému čtyř barev [1]. Mezi matematiky stále přibývá těch, kteří si možnosti počítačů uvědomují. To je jistě důvod k tomu, abychom si uvědomili také hranice jejich možností. Počítače jsou pro nás v prvé řadě nový nástroj, který musíme používat s rozvahou. Použití počítače má závažné důsledky pro to, co lze považovat za přijatelný důkaz. Chtěl bych v tomto článku nastínit některé svoje myšlenky, které se této otázky týkají.

Možná si myslíte, že se vás tenhle problém netýká a že zajímá pouze odborníky, jako jsem já, kteří se zabývají otázkami kombinatorického prohledávání. Řeknete si, že je to riziko povolání. Uvažte však symbolické manipulační programy jako je MACSYMA, REDUCE nebo MAPLE. Tyto programy podstatně snižují časovou náročnost, a tím nám dávají možnost řešit i takové úlohy, které by nebylo možné ručním výpočtem zvládnout. Předpokládejme, že jen jediný krok v nějakém důkazu je založen na použití některého z těchto programů. I v takovém případě si musíme položit otázku, jsou-li počítačové výsledky správné. Článek [3] uvádí například případ, kdy jeden z uvedených programů nenašel pro danou soustavu rovnic jedno ze dvou možných řešení. Můj kolega z univerzity, na které pracuji, požádal před časem program MACSYMA o faktorizaci jistého polynomu. Program odpověděl, že polynom je irreducibilní. Můj kolega výsledku nevěřil, zkusil proto zadat úlohu znovu. Tentokrát byla odpověď správná a výsledkem byl hledaný rozklad. Zavinil chybu program MACSYMA nebo se spletl on sám při vkládání polynomu? Odpověď pro nás není důležitá. Důležitá je tu skutečnost, že výsledek výpočtu může být chybný bez ohledu na důvody, které chybu způsobily. Výsledek tedy nelze považovat za absolutní ani v tom případě, že počítač sehrál při jeho důkazu jenom podružnou roli.

Abychom mohli nepřítele porazit, musíme mu rozumět. Ve druhém odstavci se proto podíváme na nejběžnější výpočetní chyby. Ve třetím odstavci popíši některé metody, které jsem použil, abych zvýšil spolehlivost vypočtených výsledků. Ve čtvrtém odstavci nastíním svoji představu o tom, jak hodnotit počítačové důkazy. Pátý odstavec je věnován úloze nezávislé verifikace počítačových důkazů. Šestý odstavec obsahuje několik stručných závěrečných poznámek.

Charakteristika výpočetních chyb

Výpočetní chyby lze zhruba rozdělit do dvou skupin: chyby zaviněné člověkem a hardwarové chyby. Lidské chyby jsou nejčastější. Tyto chyby lze dále rozdělit na chyby uživatele, které jsou pod kontrolou uživatele, a na systémové chyby, které jsou mimo kontrolu uživatele.

Jedním zdrojem uživatelských chyb je vstup. Zadáme-li počítači chybná data, nemůžeme očekávat správné výsledky. Znáte jistě přísloví: jak se do lesa volá, tak se z lesa ozývá. V každém případě se vám vyplatí, když vstupní data dvakrát zkontrolujete.

Další typ uživatelských chyb jsou chyby programu v případech, kdy uživatel používá vlastní program. Programování je proces, při kterém počítači zadáme posloupnost příkazů, které má provést. Počítač tyto příkazy provede bez ohledu na to, mají-li smysl. Zadáme-li počítači chybné nebo neúplné příkazy, pak výsledky, které dostaneme, mohou být v některých případech správné a v některých chybné. Většina programátorů dobře ví, že počítačové programy se musí nejprve testovat. Znamená to, že se program nechá proběhnout se zkušebními daty a kontroluje se, dává-li program očekávané výsledky. Postupu, při kterém hledáme a opravujeme chyby programu, říkáme ladění. I ladění má však svoje meze. Lidová moudrost praví: testováním programu můžeme prokázat jenom to, že v něm chyby jsou, ale nemůžeme prokázat, že v něm nejsou.

Pro chyby programu je typické to, že jsou většinou opakovatelné. Mám tím na mysli to, že při stejných vstupních datech dostaneme stejně chybné výsledky. Výjimkou z tohoto pravidla je programování počítačů s paralelní architekturou, kde chyby způsobené časovou závislostí paralelních procesů většinou opakovatelné nejsou. U opakovatelných chyb nemá nové spuštění programu význam, protože výsledky budou vždycky stejné, bez ohledu na to, je-li program správný nebo chybný.

Všechny ostatní chyby programu, které nezavinil uživatel sám, zahrneme do skupiny systémových chyb. Používáme-li jakýkoli počítač, potřebujeme obvykle operační systém, jako je například MSDOS nebo UNIX. Operační systém je balík programů, které napsal někdo jiný, aby nám používání počítače usnadnil. Nesmíme zapomenout ani na *kompilátory*. To jsou programy, které překládají uživatelské programy z vyšších programovacích jazyků (jako je třeba Pascal) do strojového kódu. Některým kompilátorům se říká *optimalizační*, protože programy, které na strojové úrovni vytvářejí, pracují rychleji. Než se takový systémový program dostane do rukou uživatele, měl by být napřed odladěn. Přesto se málokdy stává, že by takový program žádnou chybu neobsahoval. Já osobně jsem měl potíže s několika optimalizačními kompilátory. Jako příklad mohu uvést rané verze kompilátoru Pascalu na VAX pod operačním systémem VMS. Při použití optimalizační verze kompilátoru se chybně překládaly některé operace se zhuštěnými*) množinami. V našich programech pro projektivní roviny jsme pro úsporu paměti a strojového času se zhuštěnými množinami pracovali často. Museli jsme se však soustavně vyhýbat použití optimalizační verze kompilátoru, i když to bylo na úkor rychlosti programu. Měli jsme dost starostí se správností našich vlastních programů a nemohli jsme si dovolit starat se ještě o správnost překladu.

*) v orig. „packed sets" — pozn. překl.

Je vůbec možné odstranit z programu všechny chyby? Programátoři už v tomto směru dávno rezignovali a u dlouhých a komplikovaných programů považují malý počet neobjevených chyb za přijatelný. Říkají: „Jediné programy, ve kterých chyby nejsou, jsou triviální programy." To, je-li vůbec možné správnost programu prokázat, je stále ještě předmětem výzkumu.

Hardwarové chyby však naproti tomu pod kontrolou člověka nejsou. Tyto chyby se objevují náhodně a mají obvykle katastrofální důsledky, jako je zastavení počítače nebo zničení dat na disketě. Zřídka se stává, že by prošly bez povšimnutí a z hlediska správnosti výsledku nepředstavují obvykle žádný problém, protože buď žádný výsledek nedostaneme, nebo je na první pohled chybný. Největším nebezpečím jsou však chyby, které uniknou pozornosti, protože změní výsledek nenápadným způsobem. Běžnou chybou je například náhodná změna bitů v paměti počítače. U většiny velkých počítačů se proto ke snížení tohoto rizika používají paměti s korekčními obvody. Přesto však existují chybové stavy paměti, které se korekčním obvodům zdají správné. Naštěstí jsou takové chyby velice vzácné a pravděpodobnost, že ovlivní výsledek, je malá. Není však vždy zanedbatelná, jak ukazuje například frekvence výskytu takových chyb u počítače CRAY-1A.

Shrňme nyní charakteristiky obou uvedených typů počítačových chyb. Nejčastější jsou chyby zaviněné člověkem. Jsou obvykle opakovatelné a je skoro nemožné se jich vyvarovat. Hardwarové chyby jsou náhodné a obvykle nápadné. Stává se, že se vyskytne i nenápadná hardwarová chyba, avšak tento případ je velmi vzácný. Prohlédneme-li si toto shrnutí, bude nám zřejmé, že bychom se měli více zabývat chybami, které zavinil člověk, než hardwarovými chybami. Chyby, které způsobil člověk, můžeme naštěstí více ovlivňovat.

Metody ke zvýšení spolehlivosti počítačových výsledků

Naším konečným cílem by mělo být, abychom zabránili všem chybám, které by mohl způsobit člověk. Zdá se však, že je to cíl nerealisticky vysoký. Realističtější proto bude usilovat o to, aby vypočtené výsledky byly skoro jisté. Uvedeme nyní čtyři metody, které lze ke zvýšení spolehlivosti počítačových výsledků použít.

Ruční důkaz výsledku

Předpokládejme, že jsme použili některý symbolický manipulační program. Přijmout nebo nepřijmout takové výsledky bez dalších výhrad? Obávám se, že většina z nás je přijme. Měli bychom se však alespoň pokusit o ruční důkaz. Podaří-li se nám dokázat nějaký výsledek ručně, pak odpadnou všechny starosti kolem možných počítačových chyb. Ronald L. Graham v [7] napsal: „Když člověk ví, co chce dokázat, vyhrál víc než polovinu bitvy." Máme-li například najít rozklad složeného čísla na prvočinitele, potom stačí faktory ručně vynásobit a výsledek zkontrolovat. Je to jistě nejlepší řešení.

U většiny počítačových výsledků lze postupovat podobně. Není to ovšem vždy tak jednoduché a přímočaré jako v příkladě, který jsem uvedl. Velice rád bych třeba viděl počítačově nezávislý důkaz toho, že neexistuje konečná projektivní rovina řádu 10. Za mimořádný výkon bych považoval i důkaz některého z hlavních případů.

Co dělat v případě, že se nám nepodaří dokázet výsledek ručně? Zatvrzele to zkoušet a doufat, že se nám jednou podaří najít počítačově nezávislý důkaz, nemá asi smysl. Můžeme si však nechat vytisknout jednotlivé kroky. Někdy to při hledání důkazu pomůže. Považuji za nedostatek většiny symbolických manipulačních programů, že tisk jednotlivých kroků důkazu je při jejich použití velmi obtížný nebo dokonce nemožný.

Můžeme udělat i to, že pustíme program podruhé. To má však smysl jen v tom případě, že máme podezření na neobjevenou hardwarovou chybu. Chyby programu jsou opakovatelné a výsledek výpočtu bude proto vždycky stejný, bez ohledu na to, je-li správný nebo chybný. Hledejme raději jiný způsob, jak výsledek zkontrolovat.

Zdvojení výpočtu

Uvažme znovu případ, kdy řešíme nějakou matematickou úlohu pomocí symbolického manipulačního programu. Existuje několik takových programů, jejichž možnosti jsou obdobné. Stejnou úlohu můžeme tedy řešit dvakrát, a to pomocí dvou různých programů. Pokud se výsledky obou výpočtů shodují, je pravděpodobnost chyby ve výsledku velice malá. Ani v takovém případě však nelze tvrdit, že je výsledek zaručeně správný. Jistě si dokážeme představit situaci, kdy při použití dvou různých programů dostaneme stejný, ale chybný výsledek. Může k tomu dojít například tak, že oba programy používají stejnou, ale chybnou metodu nebo že uděláme stejnou chybu při vkládání dat.

Podobně můžeme postupovat i v případě, že používáme vlastní program. Můžeme napsat *dva různé programy, nejlépe takové, které používají různé metody* a byly napsány různými lidmi, a výsledky obou programů porovnat. Možná, že si pomyslíte, že to není použitelný nápad. Vždyť sestavit jeden provozuschopný program není jednoduchá záležitost a což teprve dva! Uvědomte si však, že potíže při sestavení takového programu spočívají především v tom, že požadujeme, aby *jeden* program byl současně rychlý i dostatečně obecný. To jsou často dva protichůdné požadavky, které sestavení programu komplikují. Napsat druhý, jednodušší program, u kterého nám na rychlosti a obecnosti tolik nezáleží, může být podstatně snazší. Jde-li o program, který běží dlouho, musíme často napsat programy dva: jeden jednoduchý, ale pomalý program pro zkušební účely a druhý program komplikovaný, ale rychlý pro vlastní provoz. My jsme při hledání konečné projektivní roviny řádu 10 používali dokonce řadu programů. Náš výzkum se rozpadal na řadu dílčích případů a my jsme pro každý případ napsali rychlý, „na míru šitý" program. Kromě toho jsme však měli ještě jeden program, který byl sice řádově čtyřikrát pomalejší, ale dokázal zpracovat libovolný případ. Výsledky příslušné dvojice programů jsme pak porovnávali. Pomalým programem jsme však nemohli opakovat celý výpočet pro každý dílčí případ, bylo by to trvalo příliš dlouho. Omezili jsme se proto vždy na porovnání několika malých vybraných případů. Nešlo

tedy o úplné zdvojení výpočtu, ale shodné výsledky u vybraných případů nás přesto uklidnily.

Dokonce i v případech, kdy máme program jenom jeden, můžeme zvýšit spolehlivost výsledků ověřením vnitřní konzistence programu.

Ověření konzistence

Začnu tím, že popíši jednu ze svých programátorských chyb. V roce 1975 jsem zkoumal *1-šířku* (0,1) matice. Jako součást výzkumu jsem musel generovat všechny čtvercové (0,1) matice řádu 6. (0,1) matice je matice, jejíž všechny prvky jsou buď 0, nebo 1. 1-šířka (0,1) matice je nejmenší počet sloupců matice takový, že v každém řádku je alespoň jedna 1. Každá čtvercová matice řádu 6 má 36 prvků; protože každý prvek může být buď 0, nebo 1, existuje celkem $2^{36} \approx 7 \times 10^{10}$ takových matic. Celkový počet matic je příliš vysoký na to, aby bylo možné zkoumat je přímo. Využil jsem proto skutečnosti, že 1-šířka (0,1) matice je invariantní vzhledem k permutaci řádků nebo sloupců. Tyto permutace rozdělí uvažované matice do tříd ekvivalence. Generoval jsem proto pouze jedinou matici z každé ekvivalenční třídy. Počet matic, které bylo třeba vyšetřit, se tím snížil zhruba $6! \times 6!$ krát, tedy přibližně na 10^5 případů. Když program proběhl, chtěl jsem zkontrolovat, mají-li výsledky smysl. Velikost každé ekvivalenční třídy se dá spočítat pomocí Pólyovy teorie počítání nebo Burnsideova lemmatu. Když jsem však velikosti všech ekvivalenčních tříd sečetl, dostal jsem číslo, které bylo o něco menší než 2^{36}. Během následujícího víkendu jsem šílel a hledal chybu. Nevěděl jsem přitom, kde vlastně začít. Zapomněl jsem, jak byla chyba velká. Pamatoval jsem si jen to, že šlo o malou chybu. Připomínám tuhle historku proto, že jsem si tenkrát uvědomil, že každý program, jehož úkolem je prohledávání jednotlivých případů, by měl obsahovat nějaké ověření vnitřní konzistence.

Co je ověření konzistence? Je to kontrola toho, že vypočtený výsledek není sporný. Výsledky mnoha výpočtů jsou vázány jistými vnitřními vztahy. Pro součet stupňů všech faktorů daného polynomu například platí, že musí být roven stupni původního polynomu nebo pro celkový počet čtvercových (0,1) matic řádu 6 platí, že je roven číslu 2^{36}. Chcete-li provést ověření konzistence, najděte nejprve nějaké snadno ověřitelné vztahy mezi očekávanými výsledky, nejlépe takové, které se vztahují k celému výpočtu. Splnění těchto vztahů je potom třeba ověřit, a nejsou-li splněny, signalizuje to přítomnost chyb.

Kde takové vztahy hledat? Univerzální odpověď na tuto otázku neexistuje, je třeba vycházet z povahy řešeného problému. Pro kombinatorické výpočty je typická metoda počítání, kterou jsem popsal výše pro úlohu 1-šířky. Permutace řádků a sloupců je potom třeba zobecnit na operace, které zachovávají danou vlastnost. Myšlenka, že stačí vyšetřovat pouze jediného zástupce z každé třídy ekvivalence, je základem redukce pomocí izomorfismu. Tato metoda dokáže kombinatorický program úžasně zrychlit, bývá však také zdrojem mnoha chyb v programu. Můžeme si však pomoci tím, že spočítáme objekty dvěma různými způsoby. Tento postup je velmi účinným způsobem, jak ověřit

konzistenci programu [5], a mě pomohl k objevení mnoha vlastních programátorských chyb. Cítím se mnohem lépe, když výsledky, které spočítám, projdou touto kontrolou.

Používejte dobře vyzkoušené programy

Tohle doporučení vypadá jako tautologie. Přesto se z něho dají odvodit některé užitečné zásady. Například: nepište vlastní programy, pokud to není nutné. Je jen velmi málo případů, kdy matematik skutečně musí napsat vlastní komplikovaný program. Existuje přece tolik výborných programových balíků, které se dají jednoduše ovládat a které jsou při tom schopny spočítat většinu toho, co potřebujeme. Protože tyto balíky používá mnoho lidí, mají za sebou důkladnou kontrolu, a proto jsou spolehlivější.

S tím, co jsem právě řekl, souvisí i další zásada: používejte programové balíky, které se těší všeobecné oblibě. Čím víc lidí nějaký balík používá, tím je menší pravděpodobnost, že jsou v programech chyby.

Matematici berou často v úvahu i cenu softwarových balíků. Můžeme-li zdarma použít spolehlivý softwarový balík, určitě nebudeme váhat a použijeme ho. Uvědomme si však, že navrhnout a sestavit dobrý programový balík něco stojí a že tyto náklady musí někdo zaplatit. Nesmíme proto připustit, aby se pro nás stala cena balíku rozhodující. Přezkoumejme pečlivě, co všechno balík dokáže a jak je spolehlivý. Vždyť vyšší cenou si můžeme pojistit správnost výsledku. A trochu vyšší náklady snad za klid v duši stojí.

Protože jsme matematici, používáme počítač jako nástroj. Potřebujeme však, aby to byl nástroj spolehlivý. Myslím si, že bychom měli být konzervativní a držet se raději vyšlapaných cest. Objevování nových obzorů ve výpočetní technice přenechme raději informatikům a matematikům, kteří se v informatice dobře vyznají.

Jak hodnotit počítačové důkazy?

Článek, který obsahuje počítačový důkaz, staví recenzenta do nové role. Od recenzenta se totiž obvykle očekává, že správnost důkazů zkontroluje. Jak má ubohý recenzent provést kontrolu důkazu, který „nikdo nemůže zkontrolovat?“ Předpokládá se snad, že napíše vlastní program, aby výsledky zkontroloval? A co když program spotřebuje spoustu strojového času, jako třeba program pro hledání konečné projektivní roviny řádu 10? Tohle všechno se na recenzentovi jistě žádat nedá.

Myslím si, že s počítačovými důkazy by se mělo zacházet tak, jak vědci v jiných oborech zacházejí s experimentálními výsledky. Autor takového článku by měl kromě vlastních výsledků uvést také metodiku, kterou použil, provést analýzu výsledků a diskusi toho, jak jeho výsledky zapadají do známé teorie. Recenzent může posoudit, je-li tato metodika správná a také to, jak se autor postaral o zajištění správnosti svých výsledků. Recenzent by měl také rozhodnout, jsou-li uváděné výsledky zajímavé a věrohodné.

Budeme-li používat kritérium „věrohodnosti", stane se někdy, že budou publikovány i výsledky chybné. V jiných vědních oborech už však existuje obvyklý postup, jak takový problém řešit: zdůrazňuje se v nich úloha nezávislé verifikace experimentálních výsledků.

Nezávislá verifikace

V případě počítačových důkazů je třeba podporovat to, aby byly nezávisle ověřeny jinou skupinou vědců. Nezbytnost takové nezávislé verifikace si může vyžádat revizi tradičních představ o tom, co je publikovatelné. V matematice není zvykem zveřejňovat druhý důkaz téže věty s výjimkou případů, kdy je v takovém důkazu něco nového. Nezávislá verifikace však žádnou novou myšlenku obsahovat nemusí a podle současných kritérií by se tedy uveřejnit nedala. Ve světě, který se řídí heslem „publish or perish", nemá přirozeně nikdo zájem opakovat práci někoho jiného. V důsledku takového přístupu se může stát, že se na chyby v některých počítačových důkazech dlouhou dobu nepřijde. Doufám však, že se v zájmu celé disciplíny tento přístup změní. Osobně se domnívám, že časopis, který uveřejní nějaký počítačový důkaz, by měl mít morální povinnost uveřejnit také jeho nezávislou verifikaci, i když třeba jenom ve zkrácené formě. Vždyť teprve tato verifikace celý důkaz uzavírá.

Závěr

S příchodem počítačů jsme my, matematici, dostali do rukou velice výkonný nástroj, který nám umožňuje řešit komplexní problémy. Určitá nejistota, která je počítačovým důkazům vlastní, by nás neměla od jeho používání odrazovat. Nositel Nobelovy ceny Richard Feynman kdysi řekl: „Vědecké poznání je soubor tvrzení s různým stupněm jistoty; některá z těchto tvrzení jsou velmi nejistá, některá jsou téměř jistá, ale žádné z nich není *zcela* jisté." (Feynmanova přednáška v americké Národní akademii věd z roku 1955). Když se s neurčitostí naučili žít fyzici, měli bychom se i my naučit žít s „nejistými" důkazy.

Literatura

[1] K. Appel a W. Haken: *Every planar map is fourcolorable.* Bull Amer. Math. Soc. *82* (1976), 711–712.

[2] M. W. Browne: *Is a math proof a proof if no one can check it?* The New York Times (20 December 1988).

[3] K. R. Foster a H. H. Bau: *Symbolic manipulation programs for the personal computer.* Science *243* (3 February 1989).

[4] C. W. H. Lam: *The distribution of 1-width of (0,1)-matrices.* Discrete Mathematics *20* (1977), 109–122.

[5] C. W. H. Lam, L. H. Thiel a S. Swiercz: *The nonexistence of finite projective planes of order 10.* Can. Journal of Math. (v tisku).

[6] P. Wallich: *Beyond understanding? Computers are changing the spirit of mathematics.* Scientific American (March 1989), 24.