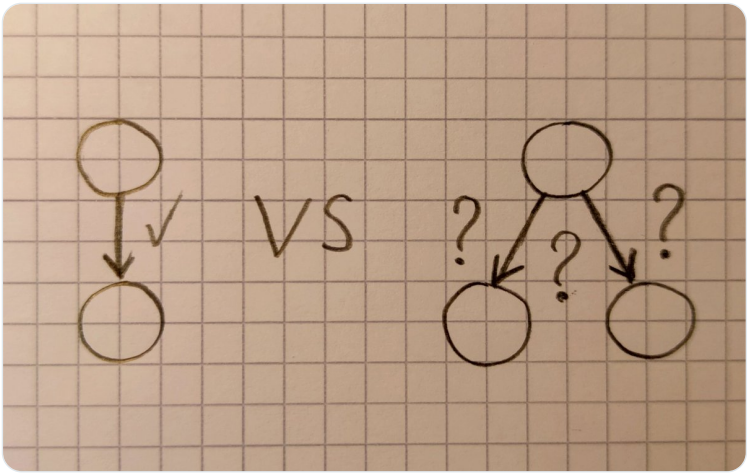
Валерий Жила @ValeriiZhyla
Oct 25, 2021 • 72 tweets • ValeriiZhyla/status/1452567582087782402

Моя цель сегодня – разобрать и объяснить суть недетерминизма простыми словами. За этим странным словом скрывается очень интересный мир. Поехали!



Речь пойдет сначала про детерминизм, дальше про недетерминизм, а потом про разные вариации конечных автоматов и машин Тьюринга.
И о том, как со всем этим работать, зачем оно нам вообще нужно, и что это нам всем предвещает.

Итак, детерминизм.

Предлагаю сначала дать очень простое определение этому сташному слову. На самом деле, детерминизм наш хороший друг (а недетринизм - тот еще мудак для понимания)

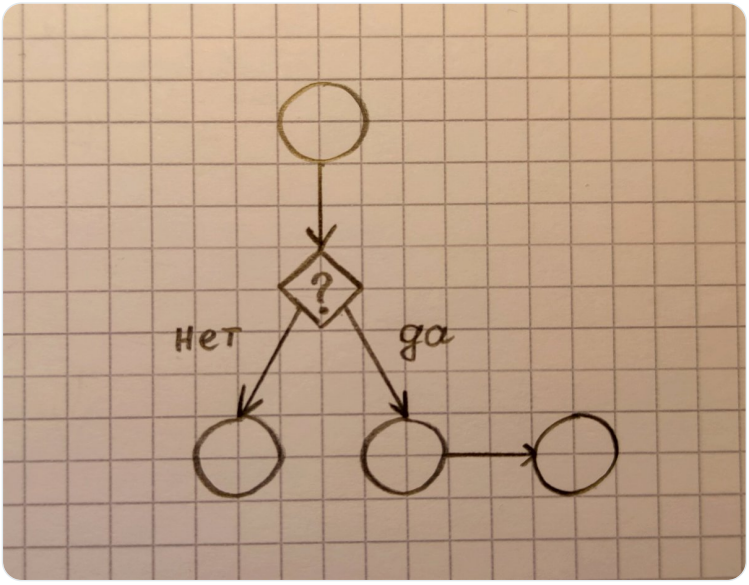
Детерминизм - это когда при одних и тех же вводных данных программа будет работать одним и тем же образом, и в итоге выдавать один и тот же ответ. Вот так вот просто. Как мы все привыкли.

Например, мы берем калькулятор (хороший калькулятор), включаем его, и жмем кнопки "3", "+", "5", "=".
Это наши вводные данные, только это. Если за калькулятором не скрывается какая-то мистическая сила, он исправен, и так далее по списку - на экране высветится цифра 8

Сколько бы раз мы не жали на эти кнопки, всегда калькулятор будет производить одну и ту же операцию в своих микросхемах, и выдавать один и тот же ответ. Это хороший пример детерминистичной (или детерминистичной, deterministic в общем, мы друг друга поняли) работы!

Все классические алгоритмы, все программы, которые мы пишем - детерминистичны. Наша жизнь - детерминистична (наверное, это не место и не время для дискуссии о свободе воли), хотя из-за сложности системы и неизвестности ее точного состояния - кажется, что это не так.

Ветвление в наших программах однозначно. Если число чётное, то делай одно. Если нечётное – делай другой. Нет и не может быть двух разных сценариев, которые будут выполнены для чётного числа. Мы к этому привыкли.



С детерминизмом всё просто, и надеюсь, понятно.
Настало время представить нашего нового друга - недетерминизм!

Недетерминизм, это, в прямом смысле НЕ_детерминизм - одни и те же входные данные могут быть обработаны разными путями. Причем то, какой вариант будет выбран - заранее неизвестно. Более того, у этого выбора нет какой-то причины, которую можно заранее проконтроллировать.

Часто в качестве примера недетерминизма приводят мысленный эксперимент с Котом Шредингера. Пока мы не открыли коробку, мы не знаем, жив кот или не очень.

Точно так же, пока программа не закончила работу - мы не знаем ничего о том, какой она даст ответ в этот раз. Мы даже не знаем, какой путь решения она выберет.

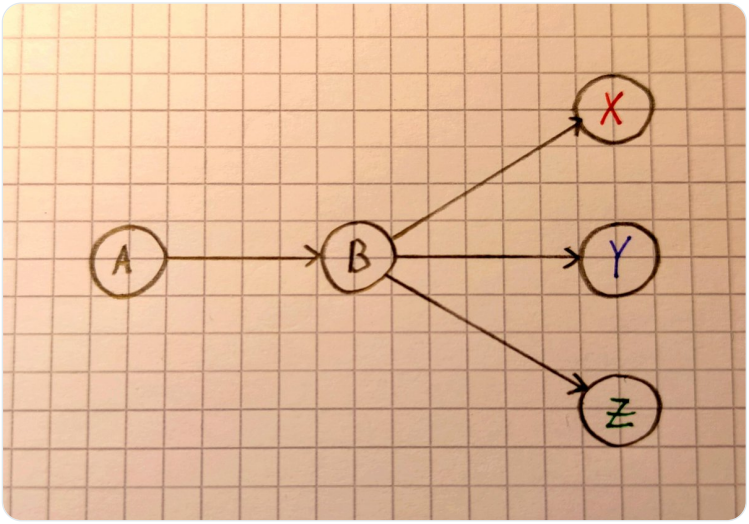
Посмотрим, окажется ли квантовый мир действительно недетерминистическим.
Сегодняшняя физика вроде как настроена по этому вопросу положительно. Я лично - хочу верить в лучшее для себя, а мне по душе исключительно детерминистическая картина мира.

Время покажет, и мои личные желания на это точно ни как не повлияют - я и физика находимся в разных вселенных.
Но деда понесло, возвращаемся к теме!

Может появиться соблазн смешать недетрминизм с теорией вероятности. Например - с вероятностью в 50% будет выбран вариант A, а с вероятностью 50% - вариант B.
Это выглядит интуитивно, но к недетерминизму статистика и теория вероятности не имеет отношения никакого.

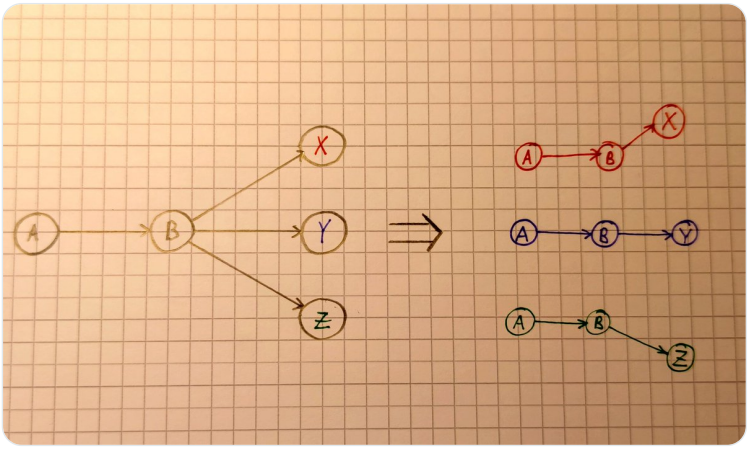
Хорошо, но как тогда понимать эту всю кухню?!
Мне лично нравится представление в виде "ветвления", и я его сейчас вкратце обрисую.

Итак, у наша программа находится в состоянии B. Из него она может недетерминистично перейти в состояние X, Y, или же Z



На каждой "точке ветвления" программа как бы создает свои копии, каждая из которых идет по одному из возможный путей.

Уникальные сценарии работы программы множатся на каждой точке ветвления, подобно клеткам в организме



На этом примере видно, что программа с тремя вариантами работы превратилась в три программы, которые дальше работают по своему сценарию, и каждая может так делиться все дальше и дальше, и каждая даст свой ответ

Замечательно, но тогда ведь и ответов будет великое множество! Как быть?

В общем случае - никак, но на общем случае мы далеко не уедем. Нужно для каждого конкретного случая установить критерий выбора ответа и всей бесконечности потенциальных сценариев.

Нас интересуют два конкретных случая, конечные автоматы и тьюринг машины.

Эти модели обычно получают какое-то слово (последовательность символов) в качестве вводных данных, и проверяют его, согласно своей внутренней программе, чтоб выдать ответ ДА или ответ НЕТ.

Так вот, в этих двух моделях мы обычно говорим, что недетерминистическая программа выдает ответ ДА в том случае, если хотя бы один ее "сценарий" выдает ответ ДА. В противном случае - НЕТ.

Это как если бы мы играли в дартс (там где дротики кидаешь), и автоматически побеждали бы, если бы попали в серединку хотя бы один раз.

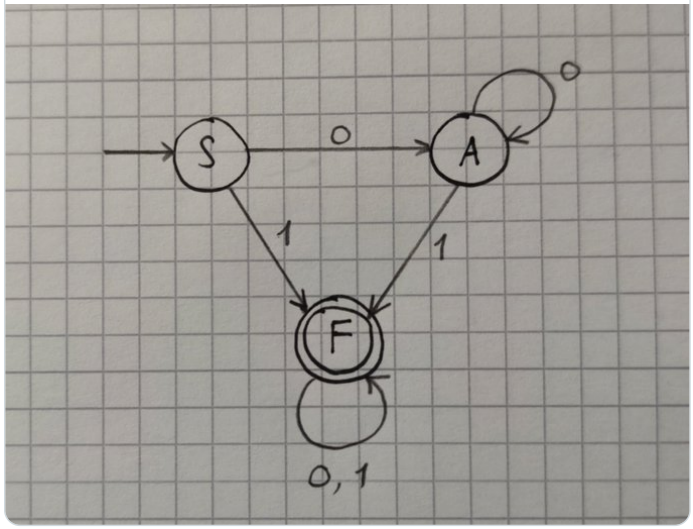
Если вы увидели словосочетания "конечный автомат" и "машина Тьюринга", и испугались - спешу вас успокоить!
Треды про них уже были любезно написаны мною же.

Валерий Жила
@ValeriiZhyla

Replying to @ValeriiZhyla

Тред про конечные автоматы:

> **Валерий Жила** @ValeriiZhyla
> Я постараюсь рассказать о конечных автоматах так, чтоб и ребёнку стало понятно!



9:20 PM · Oct 15, 2021

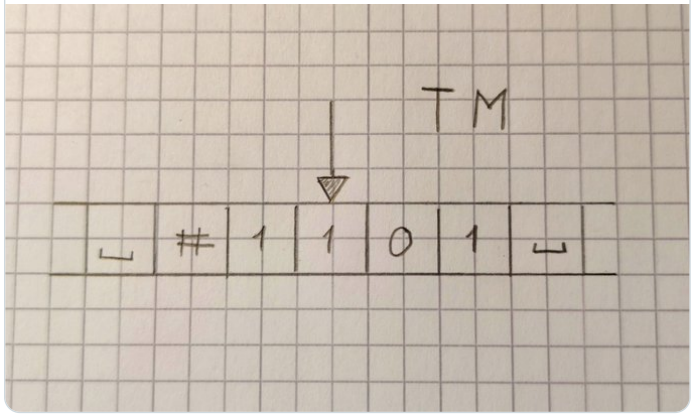
7 Copy link to Tweet

Валерий Жила
@ValeriiZhyla

Replying to @ValeriiZhyla

Тред про Машины Тьюринга:

> **Валерий Жила** @ValeriiZhyla
> Я расскажу очень простым языком о том, как устроены Тьюринг Машины, и зачем они вообще нужны!

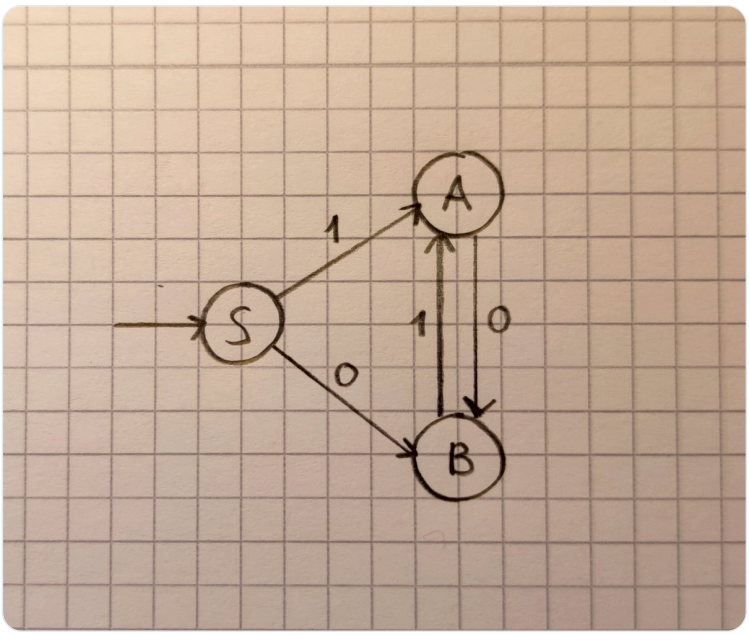


10:08 PM · Oct 19, 2021 from Ettlingen, Deutschland

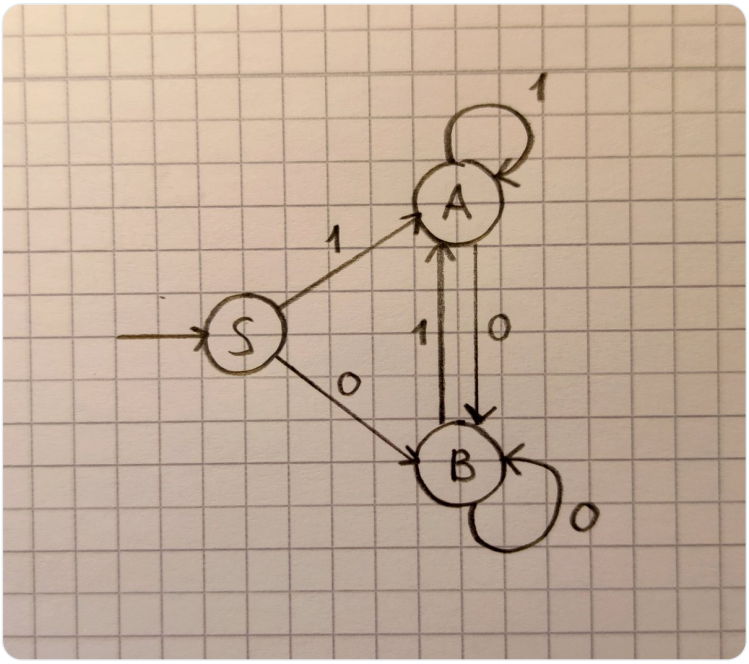
6 Copy link to Tweet

С основными понятиями разобрались, но скорее всего, про недетерминизм ещё ничего толком не понятно. Спокойно, все только начинается. Предлагаю начать с автоматов!

Итак, вот наш простой, детерминистический, конечный автомат. Коротко – ДКА. В каком бы состоянии мы бы не находились, получая символ 1 или 0, мы точно понимаем, в какое состояние нам переходить. Ни больше, ни меньше. Детерминизм на лицо!

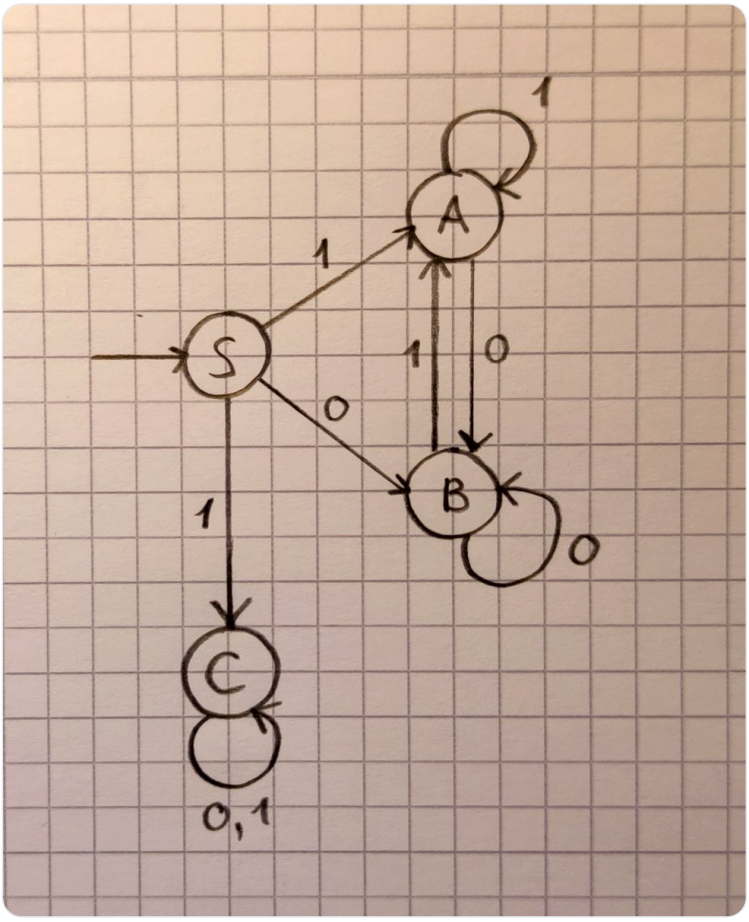


Чтоб было совсем красиво, сделаем ещё так. Теперь для каждого состояния есть переход для каждой буквы. Этот автомат отвечает всем требованиям детерминизма.



А теперь внесём одну маленькую модификацию, которая нам оторвёт весь детерминизм!

Мы добавили новое состояние, C. Куда теперь автомат перейдет из состояния S, когда получит единицу? В A или в C?



– Джонни, я детерминизм не чувствую
– Да потому что у тебя его нет!

Поздравляю, мы сделали недетерминистический конечный автомат. Коротко – НДКА. На один и тот же ввод возможны разные, равноправные и независимые ни от чего сценарии работы.

Могут ли НДКА что-то такое, чего не может ДКА? Хорошие новости для наших мозгов - нет, они вычисляют ровно тот же спектр задач.
Можно ли НДКА как-то хитро преобразовать так, чтоб он снова стал детерминированным? Да!

Суть преобразования НДКА в ДКА в переборе всех комбинаций состояний и создании новых состояний, каждое из которых соответвует одной из комбинаций.

Простым языком - НДКА можно представить как ДКА с оооочень большим количеством сотояний. Для НДКА с N состояниями в худшем случае выйдет 2^N состояний. Тобишь если в НДКА 10 состояний, и там "всё переплетено", то у ДКА, делающего ту же работу, будет около 2^10 = 1024 состояний

Автомат едет к бабушке, и возвращается очень упитанным, но детерминистичным.
Кажется сложно, доказывается не очень просто и довольно запутанно, но достаточно просто запомнить - переход из НДКА в ДКА возможен

Казалось бы, а в чем смысл? Автомат станет просто сложнее, наверное, он и потребует больше времени на работу?
Вообще-то совсем нет! Слово из 10 символов гарантированно обработается автоматом за 10 шагов. Количество состояний - просто brute force перебор комбинаций.

Программа разрастется, но медленнее работать не станет! Это возможно за счет простоты автоматов - у них нет никакой памяти, они "живут моментом".

В другую сторону всё работает совсем просто - и ежу понятно, что более простую модель (ДКА) можно номинально назвать моделью с менее строгими правилами (НДКА) по своей структуре, ей даже меняться никак не нужно для этого.
Достаточно просто не использовать это ослабление правил.

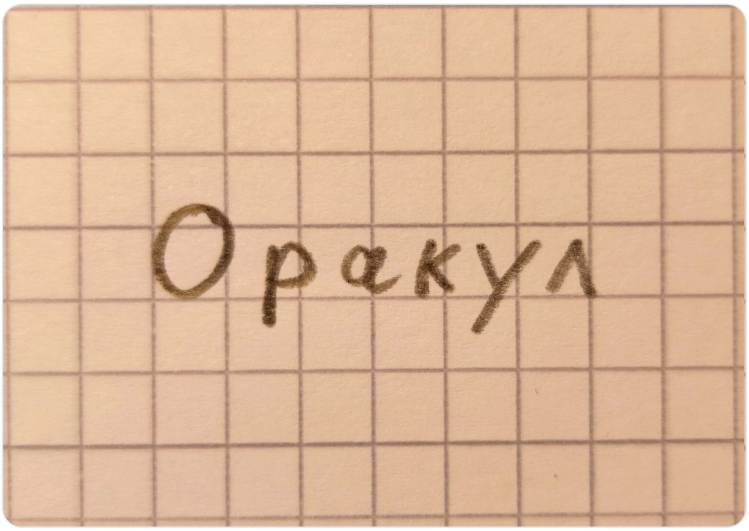
Выпускнику школы нужно очень постараться, чтоб стать выпускником университета. Но выпускнику университета и делать ничего не нужно, чтоб сказать, что он окончил школу - получение диплома никак не отменяет того, что он закончил школу.

Отлично, с автоматами с грехом пополам разобрались. Переходим к более тяжелой, но и более важной артиллерии - Тьюринг Машинам (ТМ)

Во-первых, нужно прояснить, что такое недетрминированная (недетерминистическая, nondeterministic в общем, вы меня поняли) ТМ - сокращенно НДТМ?

Существует несколько разных формулировок для НДТМ. Благо, они все эквивалентны (иначе было бы как-то тупо).
Спешу поделиться с вами самой простой и интуитивно понятной из них!

Она называется Oracle TM, она же TM с Оракулом, она же OTM



В чем суть: мы "прячем" весь этот недетерминизм в новую запчасть TM, под назаванием Оракул.
Как это работает:
1.Вводные данные (определенная последовательность символов) записана на ленту, все как в детерминированном варианте.

2.Когда машина запускается, срабатывает Оракул.
Оракул сразу угадывает ответ для вводных данных и записывает его рядом с вводными данными на ленту. После этого он отключается до конца работы TM.

3.OTM остается лишь проверить в обычном детерминистическом режиме (ДТМ), является ли решение Оракула правильным для конкретных вводных данных.

Для примера я расскажу, как можно использовать OTM для решения какой-то известной задачи. Этот пример - задача коммивояжёра.
Если что, это задача о поиске такого пути в графе, которых максимизирует выгодность маршрута.
Вот ссылка на ее описание:

Задача коммивояжёра — Википедия

https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0…

Итак, на ленту строкой записан граф (в форме линейного списка ребер, например), в котором нам нужно найти самый выгодный маршрут.

ОТМ запускает Оракула, и он сразу угадывает магическим образом самый выгодный маршрут.
Далее машина в детерминированном режиме проверяет, нет ли в графе маршрута, более выгодного, чем этот. Если такого нет - Оракул угадал правильно, и задача решена.

Вдох, выдох, спокойствие.

Угадывает? Магическим образом? Какой вообще в этом смысл? Где тут вообще недетерминизм? Ты издеваешься, да?

Я абсолютно серьезен! Самый сок в том, что ОТМ и классическая НДТМ, как я сказал, эквивалентны. Тобишь все то, что решает ОТМ, может симулировать обычная НДТМ, и наоборот. Обычная НДТМ работает так же, ДТМ, но имеет в программе недетрминированный автомат, вместо обычного

НДТМ намного сложнее для восприятия, чем ОТМ, поэтому я не хочу особо говорить о том. ОТМ замечательно служат полноценной заменой. Это доказанный факт, но доказательство этого я даже врагу не пожелаю увидеть. (А мне его нужно было выводить на экзамене☐)

Ладно, отставить сопли!
Модель с оракулом кажется более интуитивной.
Более того, она дает отправную точку в вычислениях, и использовать ее для разных целей в разы удобнее.

Посудите сами, что проще - найти самый лучший ответ, или имея вариант ответа, проверить, нет ли ответов лучше него?

Вот, в целом, и вся база недетерминизма.
Но зачем он нам вообще нужен? Чем плох детерминизм? Штош...

Если коротко: дело в том, что недетерминизм удобен.
Он очень развязывает руки в доказательстве возможности решения разных проблем.
Для практики беда в том, что недетерминистических вычислителей, в отличии от детерминистических, у нас нет.
И пока что не придвидится.

Да, сейчас много разговоров про квантовые вычисления, но квантовый компьютер это вообще другая кухня, для решения другого класса задач. Сейчас о них говорить не буду, а то тред затянется до ночи.

На практике недетерминизм удобен тем, что позволяет упростить картину детерминированного, но очень, очень, очень запутанного реального мира.

Мы не знаем, что именно лежит в базе данных, мы не знаем, что именно прийдет в запросе.
Мы исходим из того, что файла может и не быть в нужном месте.
Обычно мы рассматриваем весь рантайм программы, и компьютер в целом, и датацентр, и весь Интернет как недетерминированные системы.

Иначе нам было бы проще застрелиться, чем работать свою работу, и жить в этом мире, правда?

Остается очень вкусный и интересный вопрос, на закуску: я сказал, недетерминистические конечные автоматы можно перевести в обычные конечные автоматы, без потери скорости работы программы.
Можно ли также перевести НДТМ в ДТМ? Это будет эффективно?

Ответы: да, перевести вохможно; возможно ли эффективно - неизвестно, но скорее всего - нельзя;

Проблема заключается в различии между этими двумя моделями.
У автоматов нет памяти, они знают только свое текущее состояние, поэтому трансформация довольно проста.

У ТМ за счет бесконечной ленты в наличии бесконечная память, и из-за этого любая попытка свести НДТМ к ДТМ оканчивается экспоненциальным взрывом времени работы программы.

Проблема этого перевода лежит в корне одной из главных проблем Computer Science, и, возможно, одной из главных научных проблем всего человечества на данный момент.

Речь идет о вопросе равенства или неравенства классов сложности P и NP.
Именно этот вопрос я собираюсь показать, осветить, рассмотреть в следующем треде.

Захватывающая штука, которая может повлиять на нашу жизнь кардинально: или очень расстроить, в лучшем случае, или поставить точку на нашей цивилизации, в случае похуже.

Вот, в целом, и все)
Я надеюсь, что вышло понятно и без особых перегибов. Как всегда, готов ответить на все возникшие вопросы!

@threadreaderapp unroll

• • •