Цейтин Григорий Самуилович

ЯВЛЯЕТСЯ ЛИ МАТЕМАТИКА ЧАСТЬЮ ИНФОРМАТИКИ?

Я не согласен с утверждением, что информатика (какой бы термин для нее ни использовали) – это набор практических навыков и решений, в лучшем случае – искусство, и никакого фундаментального научного содержания она иметь не может. Я думаю, что, несмотря на возможные терминологические недоразумения, мы все более или менее одинаково понимаем, о какой области деятельности идет речь. Информационные технологии в современном мире – это уже давно утвердившаяся реальность, так же, как и то, что существуют профессионалы, специализирующиеся именно в этой области, что необходимо готовить специалистов в этой области, писать книги и статьи, издавать журналы, оценивать профессиональный уровень и т.п. Раз есть область, должно быть имя, чтобы ни с чем не спутать. Термин информатика (французское informatique) представляется мне достаточно удачным, и, во всяком случае, лучшим, чем американское computer science.

Прежде, чем пытаться уточнить содержание этой области (насколько это вообще возможно), хотелось бы проследить, как вообще формировалось самоосознание этой области, как особой области знаний, а также общественное признание ее самостоятельности (сегодня такое признание – уже свершившийся факт).

Но ни самоосознание области, пионеры которой были и считали себя специалистами в других областях, ни выделение ей места под солнцем в ряду других специальностей, не могли быть простыми. Новая область требует выделения ей отдельных ресурсов, и людских, и материальных, а это не могло происходить бесконфликтно и вполне добросовестно. Я имею в виду не только СССР, где все эти проблемы многократно усиливались научным монополизмом и порожденными им интригами. Эти проблемы существовали и в более благополучных странах.

Нового предмета, отличного от всего, что было прежде, информатика не создала. Программа вполне подходит под математическое понятие алгоритма (с некоторыми уточнениями из-за параллельного исполнения или недетерминированности), так где же новый предмет? И представители традиционных областей, стремясь удержать под своим контролем ресурсы, выделение которых на новые приложения диктовалось практическими потребностями, пользовались этим аргументом. Как говорил один мой коллега-матфизик: «И чего это все так носятся с этим системным программированием? Это ведь всего-навсего программирование для системы машин!» А другой коллега, просматривая проект учебных программ по информатике, заявлял: «Это какая-то эклектика, просто собраны вместе кусочки, принадлежащие другим дисциплинам». Впрочем, подобная аргументация известна еще из пушкинской «Сказки о царе Салтане», где «ткачиха с поварихой, с сватьей бабой Бабарихой» развенчивали (и небескорыстно) одно за другим все чудеса, о которых рассказывали заморские гости.

Надо признать, что и представители новой области допускали натяжки ради того, чтобы организационно выделить свой

предмет. Мне приходилось видеть математические работы, где поверхностно формализовывались некоторые, уже устарелые, программистские концепции, а затем доказывались «сногсшибательные» результаты, основанные на совершенно нереальных примерах (в математике это нормально, но на основе этого следовало просто заменить первоначальные понятия, чтобы они не включали подобные случаи). И все это делалось ради того, чтобы заявить о принадлежности своих (не бог весть каких) результатов новой перспективной области. Впрочем, эти болезни были постепенно преодолены.

А как осознает себя эта область теперь, когда организационное признание состоялось? К сожалению, осознания особого предмета по-настоящему нет. Авторитетные специалисты, пришедшие из других областей, зачастую связаны прежними представлениями, а молодежь, похоже, просто работает, не задумываясь об отграничении предмета.

А необходимость в таком осознании есть. Действительно, компьютерная программа может рассматриваться как разновидность алгоритма, но почему в таком случае возникают все новые и новые языки программирования и новые концепции, например, объектно-ориентированное программирование? Ведь в принципе любая программа эквивалентна некоторой машине Тьюринга, так что вроде бы ничего нового все эти языки не несут! А дело в том, что даже в одном и том же предмете с разных точек зрения могут быть важны разные стороны, и то, что важно с точки зрения математики, не совпадает с тем, что важно с точки зрения информатики. Различие между математикой и информатикой в оценочных критериях в свое время достаточно четко описал Ласло Кальмар*, пришедший в программирование из математической логики.

Главное отличие от математики, хотя бы и рассматривались одни и те же объекты, состоит, на мой взгляд, в том, что в информатике определяющим является человеческий фактор. Программы пишутся людьми, часто большими коллективами; даже если программу пишет один человек, он пользуется полученными от других людей знаниями и приемами, и, возможно, получил первоначальное задание от кого-то другого. Программа имеет жизненный цикл: после создания она может модифицироваться, переноситься в другую среду, стыковаться с другими программами, и в конце концов выходить из употребления (тоже по разным причинам). Учитывает ли эти реальности математическое понятие, претендующее на определение программы? Разве может алгоритм (в точном математическом понимании) меняться? Если что-то изменится, это будет просто другой алгоритм, который не надо путать с первым. Если же мы хотим, чтобы программа, например, адаптируемая к другому окружению, оставалась тем же самым объектом, то понятие программы должно существенно измениться (в общем, некоторые математические подходы к описанию этого тоже возможны, но никогда теоретики со стороны математики этого не предлагали). Важной особенностью программ является то, что они могут иметь ошибки, и, во всяком случае, необходимы меры для уменьшения их числа и ущерба от них. Это тоже не из математики. Вспомним, кстати, и о проблеме двухтысячного года.

Для чего придумываются новые и новые языки программирования? «Для большего удобства», возможно, ответят. А что такое удобство? Это снова человеческий фактор. Если всерьез, то это более точное соответствие элементов предлагаемого языка и тех понятий и знаний, которыми пользуется человек, ставя и решая ту же задачу. Значит, надо понимать структуру человеческого знания и человеческого мышления. Целый ряд особенностей новых языков программирования, возможно, казавшихся первоначально случайными удачными находками, отражает глубокие черты организации человеческих знаний и человеческого языка. Но достаточного внимания этому не уделяется. (Один мой коллега из промышленности,

нуждавшийся в специалистах по языкам программирования, спрашивал меня, не на филологическом ли факультете их искать.)

Кроме индивидуального фактора, есть еще и социальный. Уже упоминалось о разработке программ большими коллективами, а также о сопровождении программ. Есть еще необходимость переносить программы на вновь разрабатываемую технику или включать в новые системы. Надо считаться и с тем, что программы, работающие на одной машине или иным образом взаимодействующие либо совместно пользующиеся ресурсами, могут отражать интересы разных людей, возможно, находящихся в конфликте, и, значит, надо заниматься защитой программ.

Когда стал широко входить в жизнь Интернет, стала насущной необходимостью разработка подходов, обеспечивающих правильное взаимодействие большого числа независимо разработанных программ (или их элементов). Одновременно увеличение скорости машин сделало менее существенной экономию команд на нижнем уровне. В сумме это привело к использованию структур данных, которые раньше считались бы неприемлемыми из-за неэффективности, и, соответственно, к новой организации языков. Широкое распространение модульности вместе с приемлемостью больших накладных расходов на межмодульное взаимодействие привело к уменьшению размеров отдельных модулей, и, как следствие, к упрощению синтаксиса языков. Кстати, в этом я вижу и причину создания и последующего широкого распространения языка Java.

Кому же заниматься исследованием человеческих и социальных факторов в информатике? Снова чудится мне указующий перст чиновного скептика, внушающего, что это дело таких-то и таких-то уже существующих наук: психологии, социологии, и т.п. Не получится! И не только потому, что во всех этих науках есть свои интересы и предпочтения, а их представители могут и не понять важности проблем, диктуемых задачами информатики (у меня есть определенный отрицательный опыт). Достаточно рассмотреть, к чему привело отождествление информатики с математикой.

Информатика получила от математики ряд результатов и теорий, нашедших широкое применение, в особенности в теории языков и трансляции, а также (в меньшей мере) по верификации программ. Вместе с тем, поскольку это делали математики (или люди, относившие себя к математике) выбор задач диктовался желанием получить результаты, интересные в математическом смысле, а другие, не менее важные для информатики, задачи оставались без внимания. Теория языков и трансляции, например, оказалась слишком раздутой, а вопросы модульности (что на сегодня важнее) не получили должного развития. Преувеличена была и роль логической верификации – на практике требования к программам редко оформляются в логических понятиях. Отрицательную роль сыграла и ориентация на «диссертабельность». Программистские работы, содержавшие достаточный творческий вклад, обладавшие и новизной, и полезностью, и делавшие ее автора достойным ученой степени, искусственно подводились, ввиду отсутствия надлежащих рубрик, под вычислительную математику, экспериментальную физику и т.п., и люди, причастные к прохождению работы, закрывали глаза на то, что она заявленной специальности не соответствует. Это, в свою очередь, приводило к появлению, в силу прецедента, диссертаций, не содержавших серьезного вклада ни в «титульную» область, ни в информатику. С другой стороны, программистские работы иногда снабжались «орнаментальной» математикой, когда определения искусственно стилизовывались под математические теории.

Нет оснований рассчитывать и для других смежных областей, что вопросы, практически важные для информатики, будут успешно решаться в рамках этих областей. Отсутствие полномасштабного самоосознания информатики как особой науки начинает мешать ее развитию.

* Приводим краткое изложение доклада Ласло Кальмара (Венгрия) на IV Международной конференции по логике, методологии и философии науки (29 VIII – 4 IX 1971) и обращаем внимание читателя на любопытное замечание, которое сделал один из создателей информатики Еджер В. Дейкстра: *«С исторической точки зрения интересно, но неубедительно! Его (Л.Кальмара) точка зрения на информатику устарела, а на математику – слишком узка.»*

IS COMPUTING SCIENCE AN INDEPENDENT SCIENCE?

Computing Science has obviously its origin in Mathematics. The question is, whether it is a branch of Mathematics or it can be considered as an independent science.

Beside it special subject field, Computing Science diverges from Mathematics by its method. Indeed, while Mathematics is a proof-oriented science, Computing Science is more algorithm-oriented. In any case, a computer scientist places generally as much ingenuity into his algorithms as a mathematician into the proofs of his theorems. True enough, algorithms play some role in Mathematics as well. However, even the most sophisticated mathematical algorithms (e.g. Kronecker's algorithm for decision of the reducibility of a polynomial in the field of rationals, or Galois' algorithm, using the latter, for decision of solvability, by means of radicals, of an algebraic equation with rational coefficients) are very short relative to a compiling algorithm or to an operational system.

Also, the computer scientist has to prove his propositions, e.g. the correctness of his programmes. However, in most cases, the proof has a verificatory character. The name «debugging» given to such verifications shows that the computer scientist does not esteem this activity, though important, so high as the mathematician his proofs. In most cases, the errors found in the course of debugging can easily be corrected (at least if the programming idea is sound), while errors in mathematical proofs are in general fatal.

A mathematical problem, asking if some statement is true or not, is finally solved by a proof or disproof of the

ЯВЛЯЕТСЯ ЛИ ИНФОРМАТИКА САМОСТОЯТЕЛЬНОЙ НАУКОЙ ?

Информатика, очевидно, ведет свое происхождение из математики. Вопрос в том, является ли она ветвью математики или самостоятельной наукой.

Помимо своего особого предмета исследования информатика отличается от математики и своим методом. Действительно, в то время как математика – наука, ориентированная на доказательство, информатика ориентирована в большей мере на алгоритм. Во всяком случае информатик вкладывает в свои алгоритмы, вообще говоря, столько же изобретательности, сколько математик – в свои доказательства. Правда, и в математике алгоритмы играют определенную роль. Однако даже самые изощренные математические алгоритмы (алгоритм Кронекера для определения приводимости многочлена над полем рациональных чисел или алгоритм Галуа, использующий алгоритм Кронекера, для определения разрешимости в радикалах алгебраического уравнения с рациональными коэффициентами) имеют очень отдаленное родство с алгоритмом компиляции или операционной системой.

Информатик тоже должен доказывать свои утверждения, то есть правильность своих программ. Однако в большинстве случаев доказательство имеет скорее характер проверки. Термин «отладка», применяемый к такой проверке, показывает, что информатик не оценивает эту часть своей деятельности, хотя и важную, так же высоко, как математик - свои доказательства. В большинстве случаев ошибки, найденные в процессе отладки, могут быть легко исправлены (по крайней мере, если основная идея программы верна), в то время как ошибки в математических доказательствах, вообще говоря, фатальны.

Решение математической задачи, в которой ставится вопрос, верно или неверно некоторое утверждение, полностью завершается

statement in question. On the contrary, if one has a computational algorithm for a solution of a given problem of Computing Science, the problem is not finally settled, for one is asking for a better algorithm for the same goal (from the point of view of computer time or memory place). Well, a mathematician can also look for a simpler proof of some theorem. However, to find one is not as great an achievement as to find the first proof. On the other hand, the improvement of a computational algorithm is sometimes as (or more) valuable as production of the first algorithm for the same purpose.

These arguments show that Computing Science requires a way of thinking different from that of traditional mathematics. Hence, Computing Science is appropriately considered an independent science rather than a branch of Mathematics.

после того, как это утверждение доказано или опровергнуто. В противоположность этому, если имеется алгоритм решения некоторой задачи информатики, то решение задачи нельзя считать окончательным, так как далее ищется лучший (с точки зрения скорости или требуемой памяти) алгоритм решения той же задачи. Конечно, математик тоже ищет более простое доказательство какой-либо теоремы. Однако лучшее доказательство не является таким же достижением, как первое доказательство. С другой стороны, создание лучшего алгоритма в некоторой программе иногда столь же (или даже более) ценно, как создание первого алгоритма для решения той же задачи.

Эти доводы показывают, что информатика требует способа мышления, отличного от того, который применяется в традиционной математике. Следовательно, информатику можно рассматривать скорее как самостоятельную науку, а не как ветвь математики.

IVth International Congress for Logic, Methodology and Philosophy of Science (29 VIII – 4 IX 1971). Abstracts. Centre of Information and Documentation in Social and Political Sciences. Bucharest. Pages 89-90.

IV Международная конференция по логике, методологии и философии науки (29 VIII – 4 IX 1971). Аннотации. Центр информатики и документации в социальных и политических науках. Бухарест. Стр. 89-90.

Цейтин Григорий Самуилович,
доктор физ.-мат. наук,
профессор СПбГУ.