Теория «компьютерных» строк. Недостаточная выразительность рекурсивных функций и арифметики для теории алгоритмов.

Д. Л. Гуринович

08 июня 2020 года

УДК 510.51

Содержание

Аннотация

The generally accepted modeling of algorithms by recursive functions is not adequate for algorithms that use large input data. In them, the complexity of the operation on numbers depends on the size of the numbers, what can exceed any function of the actual complexity of calculations.

A non-arithmetic string theory is constructed.

It received a number of unexpected assertion.

In particular, the insufficient expressiveness of arithmetic in the corresponding theory of algorithms is proved.

——

Общепринятое моделирование алгоритмов рекурсивными функциями неадекватно для алгоритмов, использующих большие входные данные. В них сложность операции над числами зависит от размера чисел, что может превышать любую функцию от фактической сложности вычислений.

Построена неарифметическая теория строк.

В её рамках получен ряд неожиданных утверждений.

В частности, доказана недостаточная выразительность арифметики в соответствующей теории алгоритмов.

1 О теории алгоритмов и сложности вычислений

I. Пример со взятием подстроки

При подготовке этой статьи меня попросили сделать общий обзор основ теории алгоритмов для тех читателей, которые не совсем в теме. Мои собственные впечатления достаточны для моих предпочтений и выбора моего направления исследования, но, чтобы их убедительно излагать – надо самому разбираться гораздо детальней, а обзор этого не предполагает

Поэтому в обзоре я опишу своё понимание при «широком» взгляде на практику теории алгоритмов. Буду давать идеи доказательств, как я их понял, не претендуя на правильность в каждом примере, но, чтобы «общая картина» давала правильные ориентиры тем, кто захочет исследовать вопросы основ теории алгоритмов детально.

Обзор – это «взгляд дрона» при полёте над математическими «джунглями». У каждого свой «дрон» и свой «полёт». Точность возникает при детальном рассмотрении каждого «дерева», когда ты пробиваешься через математические «джунгли» с «мачете» логики. Но за «деревьями» тогда трудно увидеть «лес».

То, что я делаю сам (а не рассказывая о сделанном другими) и на что опираюсь – это со второго раздела. И там я пишу с соблюдением правил логического вывода и готов отвечать за «деревья».

Но я не согласен отвечать за отдельные «деревья» в обзоре и прошу не воспринимать сведения из обзора как готовую для практического применения информацию.

Однако за придуманную при написании обзора теорему о наличии стандартной интерпретации я готов отвечать. И ради одной этой теоремы стоило прилагать усилия для написания обзора.

И ещё один момент. В обзоре ниже – много дискуссионного и отличающегося от того, что написано в учебниках. Теория алгоритмов – крайне молодая специальность в математике по меркам истории математики и там многое не устоялось. Поэтому в обзоре мне интересней разобрать логические неувязки в нынешнем построении теории алгоритмов, чем вникать в частные нюансы доказательств теорем. Потому что концептуальные вопросы построения новой (по меркам истории математики) теории являются наиболее приоритетными, на мой взгляд.

После сделанных предупреждений перехожу к обзору.

Прошли те времена, когда большинству людей компьютеры казались решением всех вычислительных проблем, разве что иногда надо добавить процессоров, памяти, быстродействия и всё будет посчитано – и недорого.

К настоящему времени широкое практическое использование получили электронные подписи, стойкие к «взлому» самыми мощными компьютерами, про вычисления выяснилось, что они могут быть дороже, чем решаемые ими задача. А маргинальный сосед с майнинговой фермой в подвале и разорением всего дома счетами за электроэнергию стал заурядным злодеем из новостей.

Есть затраты по времени, пространству и энергии для проведения вычислений. Разберём это на примере одного из базовых элементов компьютерного вычисления – взятия «подстроки» длиной в один символ (букву) из очень длинной «строки». Берем эту букву из ячейки памяти, и доставляем в центральный процессор.

Итак, рассмотрим получение i-го символа из строки a. Это будет подстрока длиной 1, и команда в тексте программы будет примерно такая:

$\mathrm{str}(a, i, 1)$

Сколько времени займет эта операция для получения соответствующего символа из ячейки памяти компьютера внутрь процессора? Для простоты считаем, что ячейки данных содержат по одному символу, первый символ расположен возле процессора, и все ячейки расположены на одной прямой на линейном расстоянии от процессора относительно своих номеров. До символа от процессора и обратно сигнал идет со скоростью света, затрачивая линейное время относительно i, пусть $k_1 \times i$. И требуется какое-то время c_1 на обработку символа процессором, ещё надо время c_2 на обработку сигнала ячейкой памяти. В сумме получаем $t = k_1 * i + c_1 + c_2$.

А если мы получаем 2 символа $\mathrm{str}(a, i, 2)$ с мест i и $i + 1$, то разумно предположить, что они придут в процессор по очереди, а ячейка $i + 1$ расположена на одну единицу расстояния памяти дальше ячейки i. И тогда на это получение уйдёт времени: $t = 2 * (k_1 * i + c_1 + c_2) + k_1$

Простая, удобная модель, весьма близкая (для подобного радикального упрощения) к практическому устройству компьютеров. Будем считать такое расположение и взаимодействие процессора и строк моделью исполнения алгоритмов с «лучами данных». Сокращённо будем называть такую модель исполнения алгоритмов «Машина компьютерных алгоритмов», далее – МКА.

Кстати, нам нужен ещё один базовый элемент вычисления – отправка символа из процессора в ячейку памяти. Он будет примерно в 2 раза менее затратным, потому что управляющий сигнал и символ «летят» в один конец. Остальными элементами вычислений можно пренебречь при расчёте времени работы программы, если считать, что процессор работает только с «внешними» (из ячеек памяти) данными.

Да, процессор сравнивает символы из разных ячеек, генерирует новый символ порой по результатам этого сравнения, но каждая такая операция связана с получением данных из ячейки памяти и/или отправкой в ячейку. Поэтому если увеличить расчётное время получения/отправки «символа» c_1 на время самой долгой операции «внутри» процессора, то мы получим слегка огрублённое время, но пригодное в качестве консервативной (слегка завышенной) оценки времени, которое алгоритм истратит на свою работу при очередной элементарной операции.

Но на практике всё может немного отличаться от построенной модели работы алгоритмов. А из-за этого будут возникать расхождения с этой моделью в пределах полинома от времени и размеров данной модели или относительно времени и размеров другой, с которой сравниваем данную.

Например, наша строка располагается не на «луче данных» а в «шаре данных». Когда самые первые символы строки расположены в ячейках памяти вокруг процессора на некоторой ближайшей сфере, следующие после них символы – на сфере чуть дальше и т.д. Тогда расстояние от процессора до нужной позиции в строке будет уже пропорционально примерно корню кубическому от i, и сигнал преодолеет его быстрее, чем в модели МКА. Но и в этом случае время работы t_0 нашей модели МКА будет превосходить время работы модели с «шарами данных» t_1 так, что это не превысит некоторый полином от t_1:

$t_1 \leq t_0 \leq \mathrm{p}(t_1)$

Ограничивающий полином, в данном случае, можно взять третьей степени и не больше. Заметим, что худший показатель эффективности (то есть – большее время и/или размер использованной для работы памяти) заключён в «клещи» неравенства, где ограничения рассчитаются от лучших показателей эффективности.

Тогда будем говорить, что эффективность моделей является сопоставимой (сравнимой) в полиномиальных пределах для данного алгоритма. Размер использованной памяти не может превышать времени работы, кстати, потому что для использования новой ячейки надо сделать хотя бы одну операцию в работе программы и истратить хотя бы один «квант» времени работы. Поэтому для оценки эффективности можно сравнивать только время работы алгоритма в одной модели исполнения алгоритмов с временем его работы в другой модели.

Однако эффективность моделей исполнения алгоритмов может варьироваться в зависимости от алгоритмов. Для одних алгоритмов эффективней может быть первая модель, а для других – вторая. Поэтому запишем систему двух более универсальных неравенств для полиномиальной сравнимости работы алгоритма в двух моделях исполнения алгоритмов:

$t_2 \leq \mathrm{p}_1(t_1) \wedge t_1 \leq \mathrm{p}_2(t_2)$

И если отличия по времени работы алгоритма в 2-х моделях исполнения алгоритмов оказываются в подобных полиномиальных пределах, одинаковых для всех алгоритмов (которые завершают

свою работу) и любых «входных» данных, то эти модели исполнения алгоритмов являются сопоставимыми (сравнимыми) в полиномиальных пределах.

Или короче – «полиномиально сравнимые модели исполнения алгоритмов».

Доставлять символы к процессору можно не по одному, а «пакетами», если пропускная способность канала между процессором и ячейками памяти высокая, но это тоже создаёт отличие с моделью «один символ за раз» только в пределах полинома.

Ещё один пример отличия в эффективности работы алгоритма в разных моделях. Сортировка «пузырьком» менее эффективна, чем метод слияния упорядоченных массивов для модели МКА. Но сортировка методом «слияния упорядоченных массивов» в модели МКА оказывается хуже, чем сортировка «пузырьком» в модели «двойная тележка Тьюринга», если тележка умеет переставлять за один шаг символы в соседних ячейках как положено для сортировки. И ездит от одного конца строки к другому, пока не упорядочит всё.

Дело в том, что сортировка «слиянием упорядоченных массивов» требует от процессора (в модели МКА) обращаться к каждому символу в среднем прямо пропорционально числу $\lg(n)$, где n – длина (количество символов) строки. И каждое «обращение» требует от сигнала преодолеть расстояние, пропорциональное в среднем $n/2$ символов (а в оба конца получается n). Речь идет о стандартном методе «слияния упорядоченных массивов», если не перемещать массивы ближе к процессору на время «слияния». То есть, такая сортировка потребует для всех символов количества единиц времени в пределах, пропорциональных $\lg(n) \times n^2$.

Слияние же «пузырьком» в модели «двойная тележка Тьюринга» приведёт к полному упорядочиванию массива за n проходов «двойной тележки Тьюринга» из конца в конец, потому что каждый проход увеличивает минимум на один элемент размер полностью упорядоченного «края». Пройти n раз n символов – и будет истрачено единиц времени в пределах, пропорциональных n^2.

Разница (точнее - отношение) между временами работы 2-х разных сортировок в разных моделях исполнения алгоритмов получается пропорциональным $\lg(n)$ в пользу «двойной тележки Тьюринга». Другое дело, что «тележка» может ехать медленнее, чем движется свет и ещё затраты времени на «попутные» проверки и обмены символами между соседними ячейками... Но для огромного сортируемого «массива» (для строки в разбираемом случае) этот $\lg(n)$ может оказаться как угодно больше отношения скоростей тележки и света. И в этом случае «черепаха» всё же может обогнать «Ахиллеса».

Просто обычно не помнят об ограниченности скорости света, но если мы рассматриваем вопрос о данных произвольного размера, то забывать об этом не следует, конечно.

У модели с «тележкой Тьюринга» тоже не всё идеально. Для случая «произвольного доступа» она не может так быстро «перепрыгивать» от одних данных к другим (с одного «луча данных» на другой «луч данных»), как это возможно в модели МКА с «лучами данных» и центральным процессором.

Кстати, насчёт размера ячеек памяти, уменьшение размеров которых, вроде бы, могло устранить проблему с расстоянием и ограниченной скоростью сигнала – тут тоже есть ограничения – квантовые ограничения на размер.

В реальном мире для данных, которые могут иметь произвольный объем, невозможен «произвольный доступ» в обычном толковании этого понятия из учебников программирования. Чтоб за количество «единиц времени», пропорциональное $\lg(n)$, получить любой заданный элемент из n имеющихся – такое в реальном мире невозможно. В лучшем случае – время доступа, пропорциональное корню кубическому от n – в нашем 3-мерном мире с ограниченной скоростью сигналов и квантовыми ограничениями на размер ячеек памяти, внутренний импульс которых ещё не разносит систему в щепки.

II. P-алгоритмы и их «системы отсчёта»

Если мы согласны с тем, что можно использовать любую модель исполнения алгоритмов среди всех моделей, полиномиально сопоставимых с некоторой «эталонной» (условно) моделью, то можно взять за основу модель МКА в качестве основной модели исполнения алгоритмов. И показатели эффективности для алгоритмов в этой модели будут в полиномиальных пределах при сравнении с аналогичными параметрами эффективности в других эффективных моделях работы алгоритмов.

Этот выбор мы вынуждены сделать, потому что расчёт времени работы алгоритма существенно зависит от того, в рамках какой модели мы рассматриваем работу данного алгоритма. Брать же за основу модель «машина Тьюринга» нет оснований – я не знаю ни одной вычислительной системы на практике, которая работала бы как машина Тьюринга. Да и вычислять время «поездок» тележки Тьюринга из места её текущей дислокации в нужное место – куда труднее, чем отсчитывать передачи сигналов и данных единообразно – относительно центрального процессора.

Итак, под «полиномиальной сравнимостью» с МКА для всех видов вычислений мы понимаем то, что любой метод вычислений может быть исполнен в модели МКА в пределах полиномиального времени от времени «оригинального» вычисления. При этом, если «оригинальное» вычисление является практически осуществимым (не слишком долго в практическом плане – Солнце ещё не погаснет, а компьютеры не заржавеют), то и соответствующее вычисление в модели МКА тоже – практически осуществимо, при некотором физически возможном быстродействии.

Разумеется, нельзя доказать, что «полиномиальная сравнимость» с моделью МКА имеет место для всех эффективных моделей исполнения алгоритмов, потому что мы можем не знать чего-то крайне необычного и эффективного в вопросе способов выполнения вычислений. Но в качестве тезиса, расширяющего тезис Чёрча (применяемого теперь не к машине Тьюринга, а к МКА), предыдущий абзац можно использовать, вроде бы. Но это вопрос дискуссионный, как и всякий тезис без формального доказательства, конечно. И не факт, что он пройдёт испытание временем.

Но вопрос в том, почему нам интересна именно полиномиальная сравнимость между моделями исполнения алгоритмов?

В принципе, этот вопрос выходит за рамки темы данной статьи. Можно просто перейти к рассмотрению математических теорий, в рамках которых делались попытки вычислять время работы алгоритмов и строить ту теорию, в которой такие расчёты выполнить всё же можно.

Но чтоб не оставлять пробел в изложении, коснёмся и поставленного сейчас вопроса.

Полиномиальная сравнимость моделей исполнения алгоритмов интересна нам потому, что она никак не меняет характер «полиномиального времени» работы алгоритмов – если у алгоритма именно «полиномиальное время» работы.

Что такое «полиномиальное время» работы алгоритма A? Полиномиальная сравнимость моделей – это сравнение. А полиномиальное время работы алгоритма – это ограничение количества единиц времени t работы алгоритма неким полиномом от размера $\mathrm{p}(|s|)$ используемых им входных данных s:

$t_A \leq \mathrm{p_A}(|s|)$

Вот в разобранных примерах с сортировкой символов в строке строка и являлась «входными

данными».

Далее для краткости алгоритмы с полиномиальным временем работы будем называть P-алгоритмами. Едва ли в контексте теории алгоритмов можно спутать латинское «P» (Polynom) с русским «P» (Русским).

Метатеорема о неразрешимости временной сложности алгоритмов.

Вопрос о полиномиальном времени работы алгоритма не разрешим, так как не разрешима проблема остановки. То же верно и для других случаев временной сложности алгоритмов.

Действительно, пусть наш алгоритм получает строку длиной n символов, запускает на выполнение некий «неизвестный в плане остановки» алгоритм на n шагов и проверяет – остановится ли он. Если он не остановился, то наш алгоритм принудительно его останавливает, и завершает свою работу. Исходная «входная» строка стала результатом работы. Но если «неизвестный в плане остановки алгоритм» остановится за n шагов, то наш алгоритм повторяет в луче данных полученную «на вход» строку 10^n раз и завершает свою работу. Способа решить вопрос об остановке, нет, вообще говоря. Значит, нет и способа решить вопрос о полиномиальном времени работы нашего алгоритма, вообще говоря. Теорема доказана.

Так вот, полиномиально сравнимые модели исполнения алгоритмов, включая МКА — это те «инерциальные системы отсчёта» теории алгоритмов, при переходе между которыми P-алгоритмы остаются P-алгоритмами.

Разумеется, при переходе от одной модели исполнения к другой (полиномиально сравнимой) у алгоритма может меняться степень полинома (аргумент в полиноме – размер используемых алгоритмом «входных» данных), который ограничивает время его работы. Но само ограничение полиномом (пусть другой степени) никуда не исчезает и поэтому P-алгоритмы остаются P-алгоритмами.

Теперь о том, почему P-алгоритмы считаются «пограничными» между доступными для исполнения (их считают доступными) на практике и недоступными на практике. Хотя «на практике» тут всё равно с несколько абстрактных позиций, как увидим. Попробуем для начала «подогнать» решение под ответ из учебника, а затем найдём ответ не из учебника, а правильный.

Решение, подогнанное для «учебника»

Допустим, тебе необходим размер l_1 для объекта (l_1 – размер «входных» данных), который противостоит взлому от противника с вычислительной мощностью, равной твоей. Допустим, этот объект обладает полиномиальной сложностью (стойкостью) перед взломом. То есть, алгоритм для его взлома – это P-алгоритм со степенью m у его ограничивающего полинома.

Вопрос: какой размер l_2 у подобного объекта (с той же стойкостью к взлому) должен быть, чтобы противостоять более мощному противнику, чем ты? Если его мощность равна 2-й степени от твоей. Как тогда должен вырасти размер объекта для его стойкости к взлому со стороны этого мощного противника?

$l_2^m = (l_1^m)^2$

$l_2 = l_1^2$

Как видим, если ты используешь объект с полиномиальной сложностью к «взлому», то против атаки противника, степень мощности которого 2 от твоей, тебе придётся увеличивать размер объ-

екта соразмерно – квадрат от размера, пригодного против равного тебе противника. То же верно и для любой степени n вместо 2. А при такой «игре на равных» ты неизбежно проиграешь более мощному противнику – исчерпаешь свои ресурсы раньше.

Если же у тебя степенная (экспоненциальная) сложность объекта к взлому, то против мощностей противника равных твоим в степени n, тебе надо всего лишь линейно от n увеличить размер объекта в сравнении с противником равной тебе мощности.

Уходя от полиномиальной сложности к более серьёзной, удаётся избежать необходимости давать противнику соразмерный его атаке ответ.

Вроде, ответ получился «как в учебнике», молодец, 5 балов и вопрос решён?

Решение с позиции здравого смысла и о P-языках

Но в «решении для учебника» есть пара нюансов и первый – это размер l_1, который труден для взлома противником с равной тебе вычислительной мощностью. А дальше идут рассуждения про твои трудности наращивать размер этого объекта из-за проблем с хранением и использованием этого объекта. Но трудности возникают для тебя не с размером l_1, а только с размером i_1 (например), который сильно больше, чем l_1 при достаточно большой степени у полинома, характеризующего полиномиальное время алгоритма взлома.

Потому что размер объекта, который труден для взлома, и размер объекта, который труден для хранения и использования – это разные (сильно) размеры.

На это ещё можно возразить, что в математике мы часто считаем всё, что ограничено на фоне возможной бесконечности – частными случаями, которые никак не отменяют того, что «нет возможности в общем случае». Поэтому считаем, что расстояние от l_1 до i_1 твой мощный противник преодолеет – «как-то», а вот дальше уже начинается гонка на равных, где у тебя, как слабейшей стороны, нет шансов. Хотя на практике для преодоления расстояния от l_1 до i_1 противнику может не хватить всего кремния на Земле для процессоров, конечно.

Но есть и второй нюанс – несоразмерность наращивания защиты и атаки для сохранения равновесия. Да, степень наращивания одна и та же, но вот только противник должен прилагать эту степень к вопросу, где операций надо – степень m от твоего усилия по увеличению размера объекта. Выбери m побольше и ресурсов твоей защиты тебе хватит на ближайшую сотню лет с лишним и гарантией.

Поэтому «решение для учебника» является, при ближайшем рассмотрении, опровержением мнения, что P-алгоритмы – исполнимы на практике. В общем случае они не исполнимы – не дождёшься результата. Не в этой жизни, по крайней мере.

Но дело тут в другом – в P-языках, на самом деле.

Ведь откуда берётся этот объект, который можно взломать P-алгоритмом? На практике такие объекты и появляются от этих же P-алгоритмов. И называются «слово языка класса P». И дело вовсе не в трудности хранения и использования, а в том, что размер соответствующего объекта для тебя ограничен трудностями его получения, а вовсе не трудностями его хранения и использования.

Поэтому то, что P-алгоритмы считается приемлемыми для взлома объектов, связано с тем, что ты сам создаёшь эти объекты при помощи такого же P-алгоритма. И поэтому у тебя просто

нет ничего слишком сложного для взлома. И если степень m у полинома, ограничивающего P-алгоритм, большая – то и объекты у тебя – короткие. Вот такое объяснение правильное, как мне представляется.

Вот по указанной сейчас причине P-алгоритмы и считаются «граничными» по их возможности исполнения на практике – и нужно понимать, что «предел применимости» такого подхода – это только в отношении объектов, чьё создание требует затраты таких же (или близких) усилий, что и их взлом.

И ещё один вывод для упрощения расчёта того, сколько времени работает P-алгоритм:

Раз речь идёт о практически коротких объектах, то в расчётах времени работы P-алгоритма можно исходить из предположения о бесконечной скорости света. Потому что расстояние до каждой ячейки памяти от процессора небольшое, и можно оценить время доступа к любой ячейке памяти на основе времени доступа до самой дальней из всех ячеек памяти. И за счёт этого считать время доступа к любой ячейке константой. Но «предел применимости» такого приближения – ограниченность размеров используемых объектов неким фиксированным пределом, далёким от светового года, конечно.

Сказанное мной тут выше про «двойную тележку Тьюринга», про причины «граничности» P-алгоритмов, и про неразрешимость вопроса о полиномиальном времени алгоритма, я нигде не читал и придумал при подготовке данной статьи к публикации (про неразрешимость вопроса о полиномиальном времени я придумал несколько лет назад в дискуссии на одном математическом интернет-форуме). Поэтому всё перечисленное ещё нуждается в дополнительной проверке «со стороны».

Про учёт ограниченности скорости света при вычислении времени работы алгоритма много где написано, но в математических моделях вычислений это никогда не учитывалось – из того, что я читал. Про модель МКА для исполнения алгоритмов не уверен, что сам придумал – обычное для меня с прошлого века представление о работе компьютеров, не помню от кого.

III. «Кротовые норы» под алгоритмами взлома. Пример – RSA

Однако, кроме алгоритмов создания объектов, «симметричных» к алгоритмам взлома, есть и способы создания объектов, асимметричные к алгоритмам взлома. И алгоритмы создания таких систем работают радикально быстрее и менее затратно (по деньгам, энергии, материалам и остальному) чем доступные для твоего «противника» алгоритмы взлома созданных тобой объектов. Хотя тут есть нюанс, о котором скажу ниже.

Пример – имеющие сейчас широчайшее практическое применение «криптосистемы с открытым ключом». Система RSA стала классикой подобных систем, «Объектом» в системе RSA является число (пусть m), записанное в позиционном (пусть десятичном, например) виде. Обычно это число представляет собой произведение двух «секретных» простых чисел (пусть p_1 и p_2). Ещё есть число e – «обычная» часть соглашения о шифровании без особого «секрета». И служат эти 2 числа ключом для превращения текста t в зашифрованное сообщение s при помощи практически быстрого алгоритма:

$s = f_{e,m}(t)$, где $m = p_1 \times p_2$, где p_1 и p_2 – секретные для окружающих простые числа. При этом удобно, чтобы $t < p_1$ и $t < p_2$.

Конкретнее:

$f_{e,m}(t) = (t^e \, mod \, m)$

Возводить в степень и брать остаток от деления на m весьма просто, потому что

$(x^2 mod \, m) = (((x \, mod \, m)^2) mod \, m)$

То есть – это как обычное возведение в степень, но только результат всегда можно привести к числу, меньшему m. Поэтому брать любые такие степени можно очень быстро, записав показатель степени в двоичном виде и используя вот такую схему на примере степени $2^4 = 16$:

$x^{16} = (x^8)^2 = ((x^4)^2)^2 = (((x^2)^2)^2)^2$

Как видим, тут просто умножение x на себя с присваиванием переменной x результата произведения 4 раза. При этом с учётом $(mod \, m)$ каждое «произведение» остаётся меньше m.

Для дешифрования нужен d (секретный), такой что:

$de \equiv 1 (mod \phi(m))$ где $\phi(m)$ – функция Эйлера количества всех взаимно простых с числом m чисел, которые меньше m. Для $m = p_1 \times p_2$, где p_1 и p_2 – простые, $\phi(m) = (p_1 - 1) \times (p_2 - 1)$.

Функция Эйлера $\phi(m)$ даёт количество всех возможных остатков при делении числа s, меньшего, чем p_1 и p_2 (поэтому взаимно простого с m). Ну, или при делении степени i числа s, которое (числа s) меньше, чем p_1 и p_2.

При возведении в степень $1, 2, \ldots \phi(m)$ такого числа s результат «пробегает» (беспорядочно) все возможные взаимно простые с m остатки и они все попарно отличаются, иначе разница разных степеней будет кратной m, что докажет делимость s на m. Теорема Эйлера:

$s^{\phi(m)} \equiv 1 (mod \, m)$, где число s и число m – взаимно простые.

Вот после степени $s^{\phi(m)}$ аналогия с «обычной» степенью заканчивается и результат (остаток) повторяется, что невозможно для «обычной» степени, конечно – так как для обычной степени результаты всегда растут, и не ограничены числом m.

То есть, в терминах «сравнений» $(mod \, m)$ это означает, что de на 1 больше, чем $\phi(m)$.

А такое de нужно для того, чтобы снова получить исходное сообщение t, потому что

$t^{\phi(m)} \equiv 1(mod\, m)$, где t взаимно простое с m.

То есть, условно говоря, в терминах сравнений:

$t^{de} = t^{\phi(m)+1}$, это не буквальное равенство! Но смысл в том, что мы умножаем $t^{\phi(m)} \equiv 1(mod\, m)$ на t. А значит, по теореме Эйлера:

$t^{de} \equiv t(mod\, m)$ – это и есть расшифровка.

То есть, для расшифровки сообщения у нас имеется соответствующий «штатный взлом»

$h_{e,m}(s) = (s^d\, mod\, m)$

Очевидно, что уровень сложности у шифрования и дешифрования – одинаковый. Размеры $|t|$ и $|s|$ - в пределах размера $|m|$. Существенно может меняться только степень (e и d), но она считается в пределах остатков, а поэтому с ростом степени количество операций растёт логарифмически, как видно из примера с x^{16} выше.

Аналогично работает и электронная подпись, только для электронной подписи исходный текст берется как s, затем он «расшифровывается» в t и это t становится электронной подписью. И каждый может проверить, что $s = f_{e,m}(t)$, но вот создать такую подпись для произвольного s – не могут без знания p_1 и p_2.

Проблема именно в том, что вопрос факторизации (разложения числа на простые сомножители) – трудный, иначе вычислить $\phi(m)$ и «подогнать» d так, чтобы:

$de \equiv 1(mod\phi(m))$

было бы просто. Хотя метод «подгонки» при известной $\phi(m)$ тоже не совсем на поверхности. Но есть простой способ (аналогичный методу Евклида вычисления наименьшего общего делителя) сокращения остатков, который соответствует умножению делимого и делителя на некие коэффициенты. В результате применения этого способа остаток доходит до 1 или (-1), а коэффициенты перемножаются, и возникает нужный d.

Хотя «нужный» d будет только при остатке 1, но при остатке (-1) и равенстве

$d_0 \times e - k_0 \times \phi(m) = -1$

мы легко получаем

$d_1 = \phi(m) - d_0$, $k_1 = e - k_0$

при которых

$d_1 \times e - k_1 \times \phi(m) = 1$

Ещё одна трудность – это генерация простых чисел. Ведь нам необходимо, чтобы создание «объекта» было гораздо проще, чем его взлом.

Но и тут есть методы – о них можно почитать, например, в книге «Введение в криптографию» под общей редакцией В.В. Ященко, глава 4, параграф 6 «Как проверить большое число на простоту». И хоть есть полиномиальный способ получения простых чисел, но практики используют неполиномиальный способ – потому что для практических нужд он удобнее и быстрее. И даже если бы был способ факторизации при помощи P-алгоритма с ограничивающим полиномом 100-й степени, то это никак не отменило бы возможность применять RSA.

Разумеется, есть и простой метод факторизации числа m – перебором чисел i от 2 до $m^{1/2}$ с

проверкой, делится ли m на i. Но это потребует экспоненциального количества операций от размера $|m|$ - если число m записано в десятичном, например, виде. Для десятичного вида m количество таких i будет ограничено функцией, пропорциональной $10^{|m|/2}$.

Вот приведённый пример с перебором – это пример того алгоритма, который не является P-алгоритмом и который не может быть исполнен до конца за практически приемлемое время даже для не слишком больших чисел. Именно из-за трудности факторизации (отсутствия известных быстрых алгоритмов для этого) система шифрования RSA и считается стойкой.

С другой стороны, нет доказательства отсутствия P-алгоритма факторизации числа, записанного в «-ичном» (десятичном, например) виде с приемлемой для практического использования степенью m ограничивающего полинома. И не факт, что такой алгоритм не будет найден со временем, а какие-то критически важные для работы государств и жизни обществ системы не будут при этом взломаны злоумышленниками.

В шутку говоря, практики варят грибы, но никто точно не знает – что это за грибы и надо ли их варить. Но все едят.

И не нужно абсолютизировать, конечно, вопрос теоретической стойкости. На практике такая стойкость имеет свои пределы применимости. И упаси Бог, без учёта реальных обстоятельств, опираться на выводы о теоретической стойкости каких-то систем к взлому. У государства, например, всегда были методы взлома, далёкие от вычислительных. Так, например, хранитель монетного двора Англии, сэр Исаак Ньютон, законно и успешно использовал такие методы тоже для низведения фальшивомонетничества от массовости к ничтожеству.

И возможные протесты фальшивомонетчиков о «не математических методах сэра Ньютона» не имеют никакого значения в сравнении с тем, что такие методы есть. Просто для таких методов «взлома» есть другие книги, а тут мы рассматриваем математические методы, что является только частью общей картины, конечно.

Я не буду тут давать общего определения для языков класса NP, но отмечу, что считается, что существование тех же криптосистем с открытым ключом, вероятность взлома которых практически нулевая, требует в качестве необходимого условия выполнения неравенства:

$NP \neq P$

Однако такое требование – избыточно с практической точки зрения. Для практических нужд достаточно, чтобы было выполнено следующее неравенство – это формула-шутка:

$P \neq p$

То есть, чтобы полиномиальный метод взлома нужных объектов имел степень полинома непреодолимо (на практике) большую, чем полиномиальный алгоритм создания нужных объектов – если P-алгоритм взлома существует.

Что касается языков класса P, то это те языки, которые «распознаются» P-алгоритмами. То есть, когда «на вход» алгоритму, распознающему язык L, подано слово s размера $|s|$, то за количество единиц времени в пределах некоторого полинома $\mathrm{p}(|s|)$ будет получен результат:

1, если s является словом данного языка L

0, если s не является словом данного языка L

Считается, что вопрос о сводимости произвольного языка класса NP к некоторому языку класса P проще всего решить через поиск P-алгоритма для решения произвольной конъюнктивной нормальной формы (далее – КНФ) логики высказываний. Если такого P-алгоритма нет, то верной является неравенство классов языков: $NP \neq P$.

Такое мнение связано с «теоремой Кука» – теоремой о сводимости произвольного языка класса NP к КНФ. В силу этой теоремы КНФ является одним из «NP-полных языков». Но теорема Кука – о языках, а мнение из предыдущего абзаца – об алгоритмах. И второе не выводится из первого.

Из-за того, что языки класса NP (и класса P, так как все языки класса P являются языками класса NP) сводимы к логике высказываний, кажется – при поверхностном взгляде – что вопросы об алгоритмах «распознавания» слов языка класса NP рассматривать не требуется. Это ложное впечатление. Алгоритмы проверки способны оперировать с текстами программ и результат их работы не может быть полноценно выражен на уровне логики высказываний. И в этом смысле теорема Кука сводит вопрос на уровень такой слабой выразительности (логика высказываний), на котором вопрос не может быть – скорее всего – полноценно исследован.

Поэтому нам нужна не просто теория, способная «представлять» алгоритмы на уровне арифметики, но более выразительная, как уже можно было догадаться, и как будет явно показано в следующем подразделе. А использование «сводимости» некоторых языков на уровень теорий со слабой выразительностью может лишь усложнить – скорее всего – рассмотрение вопросов о (не)существовании алгоритмов с нужными свойствами для теории алгоритмов.

Например, логики в своём исследовании теории и алгоритмов (вопрос о разрешимости, например) используют подстановку в функцию или предикат такого аргумента, который соответствует самой этой функции или предикату (Лемма о диагонализации, например в «Вычислимость и логика» Булоса и Джеффри). Но для этого необходима достаточная выразительность используемой теории. А в рамках логики высказываний ничего подобного выразить невозможно. Поэтому наличие NP-полных языков никак не упрощает ту теорию, которую нам необходимо построить и использовать для решения вопросов теории алгоритмов.

IV. Длинная арифметика - это не-арифметика

Размер представления числа – это не из арифметики, вообще-то. Если брать «стандартную интерпретацию» для арифметики, то в рамках этой интерпретации размер числа n оказывается на 1 больше ($n+1$) самого числа, потому что ноль тоже как-то обозначен. И для стандартной интерпретации арифметики алгоритм факторизации числа n – очевидно является P-алгоритмом даже для простого перебора всех чисел, не превышающих $n^{1/2}$, с проверкой на делимость числа n каждым таким числом.

Из предыдущего абзаца следует, что те объекты, с которыми мы имеем дело в системе RSA –равно как и в других криптосистемах с открытым ключом – не являются числами арифметики. Потому что обладают ещё и теми свойствами, которые отличаются от числа арифметики. Размер числа при десятичном представлении числа пропорционален логарифму числа, например. Но свойства «размер» нет в арифметике – равно как и множества других свойств, которые мы используем в своей практике работы с алгоритмами – в том числе в криптографии.

Стандартная интерпретация для арифметики – это представление чисел в виде «счётных палочек», где число 10 выглядит примерно так:

11111111111

Записано, кстати, при помощи 11 (Одиннадцати) единиц, потому что для нуля тоже необходимо какое-то видимое изображение.

В основе всех чисел лежит именно стандартная интерпретация. Арифметика является теорией для чисел в любых формах – в том числе и в стандартной интерпретации. И арифметика описывает те свойства чисел, которые общие для всех их представлений. Ничего от специфического представления (от позиционного представления, например) в арифметике – нет. Иначе она не годилась бы для стандартной интерпретации. Мы – можем записать в позиционном десятичном представлении число 10 таким образом:

10

Но наши условности – как это представление соответствует вышеприведённому представлению 11111111111 – не являются частью арифметики, и не могут являться частью этой общей теории о натуральных числах.

Поэтому, если бы в рамках арифметики мы могли доказать какое-то свойство или провести какое-то действие, характерное именно для позиционного представления чисел, а не для стандартной интерпретации, то это означало бы, что арифметика не годится для стандартной интерпретации. Но именно для стандартной интерпретации она и создана в первую очередь.

Поэтому те вычисления, которые используются в криптографии с числами в позиционном представлении – это не арифметика, и они не могут быть формализованы в арифметике так, чтобы операции с ними отличались от операций с числами в стандартной интерпретации. А поэтому и факторизация в рамках арифметики не может быть доказана как «трудная», потому что для стандартной интерпретации факторизация осуществима простым P-алгоритмом.

Поскольку речь идёт о числах в позиционном представлении большого «размера», то операции с ними (умножение, возведение в степень, взятие остатка от деления и т.д.) называются «длинная

арифметика». Так вот, длинная арифметика – это не арифметика.

Таким образом, в арифметике нет возможности логически выделить то, что является «длиной строки», например, из-за того, что любое определение никак не меняет логику теории, лишь добавляя обозначение. Не имеет значения, какие слова русского языка мы используем, называя это обозначение – если там нет логического отличия от аналогичного определения иной «длины строки», например.

Действительно, то, как мы используем арифметику для расчёта длины некоторой строки, всегда можно использовать для работы не с данной строкой, а с числом в стандартной интерпретации, чьи «символы» будут просто результатом такого же расчёта, который мы используем в арифметике применительно к «заменителям» строк. И любые алгоритмы, работающие со строками, мы всегда можем заменить на соответствующие алгоритмы, работающие с числами в стандартной интерпретации.

И результат работы этих алгоритмов (строковых и заменяющих их «числовых») – один и тот же с точки зрения арифметики, и с точки зрения использования арифметики. Таким образом, доказана:

Метатеорема об обязательном наличии стандартной интерпретации.

Для любого использования арифметики применительно к строкам найдётся точно такое же использование арифметики применительно к решению такого же вопроса, но с заменой строк на числа в стандартной интерпретации.

Будем для краткости называть эту метатеорему «теорема о наличии стандартной интерпретации».

Эта теорема – в каком-то смысле аналог теоремы Левенгейма-Сколема, но только не в отношении обязательного наличия счётной интерпретации для любой теории первого порядка, а в отношении обязательного наличия стандартной интерпретации для любых задач, решаемых при помощи арифметики.

Но для «альтернативного» алгоритма, работающего со стандартной интерпретацией чисел, размеры этих чисел оказываются экспоненциально больше, чем для исходных строк многосимвольного алфавита или чисел в позиционном представлении. Поэтому понятие «длина», рассчитанное для чисел в позиционном представлении, оказывается несравнимым даже в полиномиальных пределах с размерами строк в иной возможной интерпретации – которая совершенно неотличима от исходной с точки зрения арифметики.

Тогда почему мы считаем введённую в арифметике при помощи определения функцию (если мы определяем такую функцию) «длиной строки»? Лучше назовём её «количество кружек пива». Даже если нам попробуют возражать, что в арифметике нет пива, то ведь и длины строк там тоже нет. Всегда есть неполиномиально отличающаяся альтернативная интерпретация. А «количество кружек пива» - более симпатичное название. Это шутка, конечно, но «длины строк» в арифметики нет так же, как и пива.

Но зачем заменять арифметику на более выразительную теорию, которая способна отличить числа в позиционной записи от чисел в стандартной интерпретации? Ведь отличие для пользо-

вателя арифметики обычно наглядны и очевидны и без теории? Заменять надо потому, что для простых практических программ многое наглядно и очевидно, но стоит перейти к «простейшим» десятиэтажным теоретическим построениям и всё «очевидное» почти всегда теряется без остатка. И вот тут путать позиционное представление чисел со «стандартной интерпретацией», размеры при которой экспоненциально больше – такое оказывается совсем не лучшей «математикой».

Примат целостности и теории для квази-строк

Кроме того, построенная на базе натуральных чисел теория не может быть полноценной теорией строк, потому что для чисел и объектов, с которыми оперируют «рекурсивные функции» верен «примат целостности».

А именно, на уровне строк мы имеем возможность получить часть содержимого строки, не имея всей информации о строке. В арифметике такой возможности нет – чтобы узнать о «содержимом числа» (если мы договоримся, что «упаковано» внутри данного числа) – надо прежде прочитать всё число до конца. Но доказано это будет только в подразделе VI раздела 2 данной статьи, когда появится возможность опереться на теорию строк. В принципе при желании можно смотреть это доказательство и сейчас – идеи там вполне простые (простые после того, как были найдены, конечно).

Поэтому оговорка «для любого использования арифметики применительно к строкам» в теореме о наличии стандартной интерпретации является существенной – речь не о любых алгоритмах, а только о таких, которые используемые ими данные используют целиком. То есть – некоторые из входных аргументов могут не использоваться, но те, которые алгоритм всё же «читает» - он «читает» до конца. Варианты «частичного чтения» никто просто не рассматривал (из того, что я читал), но на такое арифметика не способна – см. подразделе VI раздела 2 данной статьи.

Если мы строим теорию для строк (ради чего я написал эту статью), то тогда примат целостности преодолевается. Но можно попытаться построить промежуточную теорию между арифметикой и теорией строк – где примат целостности действует, но объекты («квази-строки») можно записывать с определенной логикой «длины строки» и гарантировать отсутствие альтернативной «стандартной интерпретации» с экспоненциально большими размерами объектов в сравнении с необходимыми.

Таким образом, на основе чисел (минимально расширяя арифметику «в направлении» строк) мы можем попытаться построить теорию квази-строк, которые не преодолеют «примат целостности», но будут способны представлять объект (квази-строку) как «упаковку» для символов, которые определенным образом пронумерованы.

Попытки в этом направлении делаются в некоторых книгах по теории алгоритмов. В следующем подразделе мы разберем, как строке ставится в соответствие число, соответствующее p-ичой записи с цифрами-символами из алфавита с p символами:

$\mathrm{c}(\alpha) = i_0 + i_1 \times p + \ldots + i_s \times p^s$. Где строка α имеет вид конкатенации символов из алфавита:

$\alpha = a_{i_s} \cdot \ldots \cdot a_{i_1} \cdot a_{i_0}$

И это рассматривается, как представление для строки средствами арифметики, что не может быть успешным на базе простой арифметики, как ясно из теоремы о наличии стандартной интерпре-

тации. Но можно попытаться «немного» расширить арифметику до уровня хотя бы «квази-строк».

Но квази-строка в любом случае не является числом арифметики. И если использовать квази-строки, то для них тоже надо построить соответствующую теорию первого порядка с равенством и применять её для работы с соответствующими объектами. Если не ошибаюсь, то квази-строки можно реализовать на практике. Могу предложить, например, следующую модель:

В квази-строках нет символов, расположенных друг за другом в линейном порядке. «Символы» в квази-строке лежат в «куче», но к каждому символу прикреплён ярлык с его номером. Это может быть степень десяти, если речь идёт о десятичном представлении числа при помощи квази-строки, например. И чтобы найти нужный символ в этой «куче» квази-строки, нужно сначала получить всю эту «кучу», а затем «рыться» в ней, в поисках квази-символа с нужным номером.

Если бы я пытался строить теорию первого порядка для квази-строк на основе натуральных чисел и выделенной «ичности», то первым делом я ввёл бы в рассмотрение такую константу, которая выделяла бы конкретную «ичность» (десятичность, например) в сравнении как со всеми иными «ичностями» при позиционном представлении чисел, так и в сравнении со «стандартной интерпретацией».

В теории строк, которая приведена в данной статье дальше, есть аксиома:

(2.6) $i \neq \ominus$, никакое число (включая ноль) не является пустой строкой.

И вот используя этот «особый объект» (новую константу теории) в привязке к квази-строкам выделенной «ичности» мы получим ту логику, которая выделяла бы одну «ичность» и позиционное представление чисел среди всех прочих представлений. И потребовалась бы ещё аксиома «привязки» данной константы к любой конечной квази-строке. В теории строк далее есть такая аксиома о конце строки, для случая, когда у строки есть конец:

(6.1) $(i \neq 0 \wedge \mathrm{str}(a, i, 1) = \ominus) \Rightarrow \mathrm{str}(a, i', 1) = \ominus$

Вот для теории квази-строк потребовалось бы как-то модифицировать аналог р-ичой записи с цифрами-символами из алфавита с p символами:

$\mathrm{c}(\alpha) = i_0 + i_1 \times p + \ldots + i_s \times p^s$,

Пустая строка – это то, что остаётся от строки, если удалить из неё все символы алфавита. Вот эту «оболочку» я бы и попытался использовать при построении теории квази-строк в дополнение к числам тогда. На примере десятичной системы. Вот такая конструкция:

$100{*}\ominus$, будет обозначать квази-строку «00» из двух цифр 0 (самого первого символа алфавита). А вот в случае замены одной из цифр 0 на другую цифру на соответствующем месте – меняется и коэффициент при степени 10. Например, для квази-строки «89» будет такое обозначение

$98 * \ominus = (8 \times 10^0 + 9 \times 10^1) * \ominus = (8 \times 10^0 * \ominus) + (9 \times 10^1 * \ominus)$

Если у нас квази-строка «99», то будет:

$(9 \times 10^0 * \ominus) + (9 \times 10^1 * \ominus)$

У каждого квази-символа в квази-строке есть свой номер в алфавите, и он виден в позиционной записи как цифра. А степень 10, на которую эта цифра умножена, указывает номер данного квази-символа в квази-строке. И нумерация «места» в квази-строке в данном случае начинается от нуля включительно.

Нумерация мест (степеней при 10) внутри квази-строки должна быть сплошной, должны быть прописаны равенства для квази-символов из квази-строки с разных мест, но одинаковых коэффициентах. Например, из последнего представления «99»

первый квази-символ в квази-строке: $(9 \times 10^0 * \ominus)$, а второй $(9 \times 10^1 * \ominus)$. И как символы алфавита они должны быть равны.

У первого квази-символа алфавита (цифра 0 в разбираемом примере) роль «наполнителя» квази-строки – он везде, где коэффициент при степени 10 оказывается 0.

Затем надо прописать правила конкатенации, особенности операций с пустой строкой, номер для строки из пустого символа $(0 * \ominus)$ и, может быть, получим теорию квази-строк на базе выделенной «ичности». А десятичность в разобранном примере выделена, потому что символы алфавита оказываются совпадающими (соответствующая аксиома равенства) именно с точки зрения 10-ного представления числа, когда коэффициенты при степенях 10 совпадают.

То есть, добавляя в арифметику логику до логики квази-строк мы создаём «дополнительную дискретность». Сверх обычной для натуральных чисел дискретности, мы ещё выделяем некоторую р-ичность, которая создаёт деление квази-строк на «неделимые» с точки зрения этой дополнительной дискретности «единицы» - квази-символы.

Даже при простом перечислении наиболее очевидных идей построения теории квази-строк видно, что логика построения теории квази-строк – далеко не тривиальная, там есть много чего сверх логики арифметики, а что получится при реальном построении, когда начнешь пытаться доказывать известные программистам свойства строк (точнее, квази-строк в данном примере) на основе аксиом – неизвестно. Может, будешь ещё месяцев 20 переделывать аксиоматику и переписывать доказательства в свободное от основной работы и отдыха время – как это было у меня с теорией строк, излагаемой в данной статье.

Желающие могут попытаться пройти этот путь и доказать, что полученная ими теория действительно отличается от арифметики и на неё не распространяется теорема о наличии стандартной интерпретации. Тогда будет получена теория квази-строк – промежуточная теория между арифметикой и теорией строк. Но я в этой статье строю теорию строк, чтобы иметь там все необходимые свойства объектов, с которыми работают алгоритмы, включая преодоления «примата целостности».

А у меня в теореме о локальном изменении данных доказано, что при определенных условиях то, что мы делаем с началом строки – никак не зависит от того, какой конец (хвост) у этой строки.

А для стандартной интерпретации невозможно началом строки адекватно выразить начало строки многосимвольного алфавита. Получается, что то, что доступно для теории строк с многосимвольным алфавитом недоступно для стандартной интерпретации. И вот тут есть принципиальная разница между попыткой теории строк многосимвольного алфавита извлечь начало строки из строки многосимвольного алфавита – это можно сделать локально – и попыткой извлечь ту же информацию, но «как-то» представленную посредством «стандартной интерпретации». Второй вариант потребует чтения всей строки. И вот это и есть – неприменимость теоремы о наличии стандартной интерпретации к построенной в моей статье теории строк.

V. Какие теории для представления алгоритмов в учебниках (логики и теории алгоритмов) и чего в них не хватает

Сейчас в учебниках вопросы вычислимости и алгоритмов в теоретическом отношении рассматриваются именно на основе арифметики. И «мостом» между алгоритмами (машиной Тьюринга, как правило) и арифметикой служат «рекурсивные функции». Однако, поскольку рекурсивные функции, соответствующие арифметике, способны лишь на то, что доступно при любой интерпретации – то алгоритмы, соответствующие рекурсивным функциям в учебниках являются алгоритмами, работающими со «счётными палочками», когда аргумент 10 записан на ленте данных вот так:

11111111111

Я проверил по книгам:

«Теория алгоритмов» В.И. Игошин, «Курс математической логики и теории вычислимости» А.С. Герасимов, «Вычислимость и логика» Дж. Булос, Р. Джеффри.

То есть, не идёт речь о позиционном представлении чисел, хотя все реальные компьютеры работают именно с позиционным представлением чисел.

Понятно, что такое содержание учебников связано с отсутствием теории строк (или хотя бы квази-строк) во время их написания. А в рамках арифметики (теории Пеано, например) просто нет средств, чтобы представить строки и записать числа именно в позиционном виде – так, чтобы арифметика «могла» отличить это от стандартной интерпретации.

Однако мне подсказали (кто и что ещё – в следующем подразделе) учебник, в котором автор предпринял попытку формализовать работу алгоритмов со «словами», что допускает, если всё доводить до конца, и работу с числами в позиционном представлении.

В учебнике А.И. Мальцева «Алгоритмы и рекурсивные функции» автор прилагает усилия по построению теорию «слов» (в данной статье я использую термин «строк») в параграфе 11 «Словарные множества и функции». Разберём эти усилия по подпараграфам параграфа 11.

Учебник А.И. Мальцева (параграфы 11 и 12) очень удобен для разбора и одновременного рассмотрения многих отличий между числами и строками, для которых нам надо построить теорию первого порядка с равенством (и, которая будет построена в этой статье).

Подпараграф 11.1. Словарные множества

Алфавит задаётся как множество символов («какой-нибудь конечный набор символов»). Но получается, что должна быть ещё и теория для символов? Но соответствующей аксиоматизации для символов автор не приводит. Но если строить теорию аккуратно, то надо тогда брать аксиомы для теории множества (это же теория первого порядка), добавить некоторые аксиомы для строк (слов, символов) – потому что не все множества являются строками – и строим «объединенную» теорию. Но вместо этого в подпараграфе 11.1 строке («слову» в терминах подпараграфа) ставится в соответствие число, соответствующее p-ичной записи с цифрами-символами из алфавита с p символами:

$c(\alpha) = i_0 + i_1 \times p + \ldots + i_s \times p^s$. Где строка α имеет вид конкатенации символов из алфавита:

$\alpha = a_{i_s} \cdot \ldots \cdot a_{i_1} \cdot a_{i_0}$

Раз аксиом для строк нет, а есть аксиомы о числах, то, наверно, надо опираться на арифметику и её аксиомы? Но это не может увенчаться успехом, потому что в арифметике невозможно вывести количество «цифр», которые используются для записи числа в позиционной («ичной») форме записи – смотри теорему о наличии стандартной интерпретации в предыдущем разделе.

А «самое стандартное» представление чисел для арифметики (стандартная интерпретация) – экспоненциально велико по своим размерам в сравнении с теми представлениями, которые необходимы нам в практической криптографии и теории алгоритмов.

Фактически, при отсутствии «дополнительной дискретности» любое «ичное» разбиение числа «рассыпается» на более «мелкое». При переходе от 256-ричного представления к 2-ичному прежние границы ещё имеются – это границы «байтов», которые в свою очередь состоят из «битов». Но при переходе к троичному представлению прежние границы 256-ричного представления «стираются» и заменяются новыми. А при переходе к стандартной интерпретации всё рассыпается до уровня «счётных палочек».

Вот что происходит, когда нет дополнительной – в сравнении с арифметикой – логики для выделения некоторой «ичности».

Подпараграф 11.2 Основные словарные операторы

Здесь вводится в рассмотрение «рекурсивные словарные функции» и формулируются методы их построения.

Но это – не теория, а описание некоторой модели построения и работы для словарных функций – (частично) рекурсивные словарные функции. Точно так же, как рекурсивные функции для чисел – это совсем не теория, а модель построения и исполнения некоторых функций натуральных чисел. Так как имеется теория для натуральных чисел, то рекурсивные функции оказываются одной из «интерпретаций» для арифметики. А если мы ещё не знаем никакой теории, соответствующей данной модели, то эта модель – ещё не является интерпретацией ни для какой нашей теории.

Ведь как обычно происходит создание теории: есть какие-то события в жизни, которые мы как-то практически записываем, прогнозируем, замечаем закономерности. Затем мы их обобщаем – это уже некоторая модель. А уже после этого мы составляем теорию.

Если мы возьмём рекурсивные функции – то там нет ничего ни про коммутативность сложения, ни про дистрибутивность умножения, ни про индукцию и нет вообще ничего теоретического. В теории мы новые функции определяем совсем не так, как строим их в модели. Ну, и ещё миллион отличий. Чтобы построить на основе модели теорию, нам надо применять свою неформальную способность к обобщению, изобретательность, фантазию, широту охвата проблемы и т.д. От модели до новой теории, делающей эту модель своей интерпретацией, расстояние – как «до неба».

И, кстати, из модели можно извлечь далеко не всю необходимую информацию об объектах, которые в ней используются. Например, если бы из модели рекурсивных (числовых, а не словарных) функций можно было бы извлечь информацию для теории, то это была бы даже не арифметика Пеано, а сильно урезанная в сравнении с теорией Пеано теория Q. А теория Q, как доказано – смотри Глава 14 «Представимость в Q» у Джеффри и Булоса в «Вычислимость и логика», например – не содержит даже предложения $\forall x \forall y\, x + y = y + x$ в качестве своей теоремы. И в ней нет индукции

и много чего ещё.

И, разумеется, приводимая автором модель является не моделью для строковых функций, а, строго говоря, моделью для квази-строковых функций.

Подпараграф 11.3. Прямое определение класса частично рекурсивных функций

Автор рассматривает ещё некоторые нюансы в развитие предыдущего подпараграфа.

Параграф 12. Машины Тьюринга

Здесь автор приводит теоремы соответствия между функциями, вычислимыми машиной Тьюринга и вычислимыми «словарными» рекурсивными функциям. Да, это не то же самое, что представимость работы машины Тьюринга в соответствующей теории строк. Но это – соответствие между моделями, одна из которых – модель для квази-строк. Соответствие подобно тому соответствию, что строится между функциями, вычислимыми на абаке и функциями, вычислимыми по Тьюрингу.

И то, что автор изложил логику соответствия со словарными рекурсивными функциями – его достижение в сравнении с другими авторами, которые ограничились соответствием числовых рекурсивных функций с функциями Тьюринга. Но если вопрос о представимости числовых рекурсивных функций в теории (арифметике) решен, то вопрос о представимости квази-строковых рекурсивных функций в теории квази-строк не решен. Но этот вопрос и не мог быть решён, потому что теории первого порядка для квази-строк (а тем более для строк) у автора учебника «Алгоритмы и рекурсивные функции» ещё не было.

Ещё есть 2 упущенных при сопоставлении моделей в «Алгоритмы и рекурсивные функции» (и других учебников) момента:

Во-первых, надо сравнивать эффективность вычислений при переходе от модели к модели – если мы сопоставляем их для теории алгоритмов, а не просто для вычислимости. Вдруг при каком-то переходе возникнет экспоненциальный рост времени работа алгоритма.

Во-вторых, надо сравнивать появление/исчезновение примата целостности. Этого соответствия вообще нет, кстати, при сравнении модели машины Тьюринга (или МКА) и рекурсивных функций (словарных или числовых – без разницы). Кажется странным, что доказательства соответствия проводятся, а «в засаде» всё равно остаётся какое-то не разобранное важное свойство, для которого соответствия нет.

Но то, что мы считаем совпадающим в каком-то отношении – является совпадающим лишь в степени выразительности нашего рассмотрения. Раз у нас рассматривались лишь целостные входные аргументы функций, то заметить, что в каких-то моделях нет ограничения «приматом целостности» - мы не в состоянии. А теперь, когда мы получим инструмент для рассмотрения теории и/или модели на ограничение «приматом целостности» - мы сможем отличать случаи с таким ограничением от случаев без него.

Я очень извиняюсь, если что-то из того, что отсутствует, по моему мнению, в рассматриваемом сейчас учебнике, имеется где-то в тексте «Алгоритмов и рекурсивных функций», но не замечено мной. И я ни в коем случае не считаю попытку, предпринятую в «Алгоритмах и рекурсивных функциях» ошибочной. А.И. Мальцев пошёл дальше, чем авторы других учебников, поэтому и

есть о чём дискутировать. Просто этот путь ещё не закончен, а я в этой своей статье стараюсь довести построение теории строк в качестве теории первого порядка с равенством до конца. Что вовсе не отрицает усилий других авторов, но просто надо предварительно рассмотреть, чего ещё не сделано раньше и что, поэтому, надо сделать сейчас. И я тоже вовсе не претендую на то, что сейчас я всё доделаю, и в математике больше не останется ничего, что нужно будет делать после.

Можно, конечно, строить и теорию квази-строк – что на начальном этапе построения мы видим в «Алгоритмах и рекурсивных функциях» (и не только там). Однако такая теория не преодолеет «примат целостности» арифметики. А нам необходимо преодолеть это ограничение, если мы собираемся рассматривать то, как можно получать данные с бесконечного луча данных, например. И в случае, когда алгоритм использует только начальную подстроку из строки во «входных» данных – нам тоже надо преодолевать «примат целостности».

А для теории алгоритмов и алгоритмов проверки принадлежности слова к языку класса NP, например, нам очень даже нужна возможность частичного чтения строки. Потому что бывают слишком длинные «сертификаты», которые алгоритм проверки не читает до конца именно из-за того, что они слишком длинные. И читать до конца он их не должен, потому что иначе алгоритм проверки превзойдет полиномиальное ограничение на время своей работы относительно размера проверяемого слова – которое может быть сколь угодно меньше полученного на «вход» (помимо слова) сертификата.

Арифметика является весьма выразительной теорией, способной решать вопросы вычислимости (вычислимости за конечное время) – но не может решать вопросы о скорости вычислений и размере данных. И именно из-за высокой выразительности арифметики создаётся иллюзия, что она не менее выразительна, чем строки. В отношении возможности объектов «упаковывать» в себе другие объекты – она действительно выразительна почти как строки. И числа арифметики к тому же можно поставить во взаимно однозначное соответствие со строками. Но вот в вопросе «примата целостности» - соответствия нет. Но это и не удивительно:

То, что мы можем поставить одни объекты во взаимно-однозначное соответствие с другими объектами – совсем не значит, что свойства этих объектом соответствуют друг другу. Это как инвентарные номера, привинченные к мебели фирмы и сама мебель. Никогда не считал, что офисное кресло, на котором я сижу в офисе, и его инвентарный номер в табличке у меня в компьютере обладают одинаковыми свойствами. Но между ними имеется взаимно-однозначное соответствие, что подтвердила и недавняя инвентаризация.

И понять, как отличаются между собой строки и числа, построить адекватную теорию для решения вопросов теории алгоритмов – задача более приоритетная с математической точки зрения, чем «знаменитые» вопросы типа $NP \neq P$. Прежде нужно навести «базовый» порядок, то есть – построить адекватную стоящим задачам и понятную теорию. Это банальное, нудное, но бесспорно приоритетное с точки зрения математики дело, в отличие от любых «голливудских спецэффектов», включая «дипломаты» долларов.

И эта теория должна быть способна анализировать и себя тоже, потому что теория нужна нам и для того, чтобы повышать эффективность в решении наших задач, улучшать работу алгоритмов

(находить более эффективные алгоритмы). То есть, вопрос о сложности применения теории стоит в ничуть не меньшей степени, чем вопрос о сложности работы алгоритмов.

Поэтому нам нужна такая теория, логика и язык которой не только правильно описывали бы работу алгоритмов, но чтобы то же самое получалось при применении теории к себе самой, при переносе её теорем в качестве неких улучшений в программные тексты. И при этом такое соответствие между языком теории и работой алгоритмов должно быть в практически приемлемых полиномиальных пределах между текстами теории, программными текстами и временем работы алгоритмов, включая алгоритмы проверки доказательств теории.

В принципе, сформулированное в предыдущем абзаце «техническое задание» является программой Гильберта, перенесённое из арифметики и логики в теорию алгоритмов. И из-за того, что в теории алгоритмов принципиальным является вопрос об эффективности вычислений, появились и соответствующие требования об эффективности (полиномиальных ограничениях).

Как известно, программа Гильберта оказалась чрезвычайно продуктивной для логики и арифметики. И нет оснований отказываться от её реализации в рамках решения задач теории алгоритмов.

Поэтому в данной статье строится именно теория строк, а не квази-строк. Потому что всё равно надо строить новую теорию первого порядка с равенством и оптимальный вариант – сразу строить теорию со всеми необходимыми возможностями, чем строить теорию с ограничением возможностей. Ограничение возможностей потребует неформального преодоления этих ограничений и/или необходимости построения – всё же – теории со всеми необходимыми возможностями.

V. Информационный контекст

Должен отметить, что всё введение было полностью переписано благодаря Сергею Витальевичу Знаменскому. Он:

1. Посоветовал инструмент Гугла (я не пользовался Гуглом) для подборки нужных мат. статей;

2. Разобрал мою путаницу – поэтому главную статью из подборки п.1 я все же прочитал после чтения всего остального (см. последний подраздел данного вводного раздела);

3. Порекомендовал учебник А.И. Мальцева, чьи усилия в области построения теории строк разобраны в прошлом подразделе;

4. Предложил подумать об изменении названия данной статьи с «Теории строк» из-за нюансов в администрировании знания в математическом сообществе;

5. Дал пример аннотации, которую я практически полностью скопировал в качестве аннотации данной статьи;

6. Акцентировал моё внимание на том, что длинная арифметика не является арифметикой в контексте моего рассмотрения и это важный факт в прикладном отношении.

Итак, мы видим, что строки имеют такие свойства, без которых нет никакой возможности «вывести» в арифметике ни правильный размер алфавита, ни даже получить число с ленты машины Тьюринга. Значит, есть какие-то работы, которые изучают эти дополнительные возможности у строк по сравнению с числами? Проверим это.

Начнём с немного неожиданного запроса в Гугле:

string vs Goedel numbers

И ничего по существу этого запроса не обнаружим.

Есть метрика Гёделя, чёрные дыры, теоремы Гёделя и т.п. не относящиеся к интересующей нас теме. То есть, вопрос, который задал мне один из математиков и который я не воспринял тогда всерьёз: «Почему вы не используете Гёделевых номеров?» – это серьёзный вопрос. Гёделевы номера по факту действительно считаются полноценной альтернативой строкам и никакого несоответствия («vs») между ними не видят – судя по результатам запроса. А данная статья выявляет очень существенное – для теории алгоритмов – несоответствие между нумерацией строк (не обязательно гёделевой) и самими строками.

Но, как будет доказано ниже (в VI подразделе 2-го раздела статьи), у чисел арифметики (и нумераций для строк) нет важнейшего для теории алгоритмов качества – возможности их «частичного чтения» в качестве «входных данных». Последствия этого описаны в предыдущем подразделе, а невозможность частичного чтения мы назвали «примат целостности».

Возьмём, например, Жильцова Л.П., Смирнова Т.Г. Основы теории автоматов и формальных языков в примерах и задачах: учебно-методическое пособие

Можно брать любой другой учебник на эту тему и делать те же выводы, например:

Пентус А. Е., Пентус М. Р. Теория формальных языков: Учебное пособие

В работах на эту тему нет теорий первого порядка. Приводятся просто упоминания о числах и строках на иллюстративном уровне и после этого начинается рассмотрение на уровне более высоком – с грамматиками и функциями, сопоставляющими одни слова другим – и тоже без углубления в

детали работы функций. Но проблема в том, что подобные сопоставления слов не проработаны в своей основе даже до уровня программирования. А как без этого можно оценить время на ту или иную операцию? Никак. Поясню.

У нас есть «грамматика» – а как это работает на уровне алгоритмов? Если программировать переход на основе некоторой грамматики от одного слова к другому, то это будет программа не в 10 строчек, а куда больше. В теории же необходимо прописать ещё и множество аксиом, чтобы отразить всю ту богатую логику, которая уже есть в компьютере, и которой мы просто пользуемся, когда программируем.

Возможно, я просто не знаю более проработанных в теоретическом или хотя бы программистском отношении книг в математике по строкам (словам) и алгоритмам, но они есть. Справедливости ради – есть, например, «Прикладная логика» Н.Н. Непейводы со сведениями из современного программирования по ходу изложения. Но это про логику в первую очередь. А строки (слова) на этом фоне как объекты теории лишь слегка упоминаются. А вот там, где современное программирование могло бы подтолкнуть к большей детализации и глубине в понимании именно строк (слов) и работающих с ними алгоритмов – в известных мне книгах – нет.

Немного о богатстве логики операций со строками на элементарном примере:

Если мы берём подстроку длиной один символ со второй позиции строки «abc». То у нас получится строка «b». Но – каким образом? Вот исходная строка – это конкатенация (например) трёх символов (не рассматриваем, откуда эти символы, но пусть это уже известно):

«a» · «b» · «c»

Первый символ длиной 1, на втором месте при конкатенации будет «b» из свойств конкатенации и сложения длин (что тоже надо доказать), поэтому при взятии подстроки со второй позиции именно символ «b» попадёт в неё первым. И он будет единственным, если длина подстроки 1, что тоже должно быть обосновано. Да, все эти рассуждения можно провести один раз и после пользоваться ими, но они – необходимы для теории первого порядка. Для них нужны соответствующие свойства, записанные в аксиомах.

А как иначе оценить время работы алгоритма? Если мы начинаем разбивать работу со строками на элементарные операции – то это и есть приведение теории к необходимому уровню теории первого порядка. Остановиться на уровне программирования не удастся – в основе обычных операций программы, записанных в виде некоторого программного кода, лежит логика работы этих операций. И эту логику необходимо формализовать в аксиомах.

Подразумевается, видимо, что в основе формальных языков и автоматов лежит арифметика с её «представимостью» алгоритмов через рекурсивные функции. И этот инструмент позволит нам (якобы) корректно оценивать время – с полиномиальной погрешностью относительно времени работы реального алгоритма. Но это – не так, как уже рассматривалось выше.

Алгоритм проверки принадлежности слова к языку класса NP, например, должен отвергать сертификаты слишком большого размера. Вопрос времени работы – существенный для теории алгоритмов. И поэтому время на отклонение чрезмерно большого сертификата не может включать в себя чтение всего сертификата, размер которого может быть как угодно велик.

Чтобы не рассматривать случай получения на «вход» слишком длинного сертификата, используются «хитрые» формулировки. Но мы будем смотреть на вопрос «в лоб». Алгоритм должен быть способен сам отвергать чрезмерно длинные сертификаты, не читая их целиком. А поэтому и корректная теория (отвечающая целям теории алгоритмов) должна соответствовать этой способности алгоритма.

Нет никаких проблем в том, чтобы на практике запрограммировать такую систему, которая отвергает слишком длинный «вход» после чтения первого же символа, выходящего за предельно допустимый размер входа. И теория, отвечающая целям теории алгоритмов, должна быть способна корректно представлять такой алгоритм – получая свой результат (посредством логического вывода) о работе алгоритма на базе входного аргумента, который имеет вид конкатенации типа:

$a \cdot Unknown$

Где a – обозначает известную и использованную алгоритмом для работы часть строки, а $Unknown$ – неизвестную и не использованную алгоритмом часть строки, которая подана на «вход».

Но несмотря на то, что в информационном поле сейчас отсутствует понимание несоответствия между свойствами строк и их нумерациями числами арифметики, всё же поищем – есть ли какие-нибудь теории строк или работы на эти темы. Воспользуемся специальным инструментом Гугла для математики: String theory.

Оказывается, что теория строк разрабатывалась ещё Тарским в 1935 году. Вот с чего начиналась аксиоматизация:

if a given string s is partitioned into (x, y), that is, $x \cdot y = s$, and into (u, v), that is,
$u \cdot v = s$, then there is a common refinement partition (t, w, z), that is, either $t = u$,
$t \cdot w = x$, $w \cdot z = v$, $z = y$, or $t = x$, $t \cdot w = u$, $w \cdot z = y$, $z = v$.

Идея такая – пусть строка s может быть представлена как конкатенация и в виде $x \cdot y$, и в виде $u \cdot v$. Тогда каждое такое представление создаёт внутри строки s свою «границу» между первым и вторым членами конкатенации. Раз представления – два, то это 2 границы. 2 границы делят переменную s внутри на три части. Это ещё три новых строки (t, w, z). И оба исходных представления могут быть сведены к этим новым трём строкам.

Это мы можем узнать из работы: Albert Visser. Growing Commas. A Study of Sequentiality and Concatenation

Эта работа удобна в том отношении, что там сопоставляются разные варианты аксиоматизации для «теории конкатенации». И ещё данная работа примечательна в том отношении, что в ней рассматриваются способы «упаковки» нескольких строк внутри одной строки. С возможностью их извлечения.

Если посмотреть раздел 5. «TC Does Not Have Pairing», то там использован аналог канторовой пары, насколько я понимаю. Но проблема канторовой пары и её аналогов в том, что канторова пара не может создать логику сверх логики объекта, в котором эта пара заключена. И если исходный объект нельзя читать частично, то и никого из пары не удастся извлечь из «упаковки» до тех пор, пока не прочитан весь объект, в который «упакована» пара.

И в аксиомах, которые рассмотрены в рассматриваемой работе (и это верно для всех работ

на тему теории конкатенации, насколько я понял) есть аксиомы о конкатенации, но нет аксиом о функциях извлечения подстрок.

Не формализована логика функций разрушения, если угодно. Да, можно построить определение для подстрок на базе «существует и единственно» (я сам много раз так делаю в теории ниже), но если нет аксиом для функций «разрушения», то они могут быть произвольные, в том числе и такие, которые требуют использования всего аргумента, а не его части. А раз такие расширения допустимы, то в рамках исходной теории (не расширенной) тоже нельзя доказать что-то, не используя всего аргумента – иначе это противоречило бы возможности расширения до теории, которая применяет только целостные аргументы.

Если же задавать логику функций «разрушения» в аксиомах – то это довольно богатая логика, то есть – это надо делать особо, и я это делаю с самых первых аксиом теории. И даже аксиому о функции создания (конкатенации) строю на базе ещё и функций «разрушения». Взятие подстроки – это же «разрушение». И оно может работать с частью строки, не используя (не «читая») весь аргумент, если эта функция «разрушения» так прописана на уровне аксиом.

Говоря в шутку, жизнь так устроена (у меня это доказано в подразделе VI раздела 2), что помимо «добрых» аксиом создания нужны «злые» аксиомы разрушения, разбирающие объекты до «корней», превращающие их в «пыль» исходных элементов, когда от объекта не остаётся ничего, кроме его «деталей». Вот тогда – вместе с пониманием создания объектов – мы понимаем объекты достаточно подробно для их полноценного практического использования.

Если снова вспомнить о теории формальных языков, то там на иллюстративном уровне есть и возможность взятия части слова. Но без аксиоматизации это не позволяет доказать возможность взятия части слова без чтения всего слова целиком.

Задание «грамматик» не решает вопрос получения части данной строки посредством соответствующей ей строки по некоторой аксиоме «грамматики». Потому что свойство «быть частью» существенным образом связана со свойствами равенства, и должна быть аксиоматизировано посредством равенств или определено на базе некоторых аксиом с равенством. А простое получение новых строк в качестве значения некоторых функций – не создаёт свойств, необходимых для формализации равенства и одного из его возможных проявлений (при выполнении других условий) – свойства «быть частью».

Например, свойство строки c «быть частью строки a» можно записать так:

$\exists b \exists d(a = b \cdot c \cdot d)$

Но в отрыве от необходимых для теории строк аксиом, логика этого равенства ничем не отличается от аналогичного для чисел арифметики равенства:

$\exists b \exists d(a = b + c + d)$

И ничто не мешает считать это соотношением – в том числе – вариантом слияния упорядоченных массивов в новый упорядоченный массив. Тогда вместо свойства «быть частью строки a» получается свойство «состоять из тех элементов (некоторых из них), из которых состоит a». Это менее богатая логика, чем «быть частью строки»

Конечно, на пути расширения теории конкатенации аксиомами для функций «разрушения»

можно достичь эквивалентности с теми аксиомами, которые используют номера мест символов в строке – как это сделано в теории ниже. Но при этом ведь всё равно придётся вводить сопоставление позиций в строке и чисел – в итоге всё равно не удастся избежать чисел, как в аксиомах о строках данной работы. А сопоставление мест в строках с их номерами всё равно необходимо для практических нужд – так не проще ли с этого начать?

Дело в том, что строки и алгоритмы используют миллионы программистов во всём мире, опираясь именно на соответствие мест в строках их номерам. И нужна практичная теория, которая была бы удобна в прикладном отношении. Это не означает, что другие подходы невозможны, но и тот, который избрал я – необходим, на мой взгляд. Понятно, что от момента массового использования чисел на практике до формулировки теории Пеано прошла не одна тысяча лет, и это обычное явление в истории математики. И со строками вполне может быть так же, и мир от этого не погибнет. Это хорошо, что в математике можно не спешить.

Но нет необходимости откладывать создание практически понятной теории строк на такой срок – польза от сближения теории и практики очевидна. Не повредила же аксиоматика Евклида, хотя и без неё можно было спокойно прожить ещё несколько тысячелетий.

В упомянутой работе Альберта Виззера – как и во многих других работах о строках и языках – в рассуждениях присутствует много теории множеств. Точнее – языка теории множеств, потому что в качестве теории первого порядка теорию множеств используют неизмеримо реже, чем её язык.

С точки зрения практиков (программистов) – это не облегчает восприятие. Если же приводить аргументы из математики, то теория множеств не используется в аксиоматике арифметики и арифметика является более совершенной в логическом отношении. Есть даже формула логики второго порядка, которая является полной аксиоматикой для арифметики – пусть эта аксиоматика и не даёт возможности создать полную теорию первого порядка, которая была бы эффективно аксиоматизируемой (в силу первой теоремы Гёделя о неполноте). А вот в отношении теории множеств такой ясности – хотя бы на уровне логики второго порядка – нет.

В своё время (по книге М. Клайна «Математика. Утрата определённости») Фреге пытался построить арифметику на базе теории множеств, но на завершающем этапе написания книги Рассел указал ему на противоречивость использованного в доказательстве «множества всех множеств». Поэтому надёжней – на мой взгляд – и в теории строк обходиться арифметикой и собственно аксиомами строк. Да, это отчасти вопрос вкусов, но мне по вкусу подход формалистов и Гильберта, который доказал свою силу в период великих логических открытий первой половины 20 века.

Но если добавить к рассматриваемым у Альберта Виззера теориям строк арифметику так, как это сделано в следующем подразделе, то получится... арифметика. То есть, на уровне аксиоматики без функций «разрушения» не удаётся создать даже такую теорию строк (или хотя бы квази-строк), которая содержала бы в себе логику, отличную от арифметики.

Те аксиомы, которые я использую – являются очевидными для программистов свойствами строк. И даже теоремы, которые доказываются в теории – тоже в большинстве своём самоочевидны с практической точки зрения. Зачем тогда я строю логические выводы? Чтобы убедиться, что в аксиоматике не упущено из вида ничего практически необходимого в свойствах строк.

И, кроме того, в последних подразделах Раздела 2 в данной статье доказываются теоремы о возможности локальных изменений и извлечений данных в строках – что означает преодоление примата целостности, а это уже тот факт, в котором программисты обычно не отдают себе отчёта.

А имея теорию первого порядка – мы имеем ту базу, на которую можно уверенно опираться в теоретических обсуждениях и тот математический инструмент, который можно анализировать даже при помощи самого этого инструмента для получения принципиальных результатов в отношении вопросов вроде $NP \neq P$.

Кроме того, при помощи изложенной ниже теории в этой статье доказана недостаточная выразительность арифметики для целей теории алгоритмов при «представлении» алгоритмов средствами арифметики. Вот такого доказательства я определённо нигде не видел. А оно имеет принципиальное значение – потому что доказана необходимость создания новой теории первого порядка для работы с вопросами теории алгоритмов – вместо арифметики. Подходящая теория для представимости алгоритмов – теория строк, на мой взгляд, но это уже тема других статей.

Пока же в учебниках для теории алгоритмов используют только представимость алгоритмов только в одной теории первого порядка с равенством – в арифметике. Притом речь о представимости не алгоритмов машины Тьюринга, а представимость алгоритмов из модели рекурсивных функций. А это не достаточная для задач теории алгоритмов представимость, как уже ясно и как будет дополнительно доказано в VI подразделе раздела 2.

Должен ли я был проводить данное доказательство с использованием другой теории (не излагаемой ниже теории строк)? Не обязательно – и эта теория годится. А в силу того, что результат принципиальный – его надо обнародовать. В том виде, как я это излагаю – вроде, не трудно. А другого варианта пока просто нет.

VI. Статья «String Theory»

Самую принципиальную статью по разбираемой теме я не видел на самом видном месте упомянутого инструмента Гугла String theory, до подсказки С.В. Знаменского.

Но разбирать её в последнюю очередь оказалось удачной ошибкой, на мой взгляд.

Статья String Theory. John Corcoran; William Frank; Michael Maloney. The Journal of Symbolic Logic, Vol. 39, No. 4 (Dec., 1974), 625-637.

Условимся о следующих сокращениях:

АТС – арифметические теории строк. Теории, рассмотренные в разбираемой сейчас статье «String theory». Они «синонимичны» арифметике, как там доказано, отсюда такое название.

ТКС – теория «компьютерных» строк. Теория, которую я строю ниже в данной статье.

Доказательство «синонимичности» (эквивалентности) теорий строк Тарского, Гермеса и арифметики строится в «String theory» следующим образом:

1. Доказывается эквивалентность теорий Тарского и Гермеса для алфавитов с одинаковым количеством символов. Это делается при помощи расширения теорий определениями, которые не добавляют новой логики, а затем аксиомы Тарского доказываются как теоремы в расширенной определениями теории Гермеса и наоборот. Поэтому в силу их эквивалентности буду называть обе эти теории одинаково – «теория АТС»

2. Затем доказывается эквивалентность теории АТС с алфавитом из одного символа (отличного от пустой строки) и АТС с алфавитом из многих ($n > 1$) символов. Вот тут происходит переход через неполиномиальность на самом деле. И эквивалентности в смысле соизмеримости размеров строк в пределах полинома – нет. Хотя тут есть огромный нюанс, который разберем ниже.

3. Затем доказывается эквивалентность теории АТС с алфавитом из одного символа и арифметики. В результате оказывается доказанной «синонимичность» между всеми АТС с любыми алфавитами и арифметикой.

Я излагаю очень упрощенно, конечно, вместо АТС используются некоторые их расширения определениями, которые не вносят никакой существенной логики в исходные АТС. Есть специфика в представлении чисел строками – почти что n-ичная запись числа, но в качестве цифр выступают символы, которых n в алфавите, там есть нюанс с «цифрой» ноль, однако не принципиальный для данного рассмотрения. И т.п. частные нюансы.

Очень хорошая статья – я чуть выше тут высказывал мнение, что без аксиоматизации функций «разрушения» теория строк не может использовать строки частично. То есть не может вот чего: «читать» только начало полученной на «вход» строки независимо от того «хвоста», который за этим «началом» следует. Но статья «String theory» сводит разбираемые в ней теории строк к арифметике, а вот про арифметику известно (доказано в VI подразделе раздела 2 здесь), что она не способна на «частичное чтение» числа и всегда «использует» числа целиком.

На основании сводимости теорий строк без аксиом «разрушения» к арифметике теперь то же самое (примат целостности) можно утверждать и про теории строк без аксиом «разрушения». И это уже не мнение, а утверждение, доказанное на основании теоремы о «синонимичности» из статьи

«String theory» и доказанного «целостного использования» чисел арифметикой из VI подраздела раздела 2 здесь.

Что касается аксиоматизаций Тарского и Гермеса, то там есть обычные аксиомы об алфавите и пустой строке, есть аксиома о конкатенации: Вариант аксиом Тарского о конкатенации изложен при обсуждении статьи Альберта Виззера, а у Гермеса – обычное добавление символа в начало строки.

Самое интересное – это аксиома индукции для строк в этих теориях. Это формула второго порядка, потому что есть квантор по предикату. У Тарского:

CA7: $\forall P([\mathrm{P}(0) \wedge \forall x(\mathrm{P}(x) \Rightarrow \forall y(\mathrm{A}(y) \Rightarrow \mathrm{P}(y + x)))] \Rightarrow \forall z\, \mathrm{P}(z))$

А вот арифметика, которая записана вся (полная теория, никакой «неполноты», но, увы – непрактичного 2-го порядка) в одном предложении 2-го порядка (из «Вычислимость и логика» Булоса и Джеффри):

$\forall P([\mathrm{P}(0) \wedge \forall x(\mathrm{P}(x) \Rightarrow \mathrm{P}(x'))] \Rightarrow \forall z\, \mathrm{P}(z))$

А вот аксиома индукции 2-го порядка из теории Гермеса:

SA4: $\forall P([\mathrm{P}(0) \wedge \forall x(\mathrm{P}(x) \Rightarrow [\wedge i]\, \mathrm{P}(s_i \cdot x))] \Rightarrow \forall z\, \mathrm{P}(z))$

И все эти аксиоматизации индукции для строк вместе с небольшим добавлением аксиом о простых свойствах алфавита, пустой строки и конкатенации оказываются эквивалентными арифметике. И эквивалентными в таком контексте оказываются и представленные способы записи аксиомы индукции 2-го порядка.

Например, если брать «обычную» индукцию 2-го порядка и сравнивать с 10-чной записью числа, то из числа 2 получаем следствие для 2, 3, 4 . . . 12 . . . 22 . . . 32. И вот мы получили переход к случаю добавления конкретной цифры (символа) 2 к разным строкам. И так же с остальными цифрами. И можем воспользоваться индукцией АТС.

И другой подход – когда есть индукция АТС. У нас одна последовательность сменилась на 10 «веток», такого типа: 1 – 11 – 21 – 31 - . . . 101 и т.д. По обычной индукции доказываем $\forall z\, \mathrm{P}(1 + 10 \times z)$. И ещё для всех цифр помимо 1. И можем воспользоваться обычной индукцией для каждой ветки, а в результате получаем $\forall z\, \mathrm{P}(z)$.

Я не рассматриваю, кстати, случай АТС с алфавитом из 1 символа (отличного от пустой строки) – в такой АТС индукция почти не отличается от «обычной» арифметической. И, кстати, структура доказательства в «String theory» и структура доказательства здесь, в VI подразделе 2 раздела – совпадают. Я тоже сопоставляю теории строк односимвольного алфавита с многосимвольными вариантами, а затем переношу результат с односимвольного варианта на арифметику. Но только результаты такого рассмотрения в ТКС и АТС получаются противоположными – из-за дополнительной логики в ТКС.

Почему именно логика второго порядка? Не знаю резонов Тарского и Гермеса, могу только предполагать, что при подходе через теорию 2-го порядка происходит «окончательное» решение вопроса с арифметикой – ведь в виде теории 1-го порядка можно построить только часть арифметики, всегда не полную, а теория второго порядка – это уже исчерпывающее решение вопроса в теоретическом отношении. А в «String theory» разбирается именно теоретический вопрос.

В статье «String theory» обсуждаются те теории, которые создавались во времена до компью-

терной революции. И в то время великих логических открытий стояли вопросы не в прикладном (об эффективности вычислений), а в принципиальном (о самой возможности вычислений) смысле. Поэтому свойство «синонимичности» теорий (вместе с категоричностью моделей для теории) представлялось единственно важным, а требование о полиномиальном ограничении на разницу в размерах сопоставляемых строк для «синонимичных» теорий не предъявлялось.

В АТС логика арифметики записывается языком теории строк. И нет специфических именно для строк функций «разрушения» - потому что во времена создания АТС свойства строк были не вполне осознаны, а вот арифметика была освоена очень хорошо. Поэтому в результате такой аксиоматизации для строк получается арифметика. Как в той истории про завод швейных машинок в Туле, где из деталей швейных машинок получаются только автоматы Калашникова.

В ТКС же арифметика не встраивается в теорию строк, а берется в готовом виде. И используется в аксиомах для строк не для записи арифметической логики, но в добавление к ней в аксиомах, специфичных именно для строк. За счёт этого появляется дополнительная логика, сверх логики арифметики.

Мы и сейчас находимся под «гипнозом» арифметики, интуитивно распространяя свойства чисел на объекты иных математических (и практических) «вселенных». Например, что такое значение переменной? В арифметике – это нечто целостное и однозначное, нечто отличающееся от функции, которая возвращает множество значений при разных аргументах. Но на самом деле свойства математического объекта определяются не нашими интуитивными представлениями о нём, а аксиомами, которые задают логику его поведения и применения во «вселенной» его теории.

Так в чём же отличие логики ТКС от логики АТС и от нашего интуитивного представления о значениях переменных? На примере.

В теории строк мы можем определить функцию, возвращающую символ с i-го места строки. Для АТС результат этой функции будет в семантическом отношении фиктивным, кстати, но об этом ниже. Вопрос: каким образом получается этот результат в теории ТКС и чем логика этого получения отличается от способа АТС? Обозначим искомую функцию как $a.\mathrm{f}(i)$, воспользовавшись способом записи операций с «объектами» из объектно-ориентированного программирования (ООП).

Для ТКС

$a.\mathrm{f}(i) = a.\mathrm{str}(i,1) = a.\mathrm{beg}(i).\mathrm{end}(i)$, где a – строка, а «свойства» (термин ООП) этой строки – вернуть начальную часть до i-го символа включительно ($\mathrm{beg}(i)$), а затем полученный «объект-начало» до i-го символа применяется как функция для получения конечной части – теперь от i-го символа включительно ($\mathrm{end}(i)$). А там только один символ остался начиная с i-го места – его и возвращает наша композиция в итоге.

Как видим, строка a выступает в данном случае как функция. Которая может возвращать множество значений в зависимости от задаваемой для расчёта позиции символа. При этом в «промежуточных» вычислениях эта «функция» использует аксиоматизированные функции «разрушения». Одна из которых возвращает только начало строки, а другая – последний символ из этого начала. И логика аксиоматизированных функций «разрушения» не требует обращения к «хвосту» переменной a после i-го символа для проведения своих «вычислений» в данном случае.

В разобранном примере для ТКС строка a выступала не как значение, а как функция, использующая в своих промежуточных «вычислениях» лишь часть того значения, которое переменная a вернула бы «по умолчанию» в операции конкатенации, например. То есть – что такое переменная или константа – зависит от аксиом и способа использования данного «объекта» на самом деле.

Для АТС

А вот для АТС у нас есть только аксиомы для конкатенации, и функцию $a.\mathrm{f}(i)$ нам придётся определять как обратную (в некотором смысле) к функции конкатенации. А конкатенация требует целостных значений. Поэтому и в результате определения функции $a.\mathrm{f}(i)$ эта функция – оказывается функцией «вторичной», и будет работать только с целостными значениями – просто такая логика, на основе которой она определяется. Ведь определения только добавляют обозначения, но не создают своей логики – в отличие от содержательных аксиом.

Если записать (условно и для наглядности) определение $a.\mathrm{f}(i)$ для АТС в терминах ТКС, то «существует и единственно» $a.\mathrm{f}(i)$, такое что:

$a.\mathrm{beg}(i-1) \cdot a.\mathrm{f}(i) \cdot a.\mathrm{end}(i+1) = a.$

Разумеется, это же предложение верно и для ТКС (в АТС оно будет выглядеть совершенно иначе в терминах АТС, разумеется). Но для ТКС истинным является и предложение перед последним. И одно к другому не сводится – у них разная логика. Вот поэтому ТКС и АТС не «синонимичны» в этом отношении (как и во многих других).

То, насколько сильно расширяют логику теории строк аксиомы «разрушения» ТКС в сравнении с АТС удобно разобрать на примере бесконечных строк в ТКС:

Наличие бесконечных строк в ТКС удобно с точки зрения очевидной ограниченности по времени операции извлечения начала подстроки. Раз такое извлечение можно делать из бесконечной строки в рамках ТКС (и если это можно делать ещё и в соответствующей интерпретации – а такие интерпретации есть в реальном мире), то рост размера «хвоста» заведомо не влияет неограниченно на рост времени извлечения начала строки данного размера – ведь есть ограниченный «потолок» на время извлечения начала строки при увеличении остальной части строки («хвоста»). Потому что даже для бесконечного «хвоста» извлечение начала данной строки – занимает ограниченное время в силу ограниченности количества символов в этом начале.

К извлечению «хвоста» бесконечной строки предъявлять требования об ограниченности времени извлечения его всего – нет смысла. Бесконечность в данном случае понимается как потенциальная возможность получить символ данной строки с любого места строки – и такая возможность есть. А извлечь бесконечный «хвост» строки «целиком» на практике – невозможно. Поэтому и нужды в построении теории с формализацией «извлечения целиком» для бесконечной строки не имеет смысла – для неё нет интерпретаций в реальном мире.

То есть, не нужно воспринимать строку как нечто целостное в духе числа из арифметики. И из-за перечисленных резонов понятно, почему вопрос о возможности игнорировать «хвосты» строк, когда можно обходится их началом – так важен. Конечное время необходимо только для извлечения конечных строк – каковыми и являются начала строк, например.

Теперь вернемся к 2-му пункту описания про доказательство из «String theory». Там был непо-

линомиальный переход между «синонимичными» теориями. Мы это уже рассматривали в связи со «стандартной интерпретацией» в предыдущих подразделах. Но отсутствие в арифметике возможности «частичного чтения» чисел – более радикальное проявление недостаточной выразительности. Не количественное, а качественное.

Да, есть экспоненциальное расхождение в размерах – даже при сравнении работы с аргументами, которые используются целиком. Но исправление только этого недостатка в теории строк не сделало бы её адекватной для теории алгоритмов – потому что вопрос про частичное чтение всё равно остался бы не решённым.

А можно ли доказать недостаточную выразительность арифметики на основе того, что используется в «String theory»? Нет, ведь там нечего сравнивать с арифметикой – по факту у нас там просто разные способы записи одной и той же арифметики – включая способ записи на языке строк.

Да, в АТС тоже можно определить длину строки и то, какой символ стоит в строке на каком месте, но это будет фикцией, просто функцией для расчёта неких характеристик числа при «представлениях» разной «ичности». Вот при 10-чном представлении числа 32 его «длина» равна двум, а при двоичном (100000) – шести, а для «счётных палочек» - 33 (если 0 представлен одной счётной палочкой). И какой «символ» стоит на каком месте – тоже вопрос к «ичности» и это – произвольно в арифметике. Все эти теории АТС взаимозаменяемы и расширяются определениями друг до друга без какого-либо изменения логики – только обозначения добавляются для той логики, которая есть и без новых обозначений, и с ними.

Поэтому, строго говоря, в части доказательства из «String theory», рассмотренного выше в пункте 2 – нет неполиномиального расхождения в размерах строк. Потому что там нет ни строк, ни их размеров. Это был как раз тот огромный нюанс, который упомянут в пункте 2 и который сейчас рассматривается.

В АТС невозможно определить такое свойство строки, как её длина. Потому что длина должна соответствовать некоторой логике строк, а не быть произвольно назначенной на роль «длины» функций арифметики. А соответствовать логике строк никакая функция в АТС не может, потому что соответствующая логика в АТС не аксиоматизирована.

В ТКС длина строки определяется на основе аксиомы о конце строки. Аксиома о конце строки записана с использованием функций «разрушения». А у функций «разрушения» есть свои содержательные аксиомы – совсем не вторичные (как в случае определения) к конкатенации. Поэтому в ТКС есть логика для конца строки и её длины – в отличие от АТС.

После чтения статьи «String theory» и всего остального, отчасти упомянутого выше, у меня сложилось мнение о причинах торможения в создании адекватной для теории алгоритмов теории строк до настоящего момента. Оно следующее:

Направление для исследования вопросов теории строк было задано ещё до компьютерной революции в период великих логических открытий, когда теоретический подход, вопросы логики и арифметики были приоритетными. Научное же сообщество очень преемственное, потому что все мы «стоим на плечах гигантов». И идти против образовавшегося «мейнстрима» математическое

сообщество не имело теоретических оснований – не взирая на колоссальное практическое развитие в использовании алгоритмов и строк.

К тому же математическое сообщество, творившее в период великих логических открытий, было уничтожено во время второй мировой войны – не столько люди (хотя Гильберт умер в 1943), сколько именно сообщество, лишившись, например, своего немецкого ядра. И не осталось того, что могло бы относительно легко изменить мейнстрим, потому что изначально само его создавало.

В результате «теоретический» уровень продолжил рассмотрение строк в духе арифметики, а на уровне массовом – «иллюстративные» теории автоматов, формальных языков и т.п., которые не являются теориями в смысле стандартов теорий первого порядка. Потому что для теоретиков – свой мейнстрим.

А между «иллюстративной» и «высокой» математикой происходит бурное развитие ИТ, которое глубже и гораздо детальней первого и крайне далеко от второго. Но – предельно практическое и в этом смысле не очень стабильное без теоретической базы.

Из сноски 5 «String theory»:

«Hermes and Tarski are both inclined to regard characters as somehow reducible to their instances which are in turn reducible to objects of physics. Thus, for Hermes and Tarski, the axioms are to be verified by "scientific experimentation". The present authors agree that knowledge of the truth of the axioms is to be derived from experience but they doubt that "scientific experimentation"is relevant»

Примерный перевод:

«Гермес и Тарский оба склонны рассматривать символы как так или иначе сводимые к их экземплярам, которые, в свою очередь, сводимы к объектам физики. Таким образом, для Гермеса и Тарского аксиомы должны быть проверены "научным экспериментом". Нынешние авторы согласны с тем, что знание истинности аксиом должно быть получено из опыта, но они сомневаются в том, что "научное экспериментирование"уместно»

В любом случае, все авторы, упомянутые в предыдущей цитате, согласны с важностью опыта как основы для теории. И я полностью разделяю это мнение.

В шутку говоря, математика является не демократией, но монархией – истины. Но это такая монархия, где заботиться о благе масс не зазорно, а – хорошо. Хорошо и по той причине, что успешная практика большинства является (не всегда, но как правило) одним из подтверждений истины.

За время, прошедшее от создания Гермесом и Тарским своих теорий, которые в ту пору были чистой математикой – произошло настолько колоссальное распространение практического применения алгоритмов и строк, что все условия для теоретического обобщения накопившейся практики имеются и даже – с избытком, на мой взгляд. И подобное обобщение соответствует направлению теоретического развития, которому следовали Гермес и Тарский, о чём было упомянуто в цитате из сноски 5.

На мой взгляд, нельзя ослаблять теоретические стандарты ради большей доступности теорий публике – потому что популярные объяснения могут опираются на качественную теорию, но под запретом порча теории ради её популяризации. А с другой стороны – необходимо строить теорию

о строках на базе практических достижений – это ИТ, это компьютеры в наше время. Раз уж эти достижения столь велики. И не следует отрываться сразу в чрезмерные абстракции типа логики второго порядка. Можно исследовать теоретические вопросы и с «высоких позиций», но сначала нужно создать теоретическую базу, обобщающую накопившуюся огромную практику, а уже затем будет что рассматривать с более «высоких» позиций.

2 Теория «компьютерных» строк

Общая информация и свод результатов

Математики называют объекты теории алгоритмов термином «слово», программисты используют для того же термин «строка». Далее буду использовать термин программистов, потому что аксиомы построены именно в соответствии с общеизвестной практикой программирования.

Для предметных переменных типа строка буду использовать все переменные, кроме i, j, k, l, m, n – которые оставим для натуральных чисел. И кроме переменной q, которую оставим для числовых значений и пустой строки $\ominus$. Впрочем, натуральные числа тоже являются строками особого вида, и форма представления чисел внутри теории строк будет рассмотрена в разделе 3. Но обратное (что всякая строка является числом) не верно.

Увеличение числа i на 1 буду обозначать так:

i' – такое обозначение увеличения числа на 1 является аксиоматизированной операцией в арифметике (и в её кратком варианте в виде теории Пеано), поэтому сохраним тут это стандартное обозначение.

Кроме того, вместо специального обозначения для арифметического вычитания от числа i числа j (результат которого равен $i - j$, когда $i > j$, и нулю в противном случае) буду использовать обозначение:

$\max(i, j) - j$

Результат приведенной формулы совпадает с результатом арифметического вычитания и более привычен людям, особенно если они программисты, например.

Логические выводы очевидных формул из теории Пеано строить не буду – их можно посмотреть в учебниках логики, и они очевидны для большинства читателей сами по себе. Но в той части, которая касается вывода из собственных аксиом теории строк, буду давать доказательства с подробностью, равной или близкой логическому выводу. Потому что для новой теории надо аккуратно убедиться в достаточности её аксиом для получения необходимых теорем.

Данный раздел будет разбит на подразделы, в каждом из которых будут изложены аксиомы теории строк, независимые от предыдущих подразделов. Начиная с V подраздела новых аксиом не будет, а будут только определения, теоремы и метатеоремы.

Сразу дам сводку всех результатов данного раздела в данном его подразделе. Это удобно для того, чтобы проверять доказательства – сразу видеть все предыдущие утверждения теории, на которые опираются доказательства. Разумеется, не все смотрят доказательства – тогда им вполне хватит этого подраздела вместо того, чтобы читать все остальные подразделы данного раздела.

Однако подраздел VI «Алгоритмы, не сводимые к (частично) рекурсивным функциям и арифметике» прочитать было бы полезно – там обоснована невозможность обойтись арифметикой для решения вопросов теории алгоритмов. И это является причиной для построения теории строк.

Но аксиомы Пеано – которые тоже являются частью теории строк, как и вся арифметика – записаны в следующем разделе. Единственная оговорка – аксиома индукции. Она тоже будет дана в следующем разделе, но будет использоваться и в этом. Однако схема доказательства по индукции

отлично известна и без написания аксиомы, но нужно учитывать, что мы распространим аксиому индукции и на те функции, в которых помимо числовых аргументов имеются ещё и строковые, но при этом не числовые.

Итак, ключевые результаты построения теории строк – кроме аксиом арифметики и аксиомы индукции, расширенной на формулы теории строк, следующие:

Аксиомы, определения и метатеоремы (теоремы о теории)

I. Разбиение строк

Аксиома о делении строки на начальную и конечную части в произвольной позиции i:

(1) $a = \mathrm{beg}(a, i) \cdot \mathrm{end}(a, i')$

Аксиомы пустых строк:

(2.1) $\mathrm{beg}(a, 0) = \ominus$

(2.2) $\mathrm{end}(a, 0) = \ominus$

(2.3) $\mathrm{end}(\mathrm{beg}(a, i), i') = \ominus$ - если мы взяли в качестве новой строки начало старой строки до i-го символа включительно, то после этого символа в новой строке нет символов.

Альтернативное обозначение: $a.\mathrm{beg}(i).\mathrm{end}(i') = \ominus$

(2.4) $\mathrm{beg}(\ominus, q) = \ominus$, начало пустой строки – пустая строка, независимо от того, является второй аргумент числом или нет.

(2.5) $\mathrm{end}(\ominus, q) = \ominus$, то же самое можно сказать про конец пустой строки.

(2.6) $i \neq \ominus$, никакое число (включая ноль) не является пустой строкой.

Аксиомы уменьшения вложенности:

(3.1) $\mathrm{beg}(a, i) = \mathrm{beg}(\mathrm{beg}(a, i), i)$

Альтернативное обозначение: $a.\mathrm{beg}(i) = a.\mathrm{beg}(i).\mathrm{beg}(i)$

(3.2) $\mathrm{beg}(a, i) = \mathrm{beg}(\mathrm{beg}(a, i), i')$

Альтернативное обозначение: $a.\mathrm{beg}(i) = a.\mathrm{beg}(i).\mathrm{beg}(i')$

(3.3) $\mathrm{beg}(a, i) = \mathrm{beg}(\mathrm{beg}(a, i'), i)$

Альтернативное обозначение: $a.\mathrm{beg}(i) = a.\mathrm{beg}(i').\mathrm{beg}(i)$

(3.4) $a = \mathrm{end}(a, 0')$

(3.5) $i \neq 0 \Rightarrow \mathrm{end}(a, i') = \mathrm{end}(\mathrm{end}(a, i), 0'')$

Альтернативное обозначение: $i \neq 0 \Rightarrow a.\mathrm{end}(i') = a.\mathrm{end}(i).\mathrm{end}(0'')$

Определение функции $\mathrm{str}()$ и аксиома перестановки в композиции $\mathrm{beg}()$ с $\mathrm{end}()$:

(4.1) $\mathrm{str}(a, i, j) = \mathrm{beg}(\mathrm{end}(a, i), j)$

Альтернативное обозначение: $a.\,\mathrm{str}(i, j) = a.\mathrm{end}(i).\mathrm{end}(j)$

(4.2) $\mathrm{end}(\mathrm{beg}(a, i), i) = \mathrm{beg}(\mathrm{end}(a, i), 1)$

Альтернативное обозначение: $a.\mathrm{beg}(i).\mathrm{end}(i) = a.\mathrm{end}(i).\mathrm{beg}(1)$

II. Алфавит строк

Аксиомы об «атомах», из которых образованы строки:

(5.1) $\forall a \forall i \exists j\ \mathrm{str}(a, i, 1) = \mathrm{Chr}(j)$

(5.2) Аксиома об «алфавите»:

$(\forall i > l_{ChrLim} : \mathrm{Chr}(i) = \ominus)$

$\wedge(\forall i \leq l_{ChrLim} : \mathrm{Chr}(i) \neq \ominus \wedge \mathrm{Chr}(i) = \mathrm{str}(\mathrm{Chr}(i), 1, 1))$

$\wedge(\forall i \leq l_{ChrLim} \forall j \leq l_{ChrLim} : \mathrm{Chr}(i) = \mathrm{Chr}(j) \Rightarrow i = j)$

(м.5.1) Аксиома алфавита (5.2) независима от предыдущих аксиом, так как массивы произвольных натуральных чисел удовлетворяют предыдущим аксиомам, но не удовлетворяют аксиоме (5.2).

(5.3) Определение функции Ach(u), обратной к функции Chr(i):

$(\mathrm{Ach}(u) = l'_{ChrLim} \wedge u = \ominus)$

$\vee(\mathrm{Ach}(u) = i \wedge u \neq \ominus \wedge \mathrm{Chr}(i) = u)$

$\vee(\mathrm{Ach}(u) = \ominus \wedge \forall i\, \mathrm{Chr}(i) \neq u)$

(5.4) Определение функции $\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j))$, сравнивающей произвольные элементы алфавита $\mathrm{Chr}(i)$ и $\mathrm{Chr}(j)$):

$(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 0 \wedge i > l_{ChrLim} \wedge j > l_{ChrLim})$

$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 0 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i = j)$

$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 1 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i < j))$

$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 1 \wedge i > l_{ChrLim} \wedge j \leq l_{ChrLim})$

$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 2 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i > j)$

$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 2 \wedge i \leq l_{ChrLim} \wedge j > l_{ChrLim})$

III. Концы строк, соединение строк

Аксиома о конце строки, для случая, когда у строки есть конец:

(6.1) $(i \neq 0 \wedge \mathrm{str}(a, i, 1) = \ominus) \Rightarrow \mathrm{str}(a, i', 1) = \ominus$

(м.6.1) Аксиома (6.1) независима от предыдущих аксиом, так как случай, когда пустая строка может чередоваться с другими элементами алфавита внутри «строки», соответствует всем аксиомам перед аксиомой (6.1), а аксиоме (6.1) – не соответствует.

(6.2) Определение для len(u):

$(\mathrm{len}(u) = 0 \wedge \mathrm{str}(u, 1, 1) = \ominus)$

$\vee(\mathrm{len}(u) = i \wedge \mathrm{str}(u, i, 1) \neq \ominus \wedge \mathrm{str}(u, i', 1) = \ominus)$

$\vee(\mathrm{len}(u) = \ominus \wedge \forall i\ \mathrm{str}(u, i, 1) \neq \ominus)$

(7.1) Аксиома о результате конкатенации при конечной 1-й строке:

$\mathrm{len}(a) \neq \ominus$

$\Rightarrow (i \leq \mathrm{len}(a) \Rightarrow \mathrm{str}(a \cdot b, i, 1) = \mathrm{str}(a, i, 1)) \wedge (i > \mathrm{len}(a) \Rightarrow \mathrm{str}(a \cdot b, i, 1) = \mathrm{str}(b, i - \mathrm{len}(a), 1))$

(7.2) Аксиома о результате конкатенации при бесконечной 1-й строке:

$\mathrm{len}(a) = \ominus \Rightarrow a = a \cdot b$

(м.7.1) Аксиомы (7.1) и (7.2) независимы от предыдущих аксиом, так как случай упорядоченных массивов символов соответствует всем аксиомам перед аксиомами (7.1) и (7.2), но не соответствует аксиомам (7.1) и (7.2).

IV. Равенство строк

Аксиома о равенстве строк:

(8.1) $a = b \Leftrightarrow \forall i\ \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1)$

(м.8.1) Аксиома (8.1) независима от предыдущих аксиом, так как есть пример из ООП (объектно-ориентированного программирования), когда функции $\mathrm{beg}()$ и $\mathrm{end}()$ создают «детей» у «независимых» строк. И между «независимыми» строками и «детьми» есть разница помимо последовательности символов. Такие объекты ООП соответствуют всем аксиомам перед аксиомой (8.1), но не соответствует аксиоме (8.1).

(8.2) $(x_1 = y_1 \wedge x_2 = y_2) \Rightarrow x_1 \cdot x_2 = y_1 \cdot y_2$

(8.3) $(x = y \wedge i = j) \Rightarrow \mathrm{beg}(x, i) = \mathrm{beg}(y, j)$

(8.3) $(x = y \wedge i = j) \Rightarrow \mathrm{end}(x, i) = \mathrm{beg}(y, j)$

(8.4) $(i = j) \Rightarrow \mathrm{Chr}(i) = \mathrm{Chr}(j)$

(8.5) $x = x$

(8.6) $x = y \Rightarrow (z = x \Rightarrow z = y)$

V. Сравнение, поиск, вставка

(9.1) Определение для $\mathrm{CompIn}(a, b)$:

$(\mathrm{str}(a, \mathrm{CompIn}(a, b), 1) \neq \mathrm{str}(b, \mathrm{CompIn}(a, b), 1) \wedge \forall i (i < \mathrm{CompIn}(a, b) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1)))$

$\vee (\mathrm{CompIn}(a, b) = 0 \wedge a = b)$

(9.2) Определение для $\mathrm{Comp}(a, b)$:

$\mathrm{Comp}(a, b) = \mathrm{Comp}(\mathrm{str}(a, \mathrm{CompIn}(a, b), 1), \mathrm{str}(b, \mathrm{CompIn}(a, b), 1))$

(9.3) Определение для $\mathrm{find}(in, what, 0)$:

$(what \neq \ominus \wedge \mathrm{len}(what) \neq \ominus \wedge \mathrm{str}(in, \mathrm{find}(in, what, 0), \mathrm{len}(what)) = what$

$\wedge \forall j (j < \mathrm{find}(in, what, 0) \Rightarrow \mathrm{str}(in, j, \mathrm{len}(what)) \neq what)$

$) \vee (\mathrm{find}(in, what, 0) = 0 \wedge (what = \ominus \vee \mathrm{len}(what) = \ominus \vee \forall j \ \mathrm{str}(in, j, \mathrm{len}(what)) \neq what))$

(9.4) Определение для $\mathrm{find}(in, what, i)$:

$\mathrm{find}(in, what, i) = \mathrm{find}(\mathrm{end}(in, i'), what, 0)$

(9.5) Определение для $\mathrm{if}_0(a, b, c)$:

$(\mathrm{if}_0(a, b, c) = b \wedge a = 0) \vee (\mathrm{if}_0(a, b, c) = c \wedge a \neq 0)$

(9.6) Определение для $\mathrm{Ins}(a, i, b)$:

$\mathrm{Ins}(a, i, b) = \mathrm{beg}(a, \max(i, 1) - 1) \cdot b \cdot \mathrm{end}(a, \mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b) + i))$

(9.7) Определение для функции $\mathrm{IsIns}(q_a, i, q_b)$:

$(\mathrm{IsIns}(q_a, i, q_b) = 1 \wedge i \neq 0 \wedge q_a \neq \ominus \wedge q_b \neq \ominus \wedge (q_b = 0 \vee q_a \geq i + q_b - 1))$

$\vee (\mathrm{IsIns}(q_a, i, q_b) = 0 \wedge (i = 0 \vee q_a = \ominus \vee q_b = \ominus \vee (q_b \neq 0 \wedge q_a < i + q_b - 1)))$

(9.8) Определение для функции $\mathrm{IsStr}(q_a, i, j)$:

$(\mathrm{IsStr}(q_a, i, j) = 1 \wedge q_a \neq \ominus \wedge (i = 0 \vee j = 0 \vee q_a \geq i + j - 1))$

$\vee (\mathrm{IsStr}(q_a, i, j) = 0 \wedge (q_a = \ominus \vee (i \neq 0 \wedge j \neq 0 \wedge q_a < i + j - 1)))$

(9.9) Определение для $\mathrm{ss}(i)$ и $\mathrm{ss_k}(i)$:

$\mathrm{len}(\mathrm{ss}(i)) = i \wedge \forall j \leq i : \mathrm{str}(\mathrm{ss}(i), j, 1) = \mathrm{Chr}(0)$. Функция $\mathrm{ss_k}(i)$ определяется так же, но только вместо $\mathrm{Chr}(0)$ используется $\mathrm{Chr}(k)$, где $k \leq l_{ChrLim}$.

VI. Алгоритмы, не сводимые к (частично) рекурсивным функциям и арифметике

(м.9.1) В теории строк с односимвольным алфавитом ($l_{ChrLim} = 0$) невозможно в общем случае представить работу алгоритмов со строками многосимвольного алфавита таким образом, чтобы

размер логического вывода для результата работы такого алгоритма был в полиномиальных (или любых иных) пределах относительно времени работы этого алгоритма.

(м.9.2) В арифметике невозможно в общем случае представить работу алгоритмов со строками многосимвольного алфавита таким образом, чтобы размер логического вывода для результата работы такого алгоритма был в полиномиальных (или любых иных) пределах относительно времени работы этого алгоритма.

Теоремы, леммы и следствия

I. Разбиение строк

(т.3.1) $\mathrm{beg}(a, \min(i, j)) = \mathrm{beg}(\mathrm{beg}(a, i), j)$

Альтернативное обозначение: $a.\mathrm{beg}(\min(i, j)) = a.\mathrm{beg}(i).\mathrm{beg}(j)$

(т.3.2) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{end}(a, i + j - 1) = \mathrm{end}(\mathrm{end}(a, i), j)$

Альтернативное обозначение: $(i \neq 0 \wedge j \neq 0) \Rightarrow a.\mathrm{end}(i + j - 1) = a.\mathrm{end}(i).\mathrm{end}(j)$

(т.3.3) $i < k \Rightarrow \mathrm{end}(\mathrm{beg}(a, i), k) = \ominus$

Альтернативное обозначение: $i < k \Rightarrow a.\mathrm{beg}(i).\mathrm{end}(k) = \ominus$

Определение функции $\mathrm{str}()$ и аксиома перестановки в композиции $\mathrm{beg}()$ с $\mathrm{end}()$:

(т.4.1) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{beg}(\mathrm{end}(a, i), j) = \mathrm{end}(\mathrm{beg}(a, i + j - 1), i)$

Альтернативное обозначение: $(i \neq 0 \wedge j \neq 0) \Rightarrow a.\mathrm{end}(i).\mathrm{beg}(j) = a.\mathrm{beg}(i + j - 1).\mathrm{end}(i)$

(т.4.2) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{str}(a, i, j) = \mathrm{end}(\mathrm{beg}(a, i + j - 1), i)$

Альтернативное обозначение: $(i \neq 0 \wedge j \neq 0) \Rightarrow a.\mathrm{str}(i, j) = a.\mathrm{beg}(i + j - 1).\mathrm{end}(i)$

(т.4.3) $(i \neq 0) \Rightarrow \mathrm{str}(a, i, j') = \mathrm{str}(a, i, j) \cdot \mathrm{str}(a, i + j, 1)$

(т.4.4) $j < k \Rightarrow \mathrm{str}(\mathrm{str}(a, i, j), k, 1) = \ominus$

Альтернативное обозначение: $j < k \Rightarrow a.\mathrm{str}(i, j).\mathrm{str}(k, 1) = \ominus$

(т.4.5) $1 < k \Rightarrow \mathrm{str}(\mathrm{str}(a, i, 1), k, 1) = \ominus$

Альтернативное обозначение: $1 < k \Rightarrow a.\mathrm{str}(i, 1).\mathrm{str}(k, 1) = \ominus$

(т.4.6) $\mathrm{str}(a, 1, j) = \mathrm{beg}(a, j)$

(т.4.7) $j \leq k \Rightarrow \mathrm{str}(\mathrm{str}(a, 1, j), 1, k) = \mathrm{str}(a, 1, j)$

Альтернативное обозначение: $j \leq k \Rightarrow a.\mathrm{str}(1, j).\mathrm{str}(1, k) = a.\mathrm{str}(1, j)$

(т.4.8) $\mathrm{end}(\mathrm{str}(a, 1, j), k) = \mathrm{str}(\mathrm{str}(a, 1, j), k, j)$, притом вместо одного данного $\mathrm{end}()$ может быть композиция из подобных $\mathrm{end}()$ сколь угодно большой глубины вложенности.

Альтернативное обозначение: $a.\mathrm{str}(1, j).\mathrm{end}(1, k) = a.\mathrm{str}(1, j).\mathrm{str}(k, j)$

(т.4.9) $\mathrm{str}(a, 1, j) = \mathrm{str}(\mathrm{str}(a, 1, j), 1, k) \cdot \mathrm{str}(\mathrm{str}(a, 1, j), k', j)$

Альтернативное обозначение: $a.\mathrm{str}(1, j) = a.\mathrm{str}(1, j).\mathrm{str}(1, k) \cdot a.\mathrm{str}(1, j).\mathrm{str}(k', j)$

II. Алфавит строк

(т.5.1) $\mathrm{Ach}(\mathrm{Chr}(i)) = \min(i, l'_{ChrLim})$

III. Концы строк, соединение строк

(т.6.1) $0 < i \leq j \Rightarrow (\mathrm{str}(a, i, 1) = \ominus \Rightarrow \mathrm{str}(a, j, 1) = \ominus)$

(т.6.2) $\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a, \mathrm{len}(a), 1) \neq \ominus)$

(т.6.3) $\mathrm{len}(a) \neq \ominus \Rightarrow (\mathrm{len}(a) < i \Rightarrow \mathrm{str}(a, i, 1) = \ominus)$

(т.6.4) $j \leq i \Rightarrow \mathrm{str}(\mathrm{beg}(a, i), j, 1) = \mathrm{str}(a, j, 1)$

Альтернативное обозначение: $j \le i \Rightarrow a.\mathrm{beg}(i).\mathrm{str}(j,1) = a.\mathrm{str}(j,1)$

(т.6.5) $i \ne 0 \wedge j \ne 0 \Rightarrow \mathrm{str}(\mathrm{end}(a,i),j,1) = \mathrm{str}(a,i+j-1,1)$

Альтернативное обозначение: $i \ne 0 \wedge j \ne 0 \Rightarrow a.\mathrm{end}(i).\mathrm{str}(j,1) = a.\mathrm{str}(i+j-1,1)$

(т.6.6) $\mathrm{len}(a) \ne \ominus \wedge 0 < i \le \mathrm{len}(a) \Rightarrow \mathrm{str}(a,i,1) \ne \ominus$

(т.6.7) $\mathrm{len}(a) \ne \ominus \wedge i \le \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{beg}(a,i)) = i \wedge (\mathrm{len}(\mathrm{end}(a,i)) = \mathrm{len}(a) - i + 1 \vee i = 0)$

(с.6.7) $\mathrm{len}(a) \ne \ominus \wedge i \ne 0 \wedge i + j - 1 \le \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{str}(a,i,j)) = j$

(т.6.8) $\mathrm{len}(a) = \ominus \Rightarrow \mathrm{len}(\mathrm{beg}(a,i)) = i \wedge (\mathrm{len}(\mathrm{end}(a,i)) = \ominus \vee i = 0)$

(т.6.9) $\mathrm{len}(a) \ne \ominus \wedge \mathrm{len}(a) < i \Rightarrow \mathrm{len}(\mathrm{beg}(a,i)) = \mathrm{len}(a) \wedge \mathrm{len}(\mathrm{end}(a,i)) = 0$

(с.6.9) $\mathrm{len}(a) \ne \ominus \Rightarrow \mathrm{len}(\mathrm{beg}(a,i)) = \min(i, \mathrm{len}(a)) \wedge (\mathrm{len}(\mathrm{end}(a,i)) = \max(\mathrm{len}(a), i-1) - i + 1 \vee i = 0)$

(т.7.1) $\mathrm{len}(a_1) \ne \ominus \wedge \ldots \wedge \mathrm{len}(a_n) \ne \ominus \Rightarrow \mathrm{len}(a_1 \cdot \ldots \cdot a_n) = \mathrm{len}(a_1) + \ldots + \mathrm{len}(a_n)$

(т.7.2) $i \le l_{ChrLim} \Rightarrow \mathrm{len}(\mathrm{Chr}(i)) = 1$

IV. Равенство строк

Ассоциативность конкатенации:

(т.8.1) $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

(т.8.2) $\mathrm{len}(a) \ne \ominus \Rightarrow a = \mathrm{str}(a, 1, \mathrm{len}(a))$, в силу чего в отношении конечной строки a, в виде $\mathrm{str}(a, 1, \mathrm{len}(a))$, можно применять теоремы (т.4.4) и от (т.4.6) до (т.4.9) включительно

(с.8.2) $\mathrm{len}(a) \ne \ominus \Rightarrow (\mathrm{len}(a) \le i \Rightarrow a = \mathrm{str}(a, 1, i))$

(т.8.3) $a = a \cdot \ominus = \ominus \cdot a$

(т.8.4) $a = b \Leftrightarrow \forall i (0 < i \Rightarrow \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1))$

(т.8.5) $\mathrm{len}(a) \ne \ominus \Rightarrow [a = b \Leftrightarrow \forall i ((\mathrm{len}(a) = \mathrm{len}(b) \wedge 0 < i \wedge i \le \mathrm{len}(a)) \Rightarrow \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1))]$

(т.8.6) $\mathrm{len}(a) \ne \ominus \Rightarrow a = \mathrm{beg}(a \cdot u, \mathrm{len}(a))$

(т.8.7) $\mathrm{len}(a) \ne \ominus \Rightarrow u = \mathrm{end}(a \cdot u, \mathrm{len}(a)')$

(т.8.8) $a = \ominus \Leftrightarrow \mathrm{len}(a) = 0$

(с.8.8) $\mathrm{len}(a) = \ominus \Leftrightarrow \mathrm{len}(\mathrm{len}(a)) = 0$

V. Сравнение, поиск, вставка

(л.9.1) Лемма о минимальной позиции отличия в строках:

$\mathrm{str}(a,n,1) \ne \mathrm{str}(b,n,1) \Rightarrow$

$\exists_1 m (\mathrm{str}(a,m,1) \ne \mathrm{str}(b,m,1) \wedge (\forall i < m : \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1)))$

Лемма об эквивалентном добавлении дизъюнктивных членов:

(л.9.2) $A \Rightarrow ((A \Rightarrow \neg C) \Rightarrow (B \Leftrightarrow (B \vee C)))$

(с.9.1) $\mathrm{len}(a) \ne \ominus \Rightarrow [0 < \mathrm{CompIn}(a,b) \le \mathrm{len}(a) \Leftrightarrow \forall u\, \mathrm{CompIn}(a \cdot u, b) = \mathrm{CompIn}(a,b)]$, и то же верно, если в этом следствии заменить $\mathrm{CompIn}(a,b)$ и $\mathrm{CompIn}(a \cdot u, b)$ на $\mathrm{CompIn}(b,a)$ и $\mathrm{CompIn}(b, a \cdot u)$ соответственно.

Критерий локального сравнения данных:

(с.9.2) $\mathrm{len}(a) \ne \ominus \Rightarrow [0 < \mathrm{CompIn}(a,b) \le \mathrm{len}(a) \Leftrightarrow \forall u\, \mathrm{Comp}(a \cdot u, b) = \mathrm{Comp}(a,b)]$, и то же верно, если в этом следствии заменить $\mathrm{CompIn}(a,b)$, $\mathrm{Comp}(a \cdot u, b)$ и $\mathrm{Comp}(a,b)$ на $\mathrm{CompIn}(b,a)$, $\mathrm{Comp}(b, a \cdot u)$ и $\mathrm{Comp}(b,a)$ соответственно.

(л.9.3) Лемма о минимальной позиции совпадения в строке:

$what \neq \ominus \wedge \mathrm{len}(what) \neq \ominus \wedge \mathrm{str}(in, n, \mathrm{len}(what)) = what \Rightarrow$
$\exists_1 m(\mathrm{str}(in, m, \mathrm{len}(what)) = what \wedge (\forall i < m : \mathrm{str}(in, i, \mathrm{len}(what)) \neq what))$
(л.9.4) $a \neq a \cdot \mathrm{Chr}(0) \Leftrightarrow \mathrm{len}(a) \neq \ominus$

VII. Критерий локального изменения данных

(т.9.1) $\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus \Rightarrow \mathrm{str}(a \cdot b \cdot u, \mathrm{len}(a)', \mathrm{len}(b)) = b$
(т.9.2) $\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus \Rightarrow \mathrm{end}(a \cdot b \cdot u, \mathrm{len}(a)') = b \cdot u$
(т.9.3) $\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1 \Rightarrow \mathrm{len}(\mathrm{Ins}(a, i, b)) = \mathrm{len}(a)$
(т.9.4) Критерий локальной вставки:
$(\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus) \Rightarrow$
$(\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1 \Leftrightarrow \forall u\, \mathrm{Ins}(a \cdot u, i, b) = \mathrm{Ins}(a, i, b) \cdot u)$

VIII. Критерий локального извлечения данных

(т.9.5) $\mathrm{len}(a) \neq \ominus \Rightarrow (\mathrm{IsStr}(\mathrm{len}(a), i, j) = 1 \Leftrightarrow \forall u\ \mathrm{str}(a \cdot u, i, j) = \mathrm{str}(a, i, j))$

Теперь будем записывать аксиомы и определения теории строк, разбирать их семантику и доказывать теоремы.

I. Разбиение строк

Аксиома о делении строки на начальную и конечную части в произвольной позиции i:

(1) $a = \mathrm{beg}(a, i) \cdot \mathrm{end}(a, i')$

Пояснение. Разумеется, если длина строки 6 символов, то деление с 8 позиции приведет к конкатенации исходной строки и пустой строки, обозначаемой значком $\ominus$:

$«string» = \mathrm{beg}(«string», 8) \cdot \mathrm{end}(«string», 9) = «string» \cdot \ominus$

Если же то же самое делать с 3 позиции, то получится обычное:

$«string» = \mathrm{beg}(«string», 3) \cdot \mathrm{end}(«string», 4) = «str» \cdot «ing»$

Впрочем, всё это будет понятно из дальнейших аксиом.

Кстати, «конкатенация» – это операция соединения строк. В данной аксиоме – соединение «начала» и «конца» строки, разбитой по границе между местом строки i и i'. Конкатенация в данном цикле статей будет обозначаться значком «$\cdot$» – отняв это обозначение у умножения чисел. Для умножения мы будем использовать значок «$\times$».

В данной группе аксиом («I. Разбиение строк») мы рассматриваем только соединение частей одной строки между собой. Эти аксиомы не могут решить вопрос о соединении (конкатенации) разных строк между собой – как будет показано в группе аксиом «IV. Соединение строк».

Нумерация символов в строке начинается с 1, чтобы использовать место номер «0» для «технических нужд» - что будет видно в этой и следующих статьях цикла. И длина строки при такой нумерации совпадает с номером последнего символа в ней.

Аксиомы пустых строк:

(2.1) $\mathrm{beg}(a, 0) = \ominus$ - если пытаемся взять начало до 1-го символа(не включая 1й символ), то получаем пустую строку;

(2.2) $\mathrm{end}(a, 0) = \ominus$ - если пытаемся брать окончание строки с позиции перед первой позицией, то ничего не обнаруживаем, потому что перед 1-м местом нет символа, который можно вернуть со всей цепочкой символов после него;

(2.3) $\mathrm{end}(\mathrm{beg}(a, i), i') = \ominus$ - если мы взяли в качестве новой строки начало старой строки до i-го символа включительно, то после этого символа в новой строке нет символов.

Альтернативное обозначение: $a.\mathrm{beg}(i).\mathrm{end}(i') = \ominus$

«Альтернативное обозначение» – из объектно-ориентированного программирования (ООП), когда у объектов есть свойства, а чтобы узнать значение свойства – надо задать после имени объекта через точку имя свойства со списком аргументов. Вот так:

$object.\mathrm{property}(Arg_1, \ldots, Arg_n)$

В данной аксиоме объектом является строка, а свойством – подстрока в начале данной строки. А затем объектом становится полученная строка (подстрока тоже является строкой) и его свойство – подстрока в конце этой строки. А аргумент задаёт границу, до которого нас интересует подстрока в начале, а затем граница, от которой берём подстроку в конце. И результатом тоже является строка, что снова позволяет обратиться к свойству этой новой строки. И так – с любой глубиной композиции свойств.

Такое обозначение гораздо удобней для композиций глубокой вложенности, чем традиционное. Потому что обычное обозначение приводит к необходимости читать исполнение вложенных функций справа налево, а аргументы – слева направо, при этом разрыв нарастает и всё это – с трудно уловимым вложенным соответствием скобок.

Понимая, что новые обозначения не все быстро осваивают, будем придерживаться в основном традиционного обозначения. Кроме случаев слишком большой вложенности, по крайней мере.

(2.4) $\mathrm{beg}(\ominus, x) = \ominus$, начало пустой строки – пустая строка, независимо от того, является второй аргумент числом или нет.

(2.5) $\mathrm{end}(\ominus, x) = \ominus$, то же самое можно сказать про конец пустой строки.

(2.6) $i \neq \ominus$, никакое число (включая ноль) не является пустой строкой. Тут выясняется, что в теории строк есть объект (хотя бы один), отличающийся от любого числа.

Аксиомы уменьшения вложенности:

(3.1) $\mathrm{beg}(a, i) = \mathrm{beg}(\mathrm{beg}(a, i), i)$

Альтернативное обозначение: $a.\mathrm{beg}(i) = a.\mathrm{beg}(i).\mathrm{beg}(i)$

(3.2) $\mathrm{beg}(a, i) = \mathrm{beg}(\mathrm{beg}(a, i), i')$

Альтернативное обозначение: $a.\mathrm{beg}(i) = a.\mathrm{beg}(i).\mathrm{beg}(i')$

(3.3) $\mathrm{beg}(a, i) = \mathrm{beg}(\mathrm{beg}(a, i'), i)$

Альтернативное обозначение: $a.\mathrm{beg}(i) = a.\mathrm{beg}(i').\mathrm{beg}(i)$

Пояснение для аксиом (3.1)-(3.3). Если отбрасывать все символы в строке 2 раза – после i-го и $i+1$-го символов, то независимо от последовательности этих 2 операций останется только i начальных символов после отбрасывания.

По индукции легко доказать, что

(т.3.1) $\mathrm{beg}(a, \min(i, j)) = \mathrm{beg}(\mathrm{beg}(a, i), j)$

Альтернативное обозначение: $a.\mathrm{beg}(\min(i, j)) = a.\mathrm{beg}(i).\mathrm{beg}(j)$

Нюанс: «легко доказать» - если использовать обычные правила работы с равенством. Вопрос о равенстве в данном разделе будет обсуждаться особо в подразделе IV.

Основные моменты доказательства (т.3.1) на примере. Из предположения истинности для $i < j$ предложения $\mathrm{beg}(a, \min(i, j)) = \mathrm{beg}(\mathrm{beg}(a, i), j)$ получим:

$\mathrm{beg}(\mathrm{beg}(a, i), j') = \mathrm{beg}(\mathrm{beg}(\mathrm{beg}(a, i), j), j') = \mathrm{beg}(\mathrm{beg}(a, i), j) = \mathrm{beg}(a, \min(i, j)) = \mathrm{beg}(a, \min(i, j'))$

(3.4) $a = \mathrm{end}(a, 0')$

Пояснение: если брать всю цепочку символов, начиная с 1-го, то получим всю исходную строку.

(3.5) $i \neq 0 \Rightarrow \mathrm{end}(a, i') = \mathrm{end}(\mathrm{end}(a, i), 0'')$

Альтернативное обозначение: $i \neq 0 \Rightarrow a.\mathrm{end}(i') = a.\mathrm{end}(i).\mathrm{end}(0'')$

Пояснение. Удаление первых i символов в строке – это то же самое, что удалить $i-1$ первых символов, а затем удалить ещё 1-й символ в оставшейся строке. Отсюда легко доказывается по индукции, как и для (т.3.1) и с той же оговоркой про равенство:

(т.3.2) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{end}(a, i + j - 1) = \mathrm{end}(\mathrm{end}(a, i), j)$

Альтернативное обозначение: $(i \neq 0 \wedge j \neq 0) \Rightarrow a.\mathrm{end}(i + j - 1) = a.\mathrm{end}(i).\mathrm{end}(j)$

(т.3.3) $i < k \Rightarrow \mathrm{end}(\mathrm{beg}(a,i),k) = \ominus$

Альтернативное обозначение: $i < k \Rightarrow a.\mathrm{beg}(i).\mathrm{end}(k) = \ominus$

Выводится из (2.3) $\mathrm{end}(\mathrm{beg}(a,i),i') = \ominus$, (2.5) $\mathrm{end}(\ominus,i) = \ominus$ и (3.5)

Доказывается индукцией по k от $k = i'$, что совпадает с (2.3). Допустим

$\mathrm{end}(\mathrm{beg}(a,i),k) = \ominus$ и докажем из этого, что $\mathrm{end}(\mathrm{beg}(a,i),k') = \ominus$.

$\mathrm{end}(\mathrm{beg}(a,i),k') = \mathrm{end}(\mathrm{end}(\mathrm{beg}(a,i),k),0'')$, по (3.5)

$\mathrm{end}(\mathrm{end}(\mathrm{beg}(a,i),k),0'') = \mathrm{end}(\ominus,0'')$, по предположению индукции

$\mathrm{end}(\ominus,0'') = \ominus$, по (2.5). Это завершает доказательство по индукции.

Определение функции $\mathrm{str}()$ и аксиома перестановки в композиции $\mathrm{beg}()$ с $\mathrm{end}()$:

(4.1) $\mathrm{str}(a,i,j) = \mathrm{beg}(\mathrm{end}(a,i),j)$

Альтернативное обозначение: $\mathrm{str}(a,i,j) = a.\mathrm{end}(i).\mathrm{beg}(j)$

Пояснение. Это обычная для программирования функция взятие подстроки в строке. В теории оказалось удобнее разбить логику её работы на 2 этапа: удаление начальной части строки до $i-1$ символа включительно (функция $\mathrm{end}()$) и, затем, удаление в оставшейся части строки всех символов от $j+1$-го включительно и всех символов после него (функция $\mathrm{beg}()$).

Функция $\mathrm{str}()$ чрезвычайно важна для логики строк, так как раскрывает структуру строк до уровня их «атомов» через $\mathrm{str}(a,i,1)$. На любом месте в строке находится либо символ, либо пустая строка. Но сначала нам необходима аксиома, на базе которой выводится способ построения подстроки из её минимальных «атомов». Речь пока не про произвольную строку, а только про результат применения функции $\mathrm{str}()$ к произвольной строке.

После аксиом понижения вложенности и следующей аксиомы перестановки можно любую композицию из вложенных функций $\mathrm{str}()$ с конкретными числами свести либо к одной функции $\mathrm{str}()$, либо к пустой строке $\ominus$.

Если дело сведётся к одной функции $\mathrm{str}()$ вместо их композиции, то на месте 2-го и 3-го аргументов этой единственной функции $\mathrm{str}()$ будут некие арифметические результаты из всех чисел в исходной композиции. И следующая аксиома перестановки – вместе с изложенными выше аксиомами – делает это возможным:

(4.2) $\mathrm{end}(\mathrm{beg}(a,i),i) = \mathrm{beg}(\mathrm{end}(a,i),1)$

Альтернативное обозначение: $a.\mathrm{beg}(i).\mathrm{end}(i) = a.\mathrm{end}(i).\mathrm{beg}(1)$

Пояснение. У нас есть 3 способа получить одинаковый результат:

1. Взять в качестве результата то (символ или пустую строку), что находится на i-м месте в исходной строке (способ, производный для данной теории от 2-х следующих);

2. Удалить все символы после (не включая) i-го символа и затем удалить в оставшейся строке все символы до (не включая) i-го символа;

3. Удалить в исходной строке все символы до (не включая) i-го символа и затем удалить в оставшейся строке все символы после (не включая) 1-го.

(т.4.1) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{beg}(\mathrm{end}(a,i),j) = \mathrm{end}(\mathrm{beg}(a,i+j-1),i)$

Альтернативное обозначение: $(i \neq 0 \wedge j \neq 0) \Rightarrow a.\mathrm{end}(i).\mathrm{beg}(j) = a.\mathrm{beg}(i+j-1).\mathrm{end}(i)$

Доказывать будем в виде

$\forall x\ \mathrm{beg}(\mathrm{end}(x,i'),j') = \mathrm{end}(\mathrm{beg}(x,i'+j),i')$ – квантор общности тут, чтоб подчеркнуть, что доказательство по индукции ведётся для всех строк сразу, а не для некоторой фиксированной строки a. И переменные тут в сравнении с (т.4.1) имеют вид i' вместо i и j' вместо j. Это сделано для устранения из формулы посылки $(i \neq 0 \wedge j \neq 0)$. Потому что любое натуральное число m, не равное нулю, может быть представлено как n' для некоторого n. И, обратно, любое число вида n' – не равно нулю.

Доказывать будем, отталкиваясь от (4.2)

$\mathrm{beg}(\mathrm{end}(a,i),1) = \mathrm{end}(\mathrm{beg}(a,i),i)$

Допустим, у нас доказано «предположение индукции»

$\forall x\ \mathrm{beg}(\mathrm{end}(x,i'),j') = \mathrm{end}(\mathrm{beg}(x,i'+j),i')$, что для $j = 0$ верно из-за (4.2)

Нам надо доказать

$\forall x\ \mathrm{beg}(\mathrm{end}(x,i'),j'') = \mathrm{end}(\mathrm{beg}(x,i'+j'),i')$

То, что у нас квантор общности – подчёркивает, что доказательство идёт сразу для всех строк, потому что в процессе вывода мы будем использовать предположение индукции не только для одной конкретной строки a. Но, фактически, мы будем исходить из «вторичного» предположения индукции:

$\mathrm{beg}(\mathrm{end}(a,i'),j') = \mathrm{end}(\mathrm{beg}(a,i'+j),i')$

А доказывать будем искомое следствие индукции без квантора общности:

$\mathrm{beg}(\mathrm{end}(a,i'),j'') = \mathrm{end}(\mathrm{beg}(a,i'+j'),i')$

Обоснованием такого способа действия является аксиома логики (см. раздел 4):

(а) $\forall x\,\mathrm{A}(x,j) \Rightarrow \mathrm{A}(a,j)$

Что позволяет нам перейти от предположения индукции к вторичному предположения индукции.

Точнее, мы воспользуемся этим предположением два раза, и получим результат вида:

$\forall x\,\mathrm{A}(x,j) \Rightarrow \mathrm{A}(a,j) \wedge \mathrm{A}(b,j)$

Затем мы получим искомое следствие без квантора общности, то есть – докажем импликацию вида:

$\mathrm{A}(a,j) \wedge \mathrm{A}(b,j) \Rightarrow \mathrm{A}(a,j')$

На основании силлогизма по последним двум импликациям будем иметь:

$\forall x\,\mathrm{A}(x,j) \Rightarrow \mathrm{A}(a,j')$

по правилу вывода логики (см. раздел 4):

(α) Если верно $U \Rightarrow \mathrm{B}(a)$, то верно $U \Rightarrow \forall x\,\mathrm{B}(x)$

Получим результат вида:

$\forall x\,\mathrm{A}(x,j) \Rightarrow \forall x\,\mathrm{A}(a,j')$

И, таким образом, завершим доказательство индукции. Начнём же его:

Для этого надо рассмотреть 2 ветки для 2 сторон данного равенства. Каждая ветка будет построена на основании (1)

$a = \mathrm{beg}(a,i) \cdot \mathrm{end}(a,i')$

1-я ветка: $\mathrm{beg}(\mathrm{end}(a,i'),j'') = \mathrm{beg}(\mathrm{beg}(\mathrm{end}(a,i'),j''),j') \cdot \mathrm{end}(\mathrm{beg}(\mathrm{end}(a,i'),j''),j'')$

2-я ветка: $\mathrm{end}(\mathrm{beg}(a,i'+j'),i') = \mathrm{beg}(\mathrm{end}(\mathrm{beg}(a,i'+j'),i'),j') \cdot \mathrm{end}(\mathrm{end}(\mathrm{beg}(a,i'+j'),i'),j'')$

И теперь надо доказать их равенство на основании предположения индукции.

1-я ветка 1-й член конкатенации:

$\mathrm{beg}(\mathrm{beg}(\mathrm{end}(a,i'),j''),j') = \mathrm{beg}(\mathrm{end}(a,i'),j')$, по (3.3)

1-я ветка 2-й член конкатенации:

$\mathrm{end}(\mathrm{beg}(\mathrm{end}(a,i'),j''),j'') = \mathrm{beg}(\mathrm{end}(\mathrm{end}(a,i'),j''),1)$, по (4.2) для «внешней» композиции

$\mathrm{beg}(\mathrm{end}(\mathrm{end}(a,i'),j''),1) = \mathrm{beg}(\mathrm{end}(a,i'+j'),1)$, из (т.3.2) для «внутренней» композиции

2-я ветка 1-й член конкатенации:

$\mathrm{beg}(\mathrm{end}(\mathrm{beg}(a,i'+j'),i'),j') = \mathrm{end}(\mathrm{beg}(\mathrm{beg}(a,i'+j'),i'+j),i')$, по предположению индукции, где вместо a подставлено $\mathrm{beg}(a,i'+j')$

$\mathrm{end}(\mathrm{beg}(\mathrm{beg}(a,i'+j'),i'+j),i') = \mathrm{end}(\mathrm{beg}(a,i'+j),i')$, по (3.3) для внутренней композиции

$\mathrm{end}(\mathrm{beg}(a,i'+j),i') = \mathrm{beg}(\mathrm{end}(a,i'),j')$, по предположению индукции.

Совпадение с 1-й веткой 1-м членом конкатенации

2-я ветка 2-й член конкатенации:

$\mathrm{end}(\mathrm{end}(\mathrm{beg}(a,i'+j'),i'),j'') = \mathrm{end}(\mathrm{beg}(a,i'+j'),i'+j')$, из (т.3.2) для «внешней» композиции

$\mathrm{end}(\mathrm{beg}(a,i'+j'),i'+j') = \mathrm{beg}(\mathrm{end}(a,i'+j'),1)$, по (4.2)

Совпадение с 1-й веткой 2-м членом конкатенации

Поскольку в обеих конкатенациях совпали 1-е члены с 1-ми и 2-е со 2-ми, то обе конкатенации одинаковы и доказано $\mathrm{beg}(\mathrm{end}(a,i'),j'') = \mathrm{end}(\mathrm{beg}(a,i'+j'),i')$ на основании индуктивного предположения. Это завершает доказательство по индукции.

(т.4.3) $(i \neq 0) \Rightarrow \mathrm{str}(a,i,j') = \mathrm{str}(a,i,j) \cdot \mathrm{str}(a,i+j,1)$

Альтернативное обозначение: $(i \neq 0 \wedge j \neq 0) \Rightarrow a.\mathrm{str}(i,j) = a.\mathrm{beg}(i+j-1).\mathrm{end}(i)$

Пояснение. Это та же теорема (т.4.1), переписанная с учётом определения (4.1).

У теоремы (т.4.2) есть практическое значение – если у нас имеется очень длинная (даже бесконечная) строка a, то на практике для получения подстроки $\mathrm{str}(a,i,j)$ нужно поступить иначе, чем $\mathrm{beg}(\mathrm{end}(a,i),j)$. Потому что делать буквально по определению (4.1) – это значит, в промежуточном вычислении $\mathrm{end}(a,i)$ создавать значение из «хвоста» значения a от символа i включительно. А «хвост» значения a может быть как угодно велик, и там может быть сколько угодно символов, не нужных для окончательного результата. Поэтому лучше исполнить промежуточное вычисление $\mathrm{beg}(a,i+j-1)$, создав значение из «начала» значения a до $i+j-1$ символа включительно, и лишь затем сформировать результата из «хвоста» этого промежуточного значения. При втором методе «хвост» промежуточного значения целиком попадает в окончательный результат без обработки ненужных символов.

(т.4.3) $\mathrm{str}(a,i,j') = \mathrm{str}(a,i,j) \cdot \mathrm{str}(a,i+j,1)$

Пояснение. Это простое следствие из анализа доказательства (т.4.1). Конкатенация тут состоит из 1-го и 2-го членов конкатенации ветки 1 (Или ветки 2 – без разницы).

Теорема (т.4.3) раскрывает нам то, как подстрока «построена» из своих «атомов». Всё начинается с «атома» $\mathrm{str}(a,i,1)$, а дальше «атомы» добавляются к самому концу строки по одному, до j-го «атома» из строки a, если речь у нас о подстроке $\mathrm{str}(a,i,j)$

(т.4.4) $j < k \Rightarrow \mathrm{str}(\mathrm{str}(a,i,j),k,1) = \ominus$

Альтернативное обозначение: $j < k \Rightarrow a.\mathrm{str}(i,j).\mathrm{str}(k,1) = \ominus$

Вытекает из (т.3.3) $i < k \Rightarrow \mathrm{end}(\mathrm{beg}(a,i),k) = \ominus$. Действительно:

$\mathrm{str}(\mathrm{str}(a,i,j),k,1) = \mathrm{beg}(\mathrm{end}(\mathrm{beg}(\mathrm{end}(a,i),j),k),1)$, по определению для $\mathrm{str}()$;

$\mathrm{beg}(\mathrm{end}(\mathrm{beg}(\mathrm{end}(a,i),j),k),1) = \mathrm{end}(\mathrm{beg}(\mathrm{beg}(\mathrm{end}(a,i),j),k),k)$, по (4.2) для «внешней» композиции;

$\mathrm{end}(\mathrm{beg}(\mathrm{beg}(\mathrm{end}(a,i),j),k),k) = \mathrm{end}(\mathrm{beg}(\mathrm{end}(a,i),j),k)$, из (т.3.1) для «средней» композиции, так как $k > j$;

$\mathrm{end}(\mathrm{beg}(\mathrm{end}(a,i),j),k) = \ominus$, из (т.3.3) для «внешней» композиции, так как $k > j$. Это завершает доказательство.

(т.4.5) $1 < k \Rightarrow \mathrm{str}(\mathrm{str}(a,i,1),k,1) = \ominus$

Альтернативное обозначение: $1 < k \Rightarrow a.\mathrm{str}(i,1).\mathrm{str}(k,1) = \ominus$

Очевидное следствие из (т.4.4), которое показывает, что подстрока, полученная с одного (i-го) места строки, не имеет продолжения со 2-й своей позиции включительно.

Следующие далее теоремы для подстроки $\mathrm{str}(a,1,j)$ будут позже перенесены и на любые конечные строки, потому, что мы докажем, что для конечных строк верно $a = \mathrm{str}(a,1,\mathrm{len}(a))$

(т.4.6) $\mathrm{str}(a,1,j) = \mathrm{beg}(a,j)$

Действительно:

$\mathrm{str}(a,1,j) = \mathrm{beg}(\mathrm{end}(a,1),j)$ по определению $\mathrm{str}(a,1,j)$

$\mathrm{beg}(\mathrm{end}(a,1),j) = \mathrm{beg}(a,j)$ по (3.4).

(т.4.7) $j \le k \Rightarrow \mathrm{str}(\mathrm{str}(a,1,j),1,k) = \mathrm{str}(a,1,j)$

Альтернативное обозначение: $j \le k \Rightarrow a.\mathrm{str}(1,j).\mathrm{str}(1,k) = a.\mathrm{str}(1,j)$

Очевидное следствие теоремы (т.3.1) $\mathrm{beg}(a,\min(i,j)) = \mathrm{beg}(\mathrm{beg}(a,i),j)$ и предыдущей теоремы.

(т.4.8) $\mathrm{end}(\mathrm{str}(a,1,j),k) = \mathrm{str}(\mathrm{str}(a,1,j),k,j)$, притом вместо одного данного $\mathrm{end}()$ может быть композиция из подобных $\mathrm{end}()$ сколь угодно большой глубины вложенности.

Альтернативное обозначение: $a.\mathrm{str}(1,j).\mathrm{end}(1,k) = a.\mathrm{str}(1,j).\mathrm{str}(k,j)$

В отношении «сколь угодно большой глубины вложенности» – это очевидно из (т.3.2):

$(i \ne 0 \wedge j \ne 0) \Rightarrow \mathrm{end}(a,i+j-1) = \mathrm{end}(\mathrm{end}(a,i),j)$

Понятно, что для случая равенства 0 второго аргумента в хотя бы одном из $\mathrm{end}()$ равенство в (т.4.8) будет выполнено. Так как обе его стороны будут равны $\ominus$ в силу аксиом пустых строк (2) и определения $\mathrm{str}()$. Если же числовые аргументы не равны 0, то любая вложенность $\mathrm{end}()$ сводится к одному $\mathrm{end}()$. Поэтому, если (т.4.8) истинна для одного $\mathrm{end}()$, то если

$b = \mathrm{str}(\mathrm{str}(a,1,j),k,j)$

тогда и дальше можно использовать прежнее число j для дальнейшей замены функции $\mathrm{end}()$ на функцию $\mathrm{str}()$:

$\mathrm{end}(b,n) = \mathrm{str}(b,n,j)$

Поэтому нам достаточно рассмотреть единственную функцию $\mathrm{end}()$, а не их композицию, при этом $k \ne 0$ в (т.4.8).

$\mathrm{end}(\mathrm{str}(a,1,j),k) = \mathrm{end}(\mathrm{beg}(a,j),k)$, по (т.4.6) – 1-я ветка для левой части равенства (т.4.8)

Теперь разберём 2-ю ветку для правой части равенства (т.4.8)
$\mathrm{str}(\mathrm{str}(a,1,j),k,j) = \mathrm{str}(\mathrm{beg}(a,j),k,j)$, из (т.4.6)
$\mathrm{str}(\mathrm{beg}(a,j),k,j) = \mathrm{end}(\mathrm{beg}(\mathrm{beg}(a,j),k+j-1),k)$, из (т.4.3)
$\mathrm{end}(\mathrm{beg}(\mathrm{beg}(a,j),k+j-1),k) = \mathrm{end}(\mathrm{beg}(a,j),k)$, из (т.3.1) и $k \neq 0$
Первая и вторая ветка совпали, поэтому (т.4.8) доказана.
(т.4.9) $\mathrm{str}(a,1,j) = \mathrm{str}(\mathrm{str}(a,1,j),1,k) \cdot \mathrm{str}(\mathrm{str}(a,1,j),k',j)$
Альтернативное обозначение: $a.\mathrm{str}(1,j) = a.\mathrm{str}(1,j).\mathrm{str}(1,k) \cdot a.\mathrm{str}(1,j).\mathrm{str}(k',j)$
Действительно:
$\mathrm{str}(a,1,j) = \mathrm{beg}(\mathrm{str}(a,1,j),k) \cdot \mathrm{end}(\mathrm{str}(a,1,j),k')$, по (1)
1-я ветка для 1-го члена конкатенации:
$\mathrm{beg}(\mathrm{str}(a,1,j),k) = \mathrm{str}(\mathrm{str}(a,1,j),1,k)$, из (т.4.6)
2-я ветка для 2-го члена конкатенации:
$\mathrm{end}(\mathrm{str}(a,1,j),k') = \mathrm{str}(\mathrm{str}(a,1,j),k',j)$, из (т.4.8)
Обе ветки совпали с соответствующими членами конкатенации в (т.4.9). Теорема доказана.

II. Алфавит строк

Аксиомы об «атомах», из которых образованы строки:

(5.1) $\forall a \forall i \exists j \ \mathrm{str}(a, i, 1) = \mathrm{Chr}(j)$

Пояснение. У нас есть «алфавит» (в следующей аксиоме (5.2)) и на произвольном месте i в произвольной строке a находится некоторый вариант значения (из конечного набора значений) этого алфавита. И рассматриваем мы при этом только одну позицию в строке, поэтому третий аргумент в функции $\mathrm{str}(a, i, 1)$ равен 1.

(5.2) Аксиома об «алфавите»:

$(\forall i > l_{ChrLim} : \mathrm{Chr}(i) = \ominus)$

$\wedge(\forall i \leq l_{ChrLim} : \mathrm{Chr}(i) \neq \ominus \wedge \mathrm{Chr}(i) = \mathrm{str}(\mathrm{Chr}(i), 1, 1))$

$\wedge(\forall i \leq l_{ChrLim} \forall j \leq l_{ChrLim} : \mathrm{Chr}(i) = \mathrm{Chr}(j) \Rightarrow i = j)$

Пояснение:

Данная аксиома описывает «алфавит», из символов и пустой строки, из которого строятся строки. Символы нумеруются от 0 до l_{ChrLim} включительно, а пустая строка соответствует любому номеру, большему, чем l_{ChrLim}. Будем называть «элементом алфавита» как любой символ, так и пустую строку. Символом будем называть любой элемент алфавита, отличный от пустой строки.

Для названий переменных будем придерживаться таких условностей, соответствующих их значениям:

smb, smb_1, smb_2, ... для переменных, имеющих в качестве своего значения символ;

abc, abc_1, abc_2, ... для переменных, имеющих в качестве своего значения элемент алфавита.

Теперь разберём аксиому (5.2).

Если число i больше, чем l_{ChrLim} то функция $\mathrm{Chr}(i)$ возвращает пустую строку:

$\forall i > l_{ChrLim} : \mathrm{Chr}(i) = \ominus$

Как мы увидим ниже, у пустой строки даже длина равна 0, а у символа длина равна 1 в теории строк. Но из всех аксиом выше, включая аксиому алфавита, такой вывод получить невозможно, как будет показано далее.

Если же число i не превышает l_{ChrLim}, то результат функция $\mathrm{Chr}(i)$ – символ. При этом символ по свойствам совпадает со своей собственной подстрокой из единственной первой позиции:

$\forall i \leq l_{ChrLim} : \mathrm{Chr}(i) \neq \ominus \wedge \mathrm{Chr}(i) = \mathrm{str}(\mathrm{Chr}(i), 1, 1)$

Кстати, пустая строка тоже равна подстроке из своей первой позиции – но это можно вывести из аксиом (2.2), (2.4) и определения (4.1) для $\mathrm{str}()$.

При этом все символы отличаются друг от друга, если их номера отличаются:

$\forall i \leq l_{ChrLim} \forall j \leq l_{ChrLim} : \mathrm{Chr}(i) = \mathrm{Chr}(j) \Rightarrow i = j$

Данная аксиома вводит предметную константу – некоторое число l_{ChrLim}. Первоначально я записывал аксиому с квантором существования по l_{ChrLim}, но, по сути, l_{ChrLim} может быть любым натуральным числом (даже нулём), чтобы в алфавите был хотя бы один символ, отличный от пустой строки.

Можно, конечно, представить себе кодировку, в которой у первого символа «номер» отличается от нуля, но это легко привести к «стандартному» отсчёту от нуля и теоретического смысла

рассматривать 2 предметных константы (ещё и номер первого символа алфавита), вместо одной (номер последнего символа) – нет. С прикладной точки зрения кодировки, в которых номер первого символа больше нуля – тоже не используются. Поэтому и с практической точки зрения нет смысла считать, что номер первого символа может отличаться от нуля.

Разумеется, аксиома алфавита независима от всех предыдущих аксиом, так как можно построить систему с бесконечным количеством символов, которые тоже образуют «цепочки», отвечающие всем аксиомам до (не включая) аксиомы алфавита. Например – массивы натуральных чисел, которые используется в программировании.

Поэтому сформулируем метатеорему – то есть, теорему о теории:

(м.5.1) Аксиома алфавита (5.2) независима от предыдущих аксиом, так как массивы произвольных натуральных чисел удовлетворяют предыдущим аксиомам, но не удовлетворяют аксиоме (5.2).

(5.3) Определение функции $\mathrm{Ach}(u)$, обратной к функции $\mathrm{Chr}(i)$:

$(\mathrm{Ach}(u) = l'_{ChrLim} \wedge u = \ominus)$

$\vee(\mathrm{Ach}(u) = i \wedge u \neq \ominus \wedge \mathrm{Chr}(i) = u)$

$\vee(\mathrm{Ach}(u) = \ominus \wedge \forall i\, \mathrm{Chr}(i) \neq u)$

Функция Ach(u) не «идеально» обратная к функции $\mathrm{Chr}(i)$ – для пустой строки она возвращает минимально возможный номер i, при котором $\mathrm{Chr}(i) = \ominus$. Для символа же (то есть – не пустой строки) номер вполне однозначный в силу

$(\forall i \leq l_{ChrLim} \forall j \leq l_{ChrLim} : \mathrm{Chr}(i) = \mathrm{Chr}(j) \Rightarrow i = j)$, из (5.2)

Поэтому легко доказать, что

(т.5.1) $\mathrm{Ach}(\mathrm{Chr}(i)) = \min(i, l'_{ChrLim})$

Кроме того, если аргумент u в $\mathrm{Ach}(u)$ не является элементом алфавита, то $\mathrm{Ach}(u) = \ominus$.

Формула (5.3) является определением – то есть, в (5.3) аксиоматизируется лишь обозначение, но никакой дополнительной логики в теории данная формула не создаёт.

Это следует из метода «введения новых функциональных букв и предметных констант». Подробнее смотри раздел 4, пункт VI.5 данной статьи, там имеется и ссылка на источник.

В частности, данное определение опирается на теорему:

$\exists_1 q((q = l'_{ChrLim} \wedge u = \ominus) \vee (q = i \wedge u \neq \ominus \wedge \mathrm{Chr}(i) = u) \vee (q = \ominus \wedge \forall i\, \mathrm{Chr}(i) \neq u))$

Её доказательство и является обоснованием того, что (5.3) является всего лишь определением. И данное доказательство нужно «всего лишь» для того, чтобы не вводить аксиому для равенства, применительно к функции $\mathrm{Ach}(u)$. Потому что раз функция не имеет дополнительной относительно остальных аксиом логики, то аксиомы равенства будут для неё автоматически выполнены, раз они выполнены для теории, в которой она определена.

Доказательство данной теоремы о существовании и единственности интересно только с точки зрения минимизации количества исходных утверждений теории – её аксиом. В противном случае достаточно просто дописать ещё одну аксиому равенства – пусть она и избыточна, но никакого противоречия в этом нет. Кому же доказательство всё же интересно, то вот какая у него примерная схема:

Дадим обозначение $\mathrm{C}(q,u)$ для формулы:

$(q = l'_{ChrLim} \wedge u = \ominus) \vee (q = i \wedge u \neq \ominus \wedge \mathrm{Chr}(i) = u) \vee (q = \ominus \wedge \forall j\, \mathrm{Chr}(j) \neq u)$

Сначала докажем существование q для произвольного u.

Допустим, истинно $u = \ominus$, тогда истинно

$\mathrm{C}(l'_{ChrLim}, u)$

Это очевидно по первому члену дизъюнкции в $\mathrm{C}(l'_{ChrLim}, u)$.

На основании теоремы дедукции делаем вывод об истинности импликации

$u = \ominus \Rightarrow \mathrm{C}(l'_{ChrLim}, u)$

По аксиоме логики (см. Раздел 4. Приложение):

(b) $\mathrm{A}(a) \Rightarrow \exists x\, \mathrm{A}(x)$

Получаем

$\mathrm{C}(l'_{ChrLim}, u) \Rightarrow \exists q\, \mathrm{C}(q,u)$

По правилу силлогизма из последней формулы и предыдущей $u = \ominus \Rightarrow \mathrm{C}(l'_{ChrLim}, u)$ имеем:

$u = \ominus \Rightarrow \exists q\, \mathrm{C}(q,u)$

Аналогичным образом доказываем ещё 2 формулы:

$(u \neq \ominus \wedge \mathrm{Chr}(i) = u) \Rightarrow \exists q\, \mathrm{C}(q,u)$

$(\forall j\, \mathrm{Chr}(j) \neq u) \Rightarrow \exists q\, \mathrm{C}(q,u)$

Из предпоследней формулы на основании правила вывода логики (см. Раздел 4. Приложение):

(β) Если верно $\mathrm{B}(a) \Rightarrow U$, то верно $\exists x\, \mathrm{B}(x) \Rightarrow U$

получим:

$(\exists j(u \neq \ominus \wedge \mathrm{Chr}(j) = u)) \Rightarrow \exists q\, \mathrm{C}(q,u)$

Перепишем в эквивалентном виде (эквивалентность – см. Раздел 4. Приложение, в VI.8 поменять стороны в эквивалентности на их отрицания и применить VI.7):

$(u \neq \ominus \wedge \exists j\, \mathrm{Chr}(j) = u) \Rightarrow \exists q\, \mathrm{C}(q,u)$

Теперь у нас доказаны 3 формулы:

$u = \ominus \Rightarrow \exists q\, \mathrm{C}(q,u)$

$(u \neq \ominus \wedge \exists j\, \mathrm{Chr}(j) = u) \Rightarrow \exists q\, \mathrm{C}(q,u)$

$(\forall j\, \mathrm{Chr}(j) \neq u) \Rightarrow \exists q\, \mathrm{C}(q,u)$

Из этих трёх формул и аксиомы (см. Раздел 4. Приложение)

III.3) $(A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \vee B \Rightarrow C))$

Применив её 2 раза, получим:

$((u = \ominus) \vee (u \neq \ominus \wedge \exists j\, \mathrm{Chr}(j) = u) \vee (\forall j\, \mathrm{Chr}(j) \neq u)) \Rightarrow \exists q\, \mathrm{C}(q,u)$

Посылка этой импликации является истинной и может быть отброшена по правилу MP. После чего существование доказано.

Насчёт истинной посылки обоснование такое –

Раскроем по закону дистрибутивности часть дизъюнкции

$(u = \ominus) \vee (u \neq \ominus \wedge \exists j\, \mathrm{Chr}(j) = u)$

Получится эквивалентное выражение (то есть, такое, которое можно подставлять вместо исходного в логическую формулу без изменения истинности результата):

$(u = \ominus \vee u \neq \ominus) \wedge (u = \ominus \vee \exists j\, \mathrm{Chr}(j) = u)$

Сократив заведомо истинный член конъюнкции, получим эквивалентное:

$(u = \ominus \vee \exists j\, \mathrm{Chr}(j) = u)$

Так как верно

$u = \ominus \Rightarrow \exists j\, \mathrm{Chr}(j) = u$, по аксиоме (5.1) $\forall a \forall i \exists j\ \mathrm{str}(a, i, 1) = \mathrm{Chr}(j)$ при $i = 0$, то предыдущая дизъюнкция может быть переписана в эквивалентном виде:

$\exists j\, \mathrm{Chr}(j) = u$

Теперь вся посылка

$((u = \ominus) \vee (u \neq \ominus \wedge \exists j\, \mathrm{Chr}(j) = u) \vee (\forall j\, \mathrm{Chr}(j) \neq u))$

Приобрела вид:

$(\exists j\, \mathrm{Chr}(j) = u) \vee (\forall j\, \mathrm{Chr}(j) \neq u)$

Истинность последнего утверждения очевидна и легко выводится. Поэтому мы её сокращаем по правилу MP и существование $\exists q\, \mathrm{C}(q, u)$ доказано.

Теперь надо доказать единственность, то есть, надо доказать, что

$\mathrm{C}(q_1, u) \wedge \mathrm{C}(q_2, u) \Rightarrow q_1 = q_2$

Доказывать будем аналогично тому, как для существования – перебирая возможные посылки. Начнём с $u = \ominus$

Так как все члены дизъюнкции в $\mathrm{C}(q_1, \ominus)$ ложные, кроме первой, то верно:

$q_1 = l'_{ChrLim} \wedge u = \ominus$

Отсюда:

$q_1 = l'_{ChrLim}$

Аналогично:

$q_2 = l'_{ChrLim}$

Из этих 2-х равенств получаем:

$q_1 = q_2$

Поэтому по теореме дедукции имеем:

$u = \ominus \Rightarrow (\mathrm{C}(q_1, u) \wedge \mathrm{C}(q_2, u) \Rightarrow q_1 = q_2)$

Перебрав оставшиеся 2 посылки и соединив их дизъюнкцией в единую истинную посылку по аналогии с доказательством существования, сократим эту истинную посылку и получим:

$\mathrm{C}(q_1, u) \wedge \mathrm{C}(q_2, u) \Rightarrow q_1 = q_2$.

Единственность тоже доказана. Поэтому (5.3) является определением функции $\mathrm{Ach}(u)$.

(5.4) Определение функции $\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j))$, сравнивающей произвольные элементы алфавита $\mathrm{Chr}(i)$ и $\mathrm{Chr}(j)$):

$(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 0 \wedge i > l_{ChrLim} \wedge j > l_{ChrLim})$
$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 0 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i = j)$
$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 1 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i < j))$
$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 1 \wedge i > l_{ChrLim} \wedge j \leq l_{ChrLim})$
$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 2 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i > j)$
$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 2 \wedge i \leq l_{ChrLim} \wedge j > l_{ChrLim})$

Функция $\mathrm{Comp}(abc_1, abc_2)$ здесь сравнивает между собой символы/пустые строки, которые представлены в 1-м и 2-м аргументах. И для одинаковых – возвращает 0. А для разных возвращает 1, когда значение 1-го аргумента меньше второго из-за его меньшего номера в алфавите, либо того, что в первом аргументе – пустая строка, в отличие от второго аргумента. В остальных случаях (2ой аргумент «меньше» 1-го) функция возвращает 2.

Заметим, что функция определена пока только для элементов алфавита, а не для произвольных строк, как $\mathrm{Ach}(u)$. Дело в том, что для произвольных строк функция $\mathrm{Comp}(a, b)$ будет доопределена после формулировки аксиомы равенства для строк.

Пока же, как видно из (5.4), формула строится даже не на элементах алфавита, а на числах. И ещё желательно доказать корректность такого построения. Ведь даже сама формулировка $\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j))$ – это композиция функций, «на входе» которой два числа i, j и результат – тоже число. Но вопрос, разве функция $\mathrm{Chr}(i)$ никогда не уничтожает какие-то необходимые для работы функции $\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j))$ подробности о числе i?

Перечислим детали определения.

Если оба аргумента являются пустыми строками – то функция равна 0. А если оба аргумента – символы (не пустые строки) и при этом имеют один номер в алфавите – то функция тоже возвращает 0:

$(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 0 \wedge i > l_{ChrLim} \wedge j > l_{ChrLim})$

$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 0 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i = j)$

Если аргументы являются символами и номер первого символа в алфавите меньше номера второго, то функция возвращает 1. Если же значением первого аргумента является пустая строка, а значением второго является символ (не пустая строка), то функция тоже возвращает 1:

$(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 1 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i < j))$

$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 1 \wedge i > l_{ChrLim} \wedge j \leq l_{ChrLim})$

Если аргументы являются символами и номер первого символа в алфавите больше номера второго, то функция возвращает 2. Если же значением первого аргумента является символом, а значением второго является пустая строка, то функция тоже возвращает 2:

$(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 2 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i > j)$

$\vee(\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j)) = 2 \wedge i \leq l_{ChrLim} \wedge j > l_{ChrLim})$

Для тех, кому интересна минимизация количества аксиом равенства, приведу схему доказательства того, что (5.4) является определением для $\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j))$.

Докажем это в два этапа.

Первый этап – доказательство

$\exists_1 n\, \mathrm{C}(n, i, j)$

где $\mathrm{C}(n, i, j)$ служит обозначением для формулы

$(n = 0 \wedge i > l_{ChrLim} \wedge j > l_{ChrLim})$

$\vee(n = 0 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i = j)$

$\vee(n = 1 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i < j))$

$\vee(n = 1 \wedge i > l_{ChrLim} \wedge j \leq l_{ChrLim})$

$\vee(n = 2 \wedge i \leq l_{ChrLim} \wedge j \leq l_{ChrLim} \wedge i > j)$

$\vee(n = 2 \wedge i \leq l_{ChrLim} \wedge j > l_{ChrLim})$

Доказательство существования и единственности для n тут проводится аналогично со случаем (5.3). Только вариантов посылок тут будет шесть, а не три. Но очевидно, что они охватят все варианты возможных значений для i, j и дизъюнкция посылок в итоге станет истинной и будет отброшена по правилу вывода MP как при доказательстве существования, так и при доказательстве единственности. Поэтому

$\exists_1 n\, \mathrm{C}(n, i, j)$ доказано.

Второй этап – перейти от чисел к элементам алфавита.

Можно вывести в рамках арифметики и достаточно очевидно, что:

$\mathrm{C}(n, i, j) \Leftrightarrow \mathrm{C}(n, \min(i, l'_{ChrLim}), \min(j, l'_{ChrLim}))$

Доказывается данная эквивалентность на основе эквивалентности каждого члена дизъюнкции в $\mathrm{C}(n, i, j)$ и $\mathrm{C}(n, \min(i, l'_{ChrLim}), \min(j, l'_{ChrLim}))$ соответственно.

С другой стороны,

$\min(i, l'_{ChrLim}) = \mathrm{Ach}(\mathrm{Chr}(i))$, из-за (т.5.1)

Таким образом:

$\mathrm{C}(n, i, j) \Leftrightarrow \mathrm{C}(n, \mathrm{Ach}(\mathrm{Chr}(i)), \mathrm{Ach}(\mathrm{Chr}(j)))$

Поэтому:

$\mathrm{C}(n, i, j) \Leftrightarrow \mathrm{C}(n, \mathrm{Ach}(abc_1), \mathrm{Ach}(abc_2))$

Мы построили формулу, эквивалентную $\mathrm{C}(n, i, j)$ для которой в силу эквивалентности выполнено:

$\exists_1 n\, \mathrm{C}(n, \mathrm{Ach}(abc_1), \mathrm{Ach}(abc_2))$

И эта формула представляет собой основу для определения (5.4), так как зависит уже от элементов алфавита, а не от их номеров. Поэтому то, что (5.4) является определением для $\mathrm{Comp}(\mathrm{Chr}(i), \mathrm{Chr}(j))$ – доказано.

III. Концы строк, соединение строк

Аксиома о конце строки, для случая, когда у строки есть конец:

(6.1) $(i \neq 0 \wedge \mathrm{str}(a,i,1) = \ominus) \Rightarrow \mathrm{str}(a,i',1) = \ominus$

Пояснение.

Аксиома конца строк не может быть выведена из предыдущих аксиом. Действительно, рассмотрим случай, когда все «цепочки» элементов алфавита бесконечны, а после $\ominus$ в «цепочке» может идти любой другой элемент алфавита. А функция $\mathrm{beg}()$ (и $\mathrm{str}()$ соответственно) возвращает тоже бесконечные «цепочки» элементов алфавита – добавляя после последнего элемента алфавита, извлеченному из исходной «цепочки», ещё и бесконечную «цепочку» из $\ominus$.

Изложенная в предыдущем абзаце интерпретация может быть построена в реальности, поэтому не противоречива и эти «цепочки» соответствуют всем изложенным до аксиомы (6.1) аксиомам – не включая аксиому (6.1).

Поэтому доказана метатеорема:

(м.6.1) Аксиома (6.1) независима от предыдущих аксиом, так как случай, когда пустая строка может чередоваться с другими элементами алфавита внутри «строки», соответствует всем аксиомам перед аксиомой (6.1), а аксиоме (6.1) – не соответствует.

Именно аксиома (6.1) придаёт символу $\ominus$ особый смысл, превращая его наличие в строке – на первом месте или после любого символа (не равного пустой строке $\ominus$, разумеется) строки – в критерий окончания строки.

Очевидно (доказывается по индукции), что:

(т.6.1) $0 < i \leq j \Rightarrow (\mathrm{str}(a,i,1) = \ominus \Rightarrow \mathrm{str}(a,j,1) = \ominus)$

(6.2) Определение для $\mathrm{len}(a)$:

$(\mathrm{len}(a) = 0 \wedge \mathrm{str}(a,1,1) = \ominus)$

$\vee(\mathrm{len}(a) = i \wedge \mathrm{str}(a,i,1) \neq \ominus \wedge \mathrm{str}(a,i',1) = \ominus)$

$\vee(\mathrm{len}(a) = \ominus \wedge \forall i\ \mathrm{str}(a,i,1) \neq \ominus)$

Пояснение.

Если на первом же месте строки стоит пустая строка (то есть, если $\mathrm{str}(a,1,1) = \ominus$), то длина строки считается нулевой:

$\mathrm{len}(a) = 0 \wedge \mathrm{str}(a,1,1) = \ominus$

Если же в строке есть символ (отличный от пустой строки элемент алфавита), то номер места последнего такого символа в строке считается длиной строки:

$\mathrm{len}(a) = i \wedge \mathrm{str}(a,i,1) \neq \ominus \wedge \mathrm{str}(a,i',1) = \ominus$

А то, что после такого символа уже нет символов – обеспечено из-за (т.6.1).

Если же строка a вообще нигде не кончается – то есть, на каждом её месте имеется символ (отличный от пустой строки элемент алфавита), то значением функции $\mathrm{len}(a)$ является пустая строка $\ominus$:

$\mathrm{len}(a) = \ominus \wedge \forall i\ \mathrm{str}(a,i,1) \neq \ominus$

Случай $\mathrm{len}(a) = i$ отличается от случая $\mathrm{len}(a) = \ominus$ из-за того, что $i \neq \ominus$ по (2.6).

Доказательство того, что (6.2) является определением, проводится аналогично доказательству для (5.3).

(т.6.2) $\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a, \mathrm{len}(a), 1) \neq \ominus)$

Следует из определения (6.2). Вариант посылки

$(\mathrm{len}(a) = i \wedge \mathrm{str}(a, i, 1) \neq \ominus \wedge \mathrm{str}(a, i', 1) = \ominus)$

Очевидно соответствует (т.6.2).

Вариант посылки:

$(\mathrm{len}(a) = 0 \wedge \mathrm{str}(a, 1, 1) = \ominus)$

Тоже соответствует в силу того, что в нём и записано

$\mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus$, так как $1 = \mathrm{len}(a)'$ в данном случае.

К полученным импликациям добавляем «лишнюю» посылку, что не меняет истинности, и получим 2 истинных импликации:

$\mathrm{len}(a) \neq \ominus$

$\Rightarrow ((\mathrm{len}(u) = i \wedge \mathrm{str}(a, i, 1) \neq \ominus \wedge \mathrm{str}(a, i', 1) = \ominus) \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus) \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a, \mathrm{len}(a), 1) \neq \ominus)$

$\mathrm{len}(a) \neq \ominus$

$\Rightarrow ((\mathrm{len}(u) = 0 \wedge \mathrm{str}(a, 1, 1) = \ominus) \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a, \mathrm{len}(a), 1) \neq \ominus))$

Заодно добавим истинное:

$\mathrm{len}(a) \neq \ominus$

$\Rightarrow ((\mathrm{len}(u) = \ominus \wedge \forall i\ \mathrm{str}(a, i, 1) \neq \ominus) \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a, \mathrm{len}(a), 1) \neq \ominus))$

Последнее верно потому, что вложенные импликации эквивалентны импликации с конъюнкцией посылок, а в данном случае у нас противоречие в конъюнкции посылок:

$\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(u) = \ominus$ А из противоречия следует всё.

Теперь во всех 3 импликациях меняем местами вторую посылку с первой (это даёт эквивалентное утверждение) и получим:

$Part_1 \Rightarrow (\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a, \mathrm{len}(a), 1) \neq \ominus))$

$Part_2 \Rightarrow (\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a, \mathrm{len}(a), 1) \neq \ominus))$

$Part_3 \Rightarrow (\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a, \mathrm{len}(a), 1) \neq \ominus))$

Понятно, что понимается под $Part_1$, $Part_2$, $Part_3$. Из этих 3 импликаций получим по аксиоме логики III.3 из Раздела 4. Приложение, применив её 2 раза:

$Part_1 \vee Part_2 \vee Part_3 \Rightarrow (\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a, \mathrm{len}(a), 1) \neq \ominus))$

Но посылка в данной импликации – это определение (6.2), то есть – истинное утверждение. Отбрасываем её по правилу MP. Теорема доказана.

(т.6.3) $\mathrm{len}(a) \neq \ominus \Rightarrow (\mathrm{len}(a) < i \Rightarrow \mathrm{str}(a, i, 1) = \ominus)$

Доказательство – из (т.6.1) $0 < i \leq j \Rightarrow (\mathrm{str}(a, i, 1) = \ominus \Rightarrow \mathrm{str}(a, j, 1) = \ominus)$ и (т.6.2) $\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{str}(a, \mathrm{len}(a)', 1) = \ominus$.

Следующие 2 теоремы позволяют «поглощать» функции $\mathrm{end}()$ и $\mathrm{beg}()$ – а, значит, и $\mathrm{str}()$ – во вложенной композиции тем $\mathrm{str}()$, который применяется к ним в композиции и извлекает элемент алфавита только из одной позиции строки.

Данные теоремы уместно было бы разместить в подразделе «I. Разбиение строк», но они нужны сейчас, после определения функции $\mathrm{len}()$ длины строки. Так как с их помощью будут доказаны следующие за ними теоремы о длине подстрок.

(т.6.4) $j \le i \Rightarrow \mathrm{str}(\mathrm{beg}(a,i),j,1) = \mathrm{str}(a,j,1)$

Альтернативное обозначение: $0 < j \le i \Rightarrow a.\mathrm{beg}(i).\mathrm{str}(j,1) = a.\mathrm{str}(j,1)$

Доказательство при гипотезе дедукции $0 < j \le i$:

$\mathrm{str}(\mathrm{beg}(a,i),j,1) = \mathrm{end}(\mathrm{beg}(\mathrm{beg}(a,i),j),j) = \mathrm{end}(\mathrm{beg}(a,j),j) = \mathrm{str}(a,j,1)$

Осталось рассмотреть случай $j = 0$:

$\mathrm{str}(\mathrm{beg}(a,i),j,1) = \mathrm{str}(\mathrm{beg}(a,i),0,1) = \mathrm{beg}(\mathrm{end}(\mathrm{beg}(a,i),0),1) = \mathrm{beg}(\ominus,1) = \ominus$

$\mathrm{str}(a,j,1) = \mathrm{str}(a,0,1) = \mathrm{beg}(\mathrm{end}(a,0),1) = \mathrm{beg}(\ominus,1) = \ominus$

Поэтому и для случая $j = 0$ доказано:

$\mathrm{str}(\mathrm{beg}(a,i),j,1) = \mathrm{str}(a,j,1)$

Теорема (т.6.4) доказана.

(т.6.5) $i \ne 0 \wedge j \ne 0 \Rightarrow \mathrm{str}(\mathrm{end}(a,i),j,1) = \mathrm{str}(a,i+j-1,1)$

Альтернативное обозначение: $i \ne 0 \wedge j \ne 0 \Rightarrow a.\mathrm{end}(i).\mathrm{str}(j,1) = a.\mathrm{str}(i+j-1,1)$

Доказательство при гипотезе дедукции $i \ne 0 \wedge j \ne 0$:

$\mathrm{str}(\mathrm{end}(a,i),j,1) = \mathrm{beg}(\mathrm{end}(\mathrm{end}(a,i),j),1) = \mathrm{beg}(\mathrm{end}(i+j-1),1) = \mathrm{str}(a,i+j-1,1)$

Теорема (т.6.5) доказана.

(т.6.6) $\mathrm{len}(a) \ne \ominus \wedge 0 < i \le \mathrm{len}(a) \Rightarrow \mathrm{str}(a,i,1) \ne \ominus$

Доказательство при дедуктивном предположении $\mathrm{len}(a) \ne \ominus \wedge 0 < i \le \mathrm{len}(a)$:

Допустим, имеется такой i, что:

$\mathrm{str}(a,i,1) = \ominus$

Тогда в силу (т.6.1) имеем:

$0 < i \le \mathrm{len}(a) \Rightarrow (\mathrm{str}(a,i,1) = \ominus \Rightarrow \mathrm{str}(a,\mathrm{len}(a),1) = \ominus)$

В силу чего доказано:

$\mathrm{str}(a,\mathrm{len}(a),1) = \ominus$

Но это противоречит (т.6.2)

$\mathrm{len}(a) \ne \ominus \Rightarrow \mathrm{str}(a,\mathrm{len}(a)',1) = \ominus \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a,\mathrm{len}(a),1) \ne \ominus)$

Так как при имеющимся дедуктивном предположении верно:

$\mathrm{str}(a,\mathrm{len}(a),1) \ne \ominus$

Полученное противоречие завершает доказательство.

(т.6.7) $\mathrm{len}(a) \ne \ominus \wedge i \le \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{beg}(a,i)) = i \wedge (\mathrm{len}(\mathrm{end}(a,i)) = \mathrm{len}(a) - i + 1 \vee i = 0)$

Доказательство.

Для случая $i = 0$ очевидно, так как $\mathrm{len}(\mathrm{beg}(a,0)) = 0 \wedge i = 0$.

Поэтому докажем заключение предложения (т.6.7) при дедуктивном предположении:

$\mathrm{len}(a) \ne \ominus \wedge 0 < i \le \mathrm{len}(a)$

Тогда

$\mathrm{str}(\mathrm{beg}(a,i),i,1) = \mathrm{str}(a,i,1)$, из-за (т.6.4)

$\mathrm{str}(a,i,1) \ne \ominus$, из-за (т.6.6)

Поэтому

$\mathrm{str}(\mathrm{beg}(a,i),i,1) \neq \ominus$

С другой стороны:

$\mathrm{str}(\mathrm{beg}(a,i),i',1) = \ominus$, из-за (т.4.4) $j < k \Rightarrow \mathrm{str}(\mathrm{str}(a,i,j),k,1) = \ominus$ при $i = 1$ и равенстве (т.4.6) $\mathrm{str}(a,1,j) = \mathrm{beg}(a,j)$.

Два доказанных утверждения дают в силу определение (6.2) для $\mathrm{len}(u)$:

$\mathrm{len}(\mathrm{beg}(a,i)) = i$, теорема (т.6.7) доказана для первого члена конъюнкции заключения.

$\mathrm{str}(\mathrm{end}(a,i),\mathrm{len}(a) - i + 1,1) = \mathrm{str}(a, i + (\mathrm{len}(a) - i + 1) - 1, 1)$, из-за (т.6.5).

Поэтому:

$\mathrm{str}(\mathrm{end}(a,i),\mathrm{len}(a) - i + 1,1) = \mathrm{str}(a,\mathrm{len}(a),1)$

$\mathrm{str}(a,\mathrm{len}(a),1) \neq \ominus$, из-за (т.6.2) $\mathrm{len}(a) \neq \ominus \Rightarrow \ldots \wedge (\mathrm{len}(a) = 0 \vee \mathrm{str}(a,\mathrm{len}(a),1) \neq \ominus)$.

Получили:

$\mathrm{str}(\mathrm{end}(a,i),\mathrm{len}(a) - i + 1,1) \neq \ominus$

Аналогично:

$\mathrm{str}(\mathrm{end}(a,i),(\mathrm{len}(a) - i + 1)',1) = \mathrm{str}(a,\mathrm{len}(a)',1) = \ominus$

Два последних утверждения доказывают:

$\mathrm{len}(\mathrm{end}(a,i)) = \mathrm{len}(a) - i + 1$

Теорема (т.6.7) доказана теперь и для второго члена конъюнкции заключения при $i \neq 0$.

Поэтому теорема доказана для случаев $i = 0$ и $i \neq 0$. Доказана полностью.

(с.6.7) $\mathrm{len}(a) \neq \ominus \wedge i \neq 0 \wedge i + j - 1 \leq \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{str}(a,i,j)) = j$

Доказательство при предположении дедукции $\mathrm{len}(a) \neq \ominus \wedge i \neq 0 \wedge j \neq 0 \wedge i + j - 1 \leq \mathrm{len}(a)$.

Случай 1. $j = 0$

На основании определения $\mathrm{str}()$ имеем:

$\mathrm{str}(a,i,j) = \mathrm{beg}(\mathrm{end}(a,i),0)$

На основании аксиомы (2.1) $\mathrm{beg}(a,0) = \ominus$ получаем:

$\mathrm{str}(a,i,j) = \ominus$

На основании определения $\mathrm{len}()$ и предыдущего равенства получаем:

$\mathrm{len}(\mathrm{str}(a,i,0)) = 0$

То есть, для Случая 1 ($j = 0$) следствие (с.6.7) доказано:

$\mathrm{len}(\mathrm{str}(a,i,j)) = j$

Случай 2. $j \neq 0$

На основании предположении дедукции, Случая 2 и теоремы (т.4.2) которая (напоминание):

$(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{str}(a,i,j) = \mathrm{end}(\mathrm{beg}(a,i + j - 1),i)$

Получаем:

$\mathrm{str}(a,i,j) = \mathrm{end}(\mathrm{beg}(a,i + j - 1),i)$

На основании теоремы (т.6.7) которая (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge i \leq \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{beg}(a,i)) = i \wedge (\mathrm{len}(\mathrm{end}(a,i)) = \mathrm{len}(a) - i + 1 \vee i = 0)$

Получаем:

$\mathrm{len}(\mathrm{beg}(a,i + j - 1)) = i + j - 1$

Применяя теорему (т.6.7) ещё раз – теперь к $\mathrm{beg}(a, i+j-1)$, получаем:
$\mathrm{len}(\mathrm{end}(\mathrm{beg}(a, i+j-1), i)) = \mathrm{len}(\mathrm{beg}(a, i+j-1)) - i + 1 = (i+j-1) - i + 1 = j$
То есть, для Случая 2 ($j \neq 0$) следствие (с.6.7) тоже доказано.
Следствие доказано.
(т.6.8) $\mathrm{len}(a) = \ominus \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = i \wedge (\mathrm{len}(\mathrm{end}(a, i)) = \ominus \vee i = 0)$
Доказательство.
Для случая $i = 0$ теорема очевидно выполнена. Рассмотрим случай $i \neq 0$.
При $\mathrm{len}(a) = \ominus$ все $\mathrm{str}(a, i, 1) \neq \ominus$ при $i \neq 0$.
Поэтому для $\mathrm{beg}(a, i)$ в доказательстве ничего не меняется, в сравнении с (т.6.7), поэтому:
$\mathrm{len}(\mathrm{beg}(a, i)) = i$
А для $\mathrm{end}(a, i)$ все символы оказываются из a, то есть – для любого $j \neq 0$ верно:
$\mathrm{str}(\mathrm{end}(a, i), i, 1) \neq \ominus$.
Поэтому, по определению (6.2) для $\mathrm{len}()$ получаем:
$\mathrm{len}(\mathrm{end}(a, i)) = \ominus$
Теорема доказана.
(т.6.9) $\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(a) < i \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = \mathrm{len}(a) \wedge \mathrm{len}(\mathrm{end}(a, i)) = 0$
Доказательство при дедуктивном предположении $\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(a) < i$.
Случай 1. $\mathrm{len}(a) = 0$
На основании (4.1) которая (напоминание):
$\mathrm{str}(a, i, j) = \mathrm{beg}(\mathrm{end}(a, i), j)$
Запишем:
$\mathrm{str}(\mathrm{beg}(a, i), 1, 1) = \mathrm{beg}(\mathrm{end}(\mathrm{beg}(a, i), 1), 1)$
На основании (3.4) которая (напоминание):
$a = \mathrm{end}(a, 0')$
убираем $\mathrm{end}()$:
$\mathrm{str}(\mathrm{beg}(a, i), 1, 1) = \mathrm{beg}(\mathrm{beg}(a, 1), 1)$
На основании (т.3.1) $\mathrm{beg}(a, \min(i, j)) = \mathrm{beg}(\mathrm{beg}(a, i), j)$ получим:
$\mathrm{str}(\mathrm{beg}(a, i), 1, 1) = \mathrm{beg}(a, 1)$
На основании (т.4.6) $\mathrm{str}(a, 1, j) = \mathrm{beg}(a, j)$ получим:
$\mathrm{str}(\mathrm{beg}(a, i), 1, 1) = \mathrm{str}(a, 1, 1)$
Но так как $\mathrm{len}(a) = 0$, то по определению $\mathrm{len}()$ имеем:
$\mathrm{str}(a, 1, 1) = \ominus$
Поэтому:
$\mathrm{str}(\mathrm{beg}(a, i), 1, 1) = \ominus$
И по определению $\mathrm{len}()$ получаем:
$\mathrm{len}(\mathrm{beg}(a, i)) = 0$
$\mathrm{len}(\mathrm{beg}(a, i)) = \mathrm{len}(a)$
Теорема (т.6.9) для $\mathrm{beg}(a, i)$ и Случая 1 доказана.
Теперь рассмотрим $\mathrm{str}(\mathrm{end}(a, i), 1, 1)$, тоже раскрывая $\mathrm{str}()$ по определению (4.1):

$\mathrm{str}(\mathrm{end}(a,i),1,1) = \mathrm{beg}(\mathrm{end}(\mathrm{end}(a,i),1),1)$
На основании (3.4) которая (напоминание):
$a = \mathrm{end}(a,0')$
убираем тот $\mathrm{end}()$, где второй аргумент равен 1:
$\mathrm{str}(\mathrm{end}(a,i),1,1) = \mathrm{beg}(\mathrm{end}(a,i),1)$
На основании (4.1) (см. тут чуть выше) получим:
$\mathrm{str}(\mathrm{end}(a,i),1,1) = \mathrm{str}(a,i,1)$
Но для рассматриваемого Случая 1 мы уже получили при рассмотрении $\mathrm{str}(\mathrm{beg}(a,i),1,1)$:
$\mathrm{str}(a,1,1) = \ominus$
На основании (т.6.1) которая (напоминание):
$0 < i \leq j \Rightarrow (\mathrm{str}(a,i,1) = \ominus \Rightarrow \mathrm{str}(a,j,1) = \ominus)$
Получаем:
$\mathrm{str}(a,i,1) = \ominus$
Поэтому
$\mathrm{str}(\mathrm{end}(a,i),1,1) = \ominus$
Из определения для $\mathrm{len}()$ из последнего равенства получаем:
$\mathrm{len}(\mathrm{end}(a,i)) = 0$
Теорема (т.6.9) для $\mathrm{end}(a,i)$ и Случая 1 доказана.
Теорема (т.6.9) для Случая 1 доказана.
Случай 2. $\mathrm{len}(a) \neq 0$
Так как из-за (т.4.2) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{str}(a,i,j) = \mathrm{end}(\mathrm{beg}(a,i+j-1),i)$, то:
$\mathrm{str}(\mathrm{beg}(a,i),\mathrm{len}(a),1) = \mathrm{end}(\mathrm{beg}(\mathrm{beg}(a,i),\mathrm{len}(a)),1)$
Так как (т.3.1) $\mathrm{beg}(a,\min(i,j)) = \mathrm{beg}(\mathrm{beg}(a,i),j)$, то:
$\mathrm{str}(\mathrm{beg}(a,i),\mathrm{len}(a),1) = \mathrm{end}(\mathrm{beg}(a,\mathrm{len}(a)),1)$
И используя снова (т.4.2), но теперь «в другую сторону», получаем:
$\mathrm{str}(\mathrm{beg}(a,i),\mathrm{len}(a),1) = \mathrm{str}(a,\mathrm{len}(a),1)$
Аналогично доказывается
$\mathrm{str}(\mathrm{beg}(a,i),\mathrm{len}(a)',1) = \mathrm{str}(a,\mathrm{len}(a)',1)$
Так как $\mathrm{len}(a) \neq 0$, то по определению имеем:
$\mathrm{str}(a,\mathrm{len}(a),1) \neq \ominus$
$\mathrm{str}(a,\mathrm{len}(a)',1) = \ominus$
Но раз для $\mathrm{beg}(a,i)$ выполнены те же 2 соотношения, что для a, то верно:
$\mathrm{len}(\mathrm{beg}(a,i)) = \mathrm{len}(a)$
Теорема (т.6.9) для $\mathrm{beg}(a,i)$ и Случая 2 доказана.
$\mathrm{str}(\mathrm{end}(a,i),1,1) = \mathrm{beg}(\mathrm{end}(a,i),1)$
На основании (4.1) (см. тут чуть выше) получим:
$\mathrm{str}(\mathrm{end}(a,i),1,1) = \mathrm{str}(a,i,1)$
Но для рассматриваемого Случая 2 мы уже получили при рассмотрении $\mathrm{str}(\mathrm{beg}(a,i),\mathrm{len}(a)',1)$:
$\mathrm{str}(a,\mathrm{len}(a)',1) = \ominus$

На основании (т.6.1) которая (напоминание):

$0 < i \leq j \Rightarrow (\mathrm{str}(a, i, 1) = \ominus \Rightarrow \mathrm{str}(a, j, 1) = \ominus)$

Получаем, так как $\mathrm{len}(a)' \leq i$ из-за дедуктивного предположения $\mathrm{len}(a) < i$:

$\mathrm{str}(a, i, 1) = \ominus$

Поэтому

$\mathrm{str}(\mathrm{end}(a, i), 1, 1) = \ominus$

Из определения для $\mathrm{len}()$ из последнего равенства получаем:

$\mathrm{len}(\mathrm{end}(a, i)) = 0$

Теорема (т.6.9) для $\mathrm{end}(a, i)$ и Случая 2 доказана.

Теорема (т.6.9) для Случая 2 доказана.

Теорема (т.6.9) доказана.

(с.6.9) $\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = \min(i, \mathrm{len}(a)) \wedge (\mathrm{len}(\mathrm{end}(a, i)) = \max(\mathrm{len}(a), i - 1) - i + 1 \vee i = 0)$

Это следствие очевидно из предыдущих теорем - (т.6.7), которая (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge i \leq \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = i \wedge (\mathrm{len}(\mathrm{end}(a, i)) = \mathrm{len}(a) - i + 1 \vee i = 0)$

И теоремы (т.6.9), которая (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(a) < i \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = \mathrm{len}(a) \wedge (\mathrm{len}(\mathrm{end}(a, i)) = 0)$

Как видим из этих теорем:

Значение $\mathrm{len}(\mathrm{beg}(a, i))$ всегда совпадает с тем, что меньше среди двух вариантов: i и $\mathrm{len}(a)$. То есть – равно $\min(i, \mathrm{len}(a))$.

И

Значение $\mathrm{len}(\mathrm{end}(a, i))$ либо 0 при $i = 0$, либо от $\mathrm{len}(a)$ отнимается $(i - 1)$, когда есть от чего отнимать (в (т.6.7), $i - 1 < \mathrm{len}(a)$), либо от уменьшаемого $\mathrm{len}(a)$ ничего не остаётся (в (т.6.9) , $\mathrm{len}(a) \leq i - 1$), когда уменьшаемое меньше вычитаемого. То есть, $\mathrm{len}(\mathrm{end}(a, i))$ оказывается равным арифметической разности между $\mathrm{len}(a)$ и $(i-1)$. А арифметическая разница записывается, например, формулой $\max(\mathrm{len}(a), i - 1) - i + 1$.

Следствие доказано.

(7.1) Аксиома о результате конкатенации при конечной 1-й строке:

$\mathrm{len}(a) \neq \ominus \Rightarrow (i \leq \mathrm{len}(a) \Rightarrow \mathrm{str}(a \cdot b, i, 1) = \mathrm{str}(a, i, 1)) \wedge (i > \mathrm{len}(a) \Rightarrow \mathrm{str}(a \cdot b, i, 1) = \mathrm{str}(b, i - \mathrm{len}(a), 1))$

Пояснение.

Аксиома о результате конкатенации для произвольных строк декларирует сохранении прежнего порядка (как в исходных строках) элементов алфавита на своих новых местах в новой строке относительно «соседей». То есть – начало второй «цепочки» символов «цепляется» к концу первой «цепочки» символов, но исходные «цепочки» не разрываются при создании новой строки.

Аксиома о порядке символов в конкатенации необходима: иначе слияние может быть, например, слиянием 2-х упорядоченных массивов в упорядоченный массив. И все предыдущие аксиомы будут соответствовать такой интерпретации.

При интерпретации «упорядоченные массивы символов» бесконечный 2-й массив полностью «вытесняет» первый массив при их конкатенации, если символы 2-го упорядоченного массива символов идут раньше в алфавите, чем 1-ого. И все аксиомы для упорядоченных массивов символов при разбиении строк остаются в силе – потому что такое извлечение части из массива упорядоченных символов – тоже даёт упорядоченный массив.

Кстати, видимо, можно включить в рассмотрение и случай бесконечной первой строки в конкатенации:

(7.2) Аксиома о результате конкатенации при бесконечной 1-й строке:

$\text{len}(a) = \ominus \Rightarrow a = a \cdot b$

(м.7.1) Аксиомы (7.1) и (7.2) независимы от предыдущих аксиом, так как случай упорядоченных массивов символов соответствует всем аксиомам перед аксиомами (7.1) и (7.2), но не соответствует аксиомам (7.1) и (7.2).

Данная теорема доказана чуть выше. Разумеется, доказательство метатеорем не может быт формализовано в рамках теории, о которой проведено данное доказательство, так как касается вопросов, выходящих за рамки самой теории, о которой сформулированы метатеоремы.

(т.7.1) $\text{len}(a_1) \neq \ominus \wedge \ldots \wedge \text{len}(a_n) \neq \ominus \Rightarrow \text{len}(a_1 \cdot \ldots \cdot a_n) = \text{len}(a_1) + \ldots + \text{len}(a_n)$

Очевидное следствие для $n = 2$ из аксиомы конкатенации и определения $\text{len}()$. Доказывается по индукции, так как $\text{len}(a_1 \cdot \ldots \cdot a_{n'}) = \text{len}((a_1 \cdot \ldots \cdot a_n) \cdot a_{n'})$, поэтому вопрос для n ' сводится к предыдущему n и случаю $n = 2$.

(т.7.2) $i \leq l_{ChrLim} \Rightarrow \text{len}(\text{Chr}(i)) = 1$

Очень удобная теорема вместе с предыдущей – так как не пустая строка равна конкатенации собственных символов. Впрочем, утверждение про «равна конкатенации своих символов» невозможно доказать без теоремы о равенстве строк в следующем разделе.

Но сейчас мы уже можем доказать, что длина конечной строки равна количеству этих символов. Доказательство теоремы (т.7.2) не требует аксиом (7), поэтому её можно было бы доказать перед аксиомой (7.1), но она удобна в связке с предыдущей теоремой (т.7.1), поэтому расположил эти теоремы рядом.

Доказательство при предположении дедукции $i \leq l_{ChrLim}$.

Так как в аксиоме (5.2) есть конъюнктивный член:

$\forall i \leq l_{ChrLim} : \text{Chr}(i) \neq \ominus \wedge \text{Chr}(i) = \text{str}(\text{Chr}(i), 1, 1)$

То верно:

$\text{str}(\text{Chr}(i), 1, 1) = \text{Chr}(i) \neq \ominus$.

С другой стороны, из-за (т.4.5) которая (напоминание):

$1 < k \Rightarrow \text{str}(\text{str}(a, i, 1), k, 1) = \ominus$

Получим:

$\text{str}(\text{Chr}(i), 2, 1) = \text{str}(\text{str}(\text{Chr}(i), 1, 1), 2, 1) = \ominus$

Таким образом у нас выполнено:

$\text{str}(\text{Chr}(i), 1, 1) \neq \ominus \wedge \text{str}(\text{Chr}(i), 2, 1) = \ominus$

В соответствии с определением (6.2) для $\mathrm{len}(a)$ это соответствует 2-му члену в дизъюнкции этого определения при $i = 1$:

$\mathrm{len}(a) = i \wedge \mathrm{str}(a, i, 1) \neq \ominus \wedge \mathrm{str}(a, i', 1) = \ominus$

Поэтому при сделанном дедуктивном предположении $i \leq l_{ChrLim}$ верно:

$\mathrm{len}(\mathrm{Chr}(i)) = 1$

Теорема доказана.

IV. Равенство строк

Аксиома о равенстве строк:

(8.1) $a = b \Leftrightarrow \forall i\ \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1)$

Пояснения.

Аксиома равенства строк – независима от предыдущих. Так как функции взятия подстроки могут сохранять «ссылку» на исходный объект и этим отличаться от «нового» (возникшего в результате конкатенации, например) объекта – это опять опыт из программирования. И без данной аксиомы теорию, составленную из предыдущих аксиом (до 8.1, не включая) можно расширить таким образом, что будет верным:

$\mathrm{str}(\text{«}string\text{»}, 1, 3) \neq \text{«}str\text{»}$, при этом никаких противоречий не возникнет.

Действительно, результат $\text{«}string\text{»}.\mathrm{str}(\text{«}string\text{»}, 1, 3)$ может допускать возможность выяснить – с какой позиции была получена данная подстрока и из какой строки. И получать эти значения можно при помощи свойств $i_{in}(a)$ и $\mathrm{parent}(a)$ соответственно. И тогда будет верно:

$\text{«}string\text{»}.\mathrm{str}(1, 3).i_{in} = 1$

$\text{«}string\text{»}.\mathrm{str}(1, 3).parent = \text{«}string\text{»}$.

В то время как у подстроки, содержащей все символы, свойство $parent$ может возвращать (если так запрограммировать) результат $\text{«}str\text{»}$:

$\text{«}str\text{»}.parent = \text{«}str\text{»}$

Получается, что НЕ верно:

$\text{«}string\text{»}.\mathrm{str}(1, 3) = \text{«}str\text{»}$

Так как, в соответствии с аксиомами равенства (см. Раздел 4 Приложение):

$(J_1)\ a = a$

$(J_2)\ a = b \Rightarrow (\mathrm{A}(a) \Rightarrow \mathrm{A}(b))$

И в случае равенства должно было быть:

$\text{«}string\text{»}.\mathrm{str}(1, 3).parent = \text{«}str\text{»}.parent$, что неверно в разбираемом примере.

А это значит, что без аксиомы (8.1), а только на основе изложенных ранее аксиом, невозможно доказать, что:

$\mathrm{str}(\text{«}string\text{»}, 1, 3) = \text{«}str\text{»}$, потому что у нас есть непротиворечивое расширение теории, в которой доказывается отрицание данного равенства.

Тут есть нюанс: до аксиомы алфавита (не включая аксиому алфавита) все строки (аргументы, которые находятся на месте для произвольных строк в функциях) являются одной и той же строкой, либо её производной (результатом некоторых функций от общей произвольной строки-предка) . Для таких аксиом все равенства подразумевают одинаковые дополнительные свойства ($i_{in}(a)$ и $\mathrm{parent}(a)$).

Но аксиома алфавита и аксиома о результате конкатенации являются единственными аксиомами перед аксиомой о равенстве строк, где присутствуют произвольные строки, которые не являются производными (через некоторые функции) от общей строки-предка. Однако, в обоих этих аксиомах с обеих сторон равенства всегда находятся строки из единственного элемента алфавита каждая.

Поэтому для непротиворечивого построения теории со свойствами $i_{in}(a)$ и $\mathrm{parent}(a)$ надо считать, что они сохраняются лишь для строк, длина которых больше 1. Как только из строки получена пустая строка или подстрока длиной 1 – эти свойства меняются на «базовые». Например, имеется аксиома:

$a.\mathrm{str}(i,1).parent = a.\mathrm{str}(i,1)$

То есть, строка из одного символа и пустая строки всегда – «независимы» и имеют в качестве «предков» самих себя.

А это значит, что всё же можно построить непротиворечивую теорию, расширяющую аксиомы перед аксиомой равенства строк, если считать, что равенство имеется только тогда, когда у равных строк не только последовательность символов $a.\mathrm{str}(i,1)$ одинакова по i, но одинакова последовательность символов по i и в $a.parent.\mathrm{str}(i,1)$.

Таким образом, доказана следующая метатеорема:

(м.8.1) Аксиома (8.1) независима от предыдущих аксиом, так как есть пример из ООП (объектно-ориентированного программирования), когда функции $\mathrm{beg}()$ и $\mathrm{end}()$ создают «детей» у «независимых» строк. И между «независимыми» строками и «детьми» есть разница помимо последовательности символов. Такие объекты ООП соответствуют всем аксиомам перед аксиомой (8.1), но не соответствуют аксиоме (8.1).

Как уже упоминалось, доказательство у метатеорем всегда неформально и в силу этого есть риск пропустить в рассуждениях важный нюанс, из-за пропуска которого доказательство окажется ошибочным. Например, в последней метатеореме можно было упустить нюанс с «независимостью» элементов алфавита. Поэтому к доказательствам метатеорем надо относиться с особой осторожностью.

Впрочем, метатеоремы не влияет на внутреннюю логику теории, к которой они относятся, даже если метатеорема ошибочна. Но в случае ошибочности метатеоремы будет не совсем верным наше неформальное понимание «внутренней логики» теории – что всё равно не создаёт риска построения ошибочных логических выводов средствами самой теории.

Для того, чтобы мы могли использовать знак равенства в нашей теории «обычным» образом (а мы именно это и делали) необходимо, чтобы для теории были истинны 2 следующих аксиомы теории первого порядка с равенством (см. Раздел 4. Приложение):

(J_1) $a = a$

(J_2) $a = b \Rightarrow (\mathrm{A}(a) \Rightarrow \mathrm{A}(b))$

Где $\mathrm{A}(a)$ представляет собой любую логически корректную формулу, построенную из функциональных букв, предикатных букв теории, кванторов (в область действия которых не попадают a, b) и логических связок.

Но на самом деле – тут тонкий момент, чтобы то, что ты понимаешь под знаком равенства, действительно обладало свойствами равенства. И пояснения к аксиоме (8.1) – тому пример. Просто написание знака «=» в некоторых аксиомах не делает предикаты на его основе соответствующим утверждениям (J_1) и (J_2).

Однако у нас есть критерий, который позволяет отнести теорию первого порядка в категорию

теории первого порядка с равенством – см. Раздел 4. Приложение, пункт VI.4. Вот в соответствии с этим пунктом мы и допишем необходимые нам для получения теории первого порядка с равенством утверждения. И допишем их в качестве аксиом:

a. Для каждой функциональной буквы $\mathrm{f_n}()$ с аргументами $x_1, \ldots x_m$ (где m – количество аргументов функции $\mathrm{f_n}(x_1, \ldots x_m)$) должно быть истинно:

$(x_1 = y_1 \wedge \ldots \wedge x_m = y_m) \Rightarrow (\mathrm{f_n}(x_1, \ldots x_m) = \mathrm{f_n}(y_1, \ldots y_m))$

Поэтому:

(8.2) $(x_1 = y_1 \wedge x_2 = y_2) \Rightarrow x_1 \cdot x_2 = y_1 \cdot y_2$

(8.3) $(x = y \wedge i = j) \Rightarrow \mathrm{beg}(x, i) = \mathrm{beg}(y, j)$

(8.3) $(x = y \wedge i = j) \Rightarrow \mathrm{end}(x, i) = \mathrm{beg}(y, j)$

Тут, вроде, надо было бы написать ещё аксиому для функции следования (i'). Но для неё всё нужное уже доказано в теории Пеано – как и для других арифметических операций (сравнения чисел определяются в арифметике, поэтому тоже соответствуют). А теория Пеано тоже является частью нашей теории и будет явно включена в теорию строк в следующем разделе.

(8.4) $(i = j) \Rightarrow \mathrm{Chr}(i) = \mathrm{Chr}(j)$

Заметим, что для тех функций, которые были определены – мы аксиом равенства не пишем, так как соответствующие утверждения выводятся из других аксиом, которые не являются определениями.

b. Для знака равенства должна быть истинна рефлексивность $x = x$. Поэтому:

(8.5) $x = x$

В теории Пеано данное равенство не аксиоматизируется, так как оно следует для натуральных чисел из аксиомы $x + 0 = x$. Но в теории строк я это утверждение аксиоматизирую.

c. Для знака равенства должна быть истинна транзитивность $x = y \Rightarrow (z = x \Rightarrow z = y)$. Поэтому:

(8.6) $x = y \Rightarrow (z = x \Rightarrow z = y)$

Возможно, что некоторые (или даже все) из аксиом равенства (8.2)-(8.6) можно вывести из предыдущих аксиом. Но это уже второстепенная задача, а у любителя математики (я про себя) нет возможности заниматься ещё и этим исследованием. К тому же задача по аккуратному построению теории строк является задачей для математического сообщества, и неправильно взваливать её на одного человека – даже если бы речь шла о профессионале.

Кстати, в учебнике Э. Мендельсона «Введение в математическую логику» доказательство того, что теория Пеано – теория первого порядка с равенством – занимает несколько страниц в Главе 3. Формальная арифметика. Раздел 1. Система аксиом. От леммы 3.1. до следствия 3.3 включительно. Боюсь, что в теории строк аналогичные доказательства заняли бы значительно больше места – а придумать их намного труднее, чем прочитать.

В учебнике А. С. Герасимова «Курс математической логики и теории вычислимости» нужные для равенства утверждения не доказываются, а аксиоматизируются для теории Пеано. Что никаких противоречий не создаёт и тоже вполне возможно. Занятно, что в отличном учебнике Дж. Булос, Р Джеффри – «Вычислимость и логика» вопрос о равенстве для теории Пеано оказывает-

ся упущенным из вида, и среди аксиом теории Пеано нет необходимой аксиомы о транзитивности равенства.

Можно сказать, что вопрос о равенстве для теории строк в данной статье не упущен, но до идеала не «отполирован», чего едва ли можно требовать от статьи.

Ассоциативность конкатенации:

(т.8.1) $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

Очевидная теорема, следует из аксиом (7) и (8.1). Если бы вместо аксиом (7) у нас была аксиома для слияния упорядоченных массивов символов в упорядоченный массив символов, то помимо ассоциативности была доказана ещё и коммутативность. Но конкатенация строк не обладает таким свойством, разумеется.

(т.8.2) $\mathrm{len}(a) \neq \ominus \Rightarrow a = \mathrm{str}(a, 1, \mathrm{len}(a))$, в силу чего в отношении конечной строки a, в виде $\mathrm{str}(a, 1, \mathrm{len}(a))$, можно применять теоремы (т.4.4) и от (т.4.6) до (т.4.9) включительно.

Доказательство.

Исходим из предположения $\mathrm{len}(a) \neq \ominus$.

Будем использовать (т.4.6) $\mathrm{str}(a, 1, j) = \mathrm{beg}(a, j)$.

При $\mathrm{len}(a) = 0$ теорема истинна, так как тогда $a = \ominus$ и $\mathrm{str}(a, 1, \mathrm{len}(a)) = \mathrm{str}(a, 1, 0) = \ominus$, поэтому:

$a = \mathrm{str}(a, 1, \mathrm{len}(a))$

Поэтому нам осталось рассмотреть случай $\mathrm{len}(a) \neq 0 \wedge \mathrm{len}(a) \neq \ominus$.

Из (т.4.2) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{str}(a, i, j) = \mathrm{end}(\mathrm{beg}(a, i + j - 1), i)$ при $j = 1$ получим:

$\mathrm{str}(\mathrm{beg}(a, \mathrm{len}(a)), i, 1) = \mathrm{end}(\mathrm{beg}(\mathrm{beg}(a, \mathrm{len}(a)), i), i)$

При $i \leq \mathrm{len}(a)$, из (т.3.1) $\mathrm{beg}(a, \min(i, j)) = \mathrm{beg}(\mathrm{beg}(a, i), j)$ получим:

$\mathrm{end}(\mathrm{beg}(\mathrm{beg}(a, \mathrm{len}(a)), i), i) = \mathrm{end}(\mathrm{beg}(a, i), i)$

По (4.2) $\mathrm{end}(\mathrm{beg}(a, i), i) = \mathrm{beg}(\mathrm{end}(a, i), 1)$ и определению (4.1) для $\mathrm{str}()$ получим:

$\mathrm{end}(\mathrm{beg}(a, i), i) = \mathrm{str}(a, i, 1)$

Таким образом (раз доказано для случая $i \neq 0 \wedge i \leq \mathrm{len}(a)$):

$0 < i \leq \mathrm{len}(a) \Rightarrow \mathrm{str}(\mathrm{beg}(a, \mathrm{len}(a)), i, 1) = \mathrm{str}(a, i, 1)$

При $\mathrm{len}(a) < i$ из (т.4.4) $j < k \Rightarrow \mathrm{str}(\mathrm{str}(a, i, j), k, 1) = \ominus$ получим:

$\mathrm{str}(\mathrm{str}(a, 1, \mathrm{len}(a)), i, 1) = \ominus$

Так как, (т.4.6) $\mathrm{str}(a, 1, \mathrm{len}(a)) = \mathrm{beg}(a, \mathrm{len}(a))$ то доказано:

$\mathrm{len}(a) < i \Rightarrow \mathrm{str}(\mathrm{beg}(a, \mathrm{len}(a)), i, 1) = \ominus$

С другой стороны, (т.6.3) $\mathrm{len}(a) \neq \ominus \Rightarrow (\mathrm{len}(a) < i \Rightarrow \mathrm{str}(a, 1, i) = \ominus)$, поэтому при данных условиях

$\mathrm{len}(a) < i \Rightarrow \mathrm{str}(a, 1, i) = \ominus$

Из последних 2 равенств получаем:

$\mathrm{len}(a) < i \Rightarrow \mathrm{str}(\mathrm{beg}(a, \mathrm{len}(a)), i, 1) = \mathrm{str}(a, 1, i)$

Таким образом, у нас при сделанном предположении $\mathrm{len}(a) \neq \ominus$ доказаны 2 импликации:

$0 < i \leq \mathrm{len}(a) \Rightarrow \mathrm{str}(\mathrm{beg}(a, \mathrm{len}(a)), i, 1) = \mathrm{str}(a, i, 1)$

$\mathrm{len}(a) < i \Rightarrow \mathrm{str}(\mathrm{beg}(a, \mathrm{len}(a)), i, 1) = \mathrm{str}(a, 1, i)$

Добавляем очевидную третью импликацию, где в заключении стоит равенство, в которой обе стороны равны пустой строке $\ominus$:

$i = 0 \Rightarrow \mathrm{str}(\mathrm{beg}(a, \mathrm{len}(a)), i, 1) = \mathrm{str}(a, i, 1)$

Объединяя их в одну импликацию по аксиоме логики III.3 (см. Раздел 4. Приложение) получим:

$(0 < i \leq \mathrm{len}(a) \vee \mathrm{len}(a) < i \vee i = 0) \Rightarrow \mathrm{str}(\mathrm{beg}(a, \mathrm{len}(a)), i, 1) = \mathrm{str}(a, i, 1)$

В посылке стоит истинное утверждение, поэтому отбрасываем его по правилу вывода MP. Получаем утверждение:

$\mathrm{str}(\mathrm{beg}(a, \mathrm{len}(a)), i, 1) = \mathrm{str}(a, i, 1)$

По аксиоме равенства строк (8.1) получаем:

$a = \mathrm{beg}(a, \mathrm{len}(a))$

То же самое иными обозначениями:

$a = \mathrm{str}(a, 1, \mathrm{len}(a))$

оно истинно при истинности предположения нашего доказательства:

$\mathrm{len}(a) \neq \ominus$

Поэтому по теореме дедукции получаем:

$\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{str}(a, 1, \mathrm{len}(a))$

Что и требовалось доказать.

(с.8.2) $\mathrm{len}(a) \neq \ominus \Rightarrow (\mathrm{len}(a) \leq i \Rightarrow a = \mathrm{str}(a, 1, i))$

Очевидное следствие теорем (т.8.2) и (т.4.7) $j \leq k \Rightarrow \mathrm{str}(\mathrm{str}(a, 1, j), 1, k) = \mathrm{str}(a, 1, j)$

(т.8.3) $a = a \cdot \ominus = \ominus \cdot a$

Очевидные равенства – из свойств пустой строки $\ominus$, аксиом для конкатенации и для равенства строк.

(т.8.4) $a = b \Leftrightarrow \forall i(0 < i \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))$

Очевидное следствие из аксиомы для равенства строк (8.1), так как для случая $i = 0$ равенство функций:

$\mathrm{str}(a, 0, 1) = \mathrm{str}(b, 0, 1)$

выполнено по (4.1) – определению $\mathrm{str}()$, по (2.2) $\mathrm{end}(a, 0) = \ominus$ и (2.4) $\mathrm{beg}(\ominus, q) = \ominus$.

(т.8.5) $\mathrm{len}(a) \neq \ominus \Rightarrow [a = b \Leftrightarrow \forall i((\mathrm{len}(a) = \mathrm{len}(b) \wedge 0 < i \wedge i \leq \mathrm{len}(a)) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))]$

Тоже очевидное следствие – уже из (т.8.4) для равенства конечных строк. Если их длины не равны, то для более короткой (пусть b) будет $\mathrm{str}(b, \mathrm{len}(b)', 1) = \ominus$ при $\mathrm{str}(a, \mathrm{len}(b)', 1) \neq \ominus$, то есть – строки не равны тогда. Если же длины равны, а внутри своих длин строки тоже равны, то за пределами своей длины (одинаковой для них) $\mathrm{str}()$ для каждой такой позиции в этих строках возвращает пустую строку и они равны там тоже:

$\mathrm{len}(a) < i \Rightarrow \mathrm{str}(a, i, 1) = \ominus$, что верно и для строки b из (т.6.3) которая (напоминание):

$\mathrm{len}(a) \neq \ominus \Rightarrow (\mathrm{len}(a) < i \Rightarrow \mathrm{str}(a, 1, i) = \ominus)$.

(т.8.6) $\mathrm{len}(a) \neq \ominus \Rightarrow a = \mathrm{beg}(a \cdot u, \mathrm{len}(a))$

Доказательство. Предположение дедукции - $\mathrm{len}(a) \neq \ominus$.

Сравним $\mathrm{beg}(a \cdot u, \mathrm{len}(a))$ и $\mathrm{beg}(a, \mathrm{len}(a))$. Про последнее мы знаем, что

$\mathrm{beg}(a, \mathrm{len}(a)) = a$, из-за (т.8.2) $\mathrm{len}(a) \neq \ominus \Rightarrow a = \mathrm{str}(a, 1, \mathrm{len}(a))$ и (т.4.6) $\mathrm{str}(a, 1, j) = \mathrm{beg}(a, j)$.

Случай 1. $i \leq \text{len}(a)$

Из-за теоремы (т.6.4) $j \leq i \Rightarrow \text{str}(\text{beg}(a, i), j, 1) = \text{str}(a, j, 1)$ получаем:

$\text{str}(\text{beg}(a \cdot u, \text{len}(a)), i, 1) = \text{str}(a \cdot u, i, 1)$

По аксиоме (7.1) которая (напоминание):

$\text{len}(a) \neq \ominus$

$\Rightarrow (i \leq \text{len}(a) \Rightarrow \text{str}(a \cdot b, i, 1) = \text{str}(a, i, 1)) \wedge (i > \text{len}(a) \Rightarrow \text{str}(a \cdot b, i, 1) = \text{str}(b, i - \text{len}(a), 1))$

Получаем с учётом $i \leq \text{len}(a)$:

$\text{str}(a \cdot u, i, 1) = \text{str}(a, i, 1)$

$\text{str}(\text{beg}(a \cdot u, \text{len}(a)), i, 1) = \text{str}(a, i, 1)$

Устраняя условие Случая 1, получим:

$i \leq \text{len}(a) \Rightarrow \text{str}(a, i, 1) = \text{str}(\text{beg}(a \cdot u, \text{len}(a)), i, 1)$

Случай 2. $\text{len}(a) < i$

$\text{str}(\text{beg}(a \cdot u, \text{len}(a)), i, 1) = \text{str}(\text{str}(a \cdot u, 1, \text{len}(a)), i, 1)$

Из теоремы (т.4.4) $j < k \Rightarrow \text{str}(\text{str}(a, i, j), k, 1) = \ominus$ получаем:

$\text{str}(\text{str}(a \cdot u, 1, \text{len}(a)), i, 1) = \ominus$

Аналогично:

$\text{str}(a, i, 1) = \text{str}((\text{str}(a, 1, \text{len}(a)), i, 1) = \ominus$

Из двух последних предложений:

$\text{str}(a, i, 1) = \text{str}(\text{beg}(a \cdot u, \text{len}(a)), i, 1)$

Устраняя условие Случая 2, получим:

$\text{len}(a) < i \Rightarrow \text{str}(a, i, 1) = \text{str}(\text{beg}(a \cdot u, \text{len}(a)), i, 1)$

Объединяя Случай 1 и Случай 2 по III.3 из раздела 4. Приложение, получим:

$(i \leq \text{len}(a) \vee \text{len}(a) < i) \Rightarrow \text{str}(a, i, 1) = \text{str}(\text{beg}(a \cdot u, \text{len}(a)), i, 1)$

Посылка истинная, поэтому отбрасываем её по MP и получаем:

$\text{str}(a, i, 1) = \text{str}(\text{beg}(a \cdot u, \text{len}(a)), i, 1)$

В соответствии с аксиомой (8.1) равенства строк получаем:

$a = \text{beg}(a \cdot u, \text{len}(a))$

Устраняя предположение дедукции, завершаем доказательство:

$\text{len}(a) \neq \ominus \Rightarrow a = \text{beg}(a \cdot u, \text{len}(a))$, что и требовалось доказать.

(т.8.7) $\text{len}(a) \neq \ominus \Rightarrow u = \text{end}(a \cdot u, \text{len}(a)')$

Доказательство. Предположение дедукции - $\text{len}(a) \neq \ominus$

Если $\text{len}(a) = 0$, то $a \cdot u = \ominus \cdot u = u$. Откуда $\text{end}(a \cdot u, \text{len}(a)') = \text{end}(u, 1) = u$.

Если $\text{len}(u) = 0$, то $a \cdot u = a \cdot \ominus = a$. Откуда $\text{end}(a \cdot u, \text{len}(a)') = \text{end}(a, \text{len}(a)') = \ominus = u$.

Поэтому для $\text{len}(a) = 0 \vee \text{len}(u) = 0$ теорема истинна и исходим из «внутреннего предположения дедукции №1» $(\text{len}(a) \neq 0 \wedge \text{len}(u) \neq 0)$, будем пропускать эту посылку при использовании аксиом (она выполнена).

$\text{str}(\text{end}(a \cdot u, \text{len}(a)'), i, 1) = \text{beg}(\text{end}(\text{end}(a \cdot u, \text{len}(a)'), i), 1)$, по (4.1) – определению $\text{str}()$

$\text{beg}(\text{end}(\text{end}(a \cdot u, \text{len}(a)'), i), 1) = \text{beg}(\text{end}(a \cdot u, \text{len}(a) + i), 1)$,

Из (т.3.2) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{end}(a, i+j-1) = \mathrm{end}(\mathrm{end}(a,i),j)$, но только при $i \neq 0$ или эквивалентном $0 < i$ («внутреннее предположение дедукции №2»):

$\mathrm{beg}(\mathrm{end}(a \cdot u, \mathrm{len}(a)+i), 1) = \mathrm{str}(a \cdot u, \mathrm{len}(a)+i, 1)$, по (4.1) – определению $\mathrm{str}()$

$\mathrm{str}(a \cdot u, \mathrm{len}(a)+i, 1) = \mathrm{str}(u, \mathrm{len}(a)+i-\mathrm{len}(a), 1)$, по (7.1) о конкатенации с конечной строкой и того, что $i > 0$.

Поэтому по всей цепочке предыдущих равенств приравниваем начало с концом:

$\mathrm{str}(\mathrm{end}(a \cdot u, \mathrm{len}(a)'), i, 1) = \mathrm{str}(u, i, 1)$

Устраняем «внутреннее предположение дедукции №2»:

$0 < i \Rightarrow \mathrm{str}(\mathrm{end}(a \cdot u, \mathrm{len}(a)'), i, 1) = \mathrm{str}(u, i, 1)$

Обобщаем данную формулу по i:

$\forall i(0 < i \Rightarrow \mathrm{str}(\mathrm{end}(a \cdot u, \mathrm{len}(a)'), i, 1) = \mathrm{str}(u, i, 1))$

И из (т.8.4) $a = b \Leftrightarrow \forall i(0 < i \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))$ получаем:

$u = \mathrm{end}(a \cdot u, \mathrm{len}(a)')$

Устраняем «внутреннее предположение дедукции №1»:

$(\mathrm{len}(a) \neq 0 \wedge \mathrm{len}(u) \neq 0) \Rightarrow u = \mathrm{end}(a \cdot u, \mathrm{len}(a)')$

Поскольку ранее было доказано:

$(\mathrm{len}(a) = 0 \vee \mathrm{len}(u) = 0) \Rightarrow u = \mathrm{end}(a \cdot u, \mathrm{len}(a)')$

То соединение этих предложений даёт:

$u = \mathrm{end}(a \cdot u, \mathrm{len}(a)')$

Теперь устраняем предположение дедукции и получаем:

$\mathrm{len}(a) \neq \ominus \Rightarrow u = \mathrm{end}(a \cdot u, \mathrm{len}(a)')$, что и требовалось доказать.

(т.8.8) $a = \ominus \Leftrightarrow \mathrm{len}(a) = 0$

В части $a = \ominus \Rightarrow \mathrm{len}(a) = 0$ эта эквивалентность выводится из определения для $\mathrm{len}(\ominus)$, на основании дедуктивного предположения $a = \ominus$ и результата расчёта $\mathrm{len}(a) = 0$. Но, так как в данном подразделе мы разбираем равенство для теории строк, то упомяну, на чём основывается возможность логического вывода $a = \ominus \Rightarrow \mathrm{len}(a) = 0$ для равенства на уровне свойств равенства.

В рамках свойств равенства (если обходиться без теоремы дедукции) выводится:

$a = \ominus \Rightarrow \mathrm{len}(a) = \mathrm{len}(\ominus)$

А затем надо воспользоваться выводимым свойством транзитивности для равенства:

$\mathrm{len}(\ominus) = 0 \Rightarrow (\mathrm{len}(a) = \mathrm{len}(\ominus) \Rightarrow \mathrm{len}(a) = 0)$

Посылка этой импликации истинная и она отбрасывается, после чего остаётся:

$\mathrm{len}(a) = \mathrm{len}(\ominus) \Rightarrow \mathrm{len}(a) = 0$

И из этого утверждения и ранее полученного $a = \ominus \Rightarrow \mathrm{len}(a) = \mathrm{len}(\ominus)$ получаем по правилу силлогизма:

$a = \ominus \Rightarrow \mathrm{len}(a) = 0$

Все подробности вывода свойств для равенства можно прочитать в любом учебнике логики, но они и так самоочевидны для читателя с математической подготовкой уровня программиста, например.

Осталось доказать эквивалентность в части:

$\mathrm{len}(a) = 0 \Rightarrow a = \ominus$

Гипотеза теоремы дедукции: $\mathrm{len}(a) = 0$

Из определения (6.2) для $\mathrm{len}(a)$ при $\mathrm{len}(a) = 0$ устраняем 2 ложных члена дизъюнкции, превратив их в истинные посылки. Вот по такой схеме:

$A \vee B \vee C$ – исходное определение. Преобразуем его в эквивалентное:

$\neg B \Rightarrow (A \vee C)$, но B ложное, так как для него надо $\mathrm{len}(a) = i$, а по условию $\mathrm{len}(a) = 0$. Поэтому $\neg B$ истинное и его можно убрать по MP. Аналогично с C, где надо $\mathrm{len}(a) = \ominus$, а имеем $\mathrm{len}(a) = 0$ и по (2.6) $i \neq \ominus$. После удаления по MP члена C остался член дизъюнкции A. В нашем случае это:

$\mathrm{len}(a) = 0 \wedge \mathrm{str}(a, 1, 1) = \ominus$

В истинной конъюнкции истинным является каждый член конъюнкции, поэтому получаем:

$\mathrm{str}(a, 1, 1) = \ominus$

А это значит, что

$\mathrm{str}(a, i, 1) = \ominus$, из-за (т.6.1) $i \leq j \Rightarrow (\mathrm{str}(a, i, 1) = \ominus \Rightarrow \mathrm{str}(a, j, 1) = \ominus)$

что совпадает с: $\mathrm{str}(\ominus, i, 1) = \ominus$

И по аксиоме равенства строк получаем: $a = \ominus$

Поэтому по теореме дедукции выведено: $\mathrm{len}(a) = 0 \Rightarrow a = \ominus$

Вторая часть эквивалентности доказана, а вместе с ней доказана и эквивалентность (т.8.8).

(с.8.8) $\mathrm{len}(a) = \ominus \Leftrightarrow \mathrm{len}(\mathrm{len}(a)) = 0$

Очевидное следствие из теоремы (т.8.8), которое получается при подстановке в (т.8.8) выражения $\mathrm{len}(a)$ вместо a.

V. Сравнение, поиск, вставка

Лемма о минимальной позиции отличия в строках:

(л.9.1) $\mathrm{str}(a,n,1) \neq \mathrm{str}(b,n,1) \Rightarrow \exists_1 m(\mathrm{str}(a,m,1) \neq \mathrm{str}(b,m,1) \wedge (\forall i < m : \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1)))$

Пояснение: Очевидная лемма о том, что если строки не равны, то есть такой номер m позиции в этих строках, что именно на нем встречается самое первое отличие в строках, а до него (если есть позиции раньше) у строк одинаковые элементы алфавита на соответствующих местах.

Существование m доказывается по индукции для n'. Рассматриваются 2 варианта: при первом варианте имеется i:

$i \leq n$, такой что $\mathrm{str}(a,i,1) \neq \mathrm{str}(b,i,1)$,

а при втором варианте для каждого i:

$i \leq n$ означает $\mathrm{str}(a,i,1) = \mathrm{str}(b,i,1)$.

При первом варианте для n' теорема верна в силу предположения индукции, а во втором – сама позиция n' является искомым решением. Это завершает доказательство по индукции для существования.

Единственность m тоже очевидна: если бы были два отличных m_1 и m_2, то для меньшего – пусть m_1 – было бы верно:

$\mathrm{str}(a,m_1,1) \neq \mathrm{str}(b,m_1,1)$

А для второго было бы верно:

$i < m_2 \Rightarrow \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1)$, в частности, для $i = m_1$ получается (так как $m_1 < m_2$):

$\mathrm{str}(a,m_1,1) = \mathrm{str}(b,m_1,1)$

Что противоречит написанному чуть выше неравенству. Поэтому единственность тоже доказана.

Лемма об эквивалентном добавлении дизъюнктивных членов:

(л.9.2) $A \Rightarrow ((A \Rightarrow \neg C) \Rightarrow (B \Leftrightarrow (B \vee C)))$

Это тавтология, что легко выявить прямой проверкой, но приведу схему вывода ради демонстрации некоторых технических приёмов.

Для доказательства леммы (л.9.2) достаточно доказать, что:

$\neg C \Rightarrow (B \Leftrightarrow (B \vee C))$, так как лемма (л.9.2) следует по правилу силлогизма из данного предложения и тавтологии:

$A \Rightarrow ((A \Rightarrow \neg C) \Rightarrow \neg C)$.

Поэтому сводим эквивалентность к эквивалентной эквивалентности отрицаний:

$\neg C \Rightarrow (\neg B \Leftrightarrow (\neg B \wedge \neg C))$

А последнее предложение очевидно – при дедуктивном предположении $\neg C$ из очевидных тавтологий:

$\neg B \Rightarrow \neg B$

$\neg C \Rightarrow (\neg B \Rightarrow \neg B)$

$\neg C \Rightarrow (\neg B \Rightarrow \neg C)$

$\neg C \Rightarrow (\neg B \Rightarrow \neg B \wedge \neg C)$

$\neg B \wedge \neg C \Rightarrow \neg B$

$\neg C \Rightarrow (\neg B \wedge \neg C \Rightarrow \neg B)$

$\neg C \Rightarrow (\neg B \Leftrightarrow (\neg B \wedge \neg C))$

Теорема доказана.

(9.1) Определение для $\mathrm{CompIn}(a, b)$:

$(\mathrm{str}(a, \mathrm{CompIn}(a, b), 1) \neq \mathrm{str}(b, \mathrm{CompIn}(a, b), 1) \wedge \forall i (i < \mathrm{CompIn}(a, b) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1)))$

$\vee (\mathrm{CompIn}(a, b) = 0 \wedge a = b)$

Пояснение. Эта функция возвращает первое место, на котором в строке a и строке b находятся разные элементы алфавита. А если такого места нет, то $\mathrm{CompIn}(a, b)$ возвращает 0.

Для доказательства того, что (9.1) является определением, будем рассматривать следующую формулу, которую обозначим как $\mathrm{C}(m, a, b)$:

$(\mathrm{str}(a, m, 1) \neq \mathrm{str}(b, m, 1) \wedge (\forall i < m : \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))) \vee (m = 0 \wedge a = b)$

И надо доказать:

$\exists_1 m \, \mathrm{C}(m, a, b)$

Ветка 1.

Докажем данную формулу при условии

$\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1)$

Тогда у нас уже доказано по (л.9.1):

$\exists_1 m (\mathrm{str}(a, m, 1) \neq \mathrm{str}(b, m, 1) \wedge (\forall i < m : \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1)))$

Верной является следующее предложение:

1.1. $\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1) \Rightarrow \neg(m = 0 \wedge a = b)$

Это следует из

$a = b \Rightarrow (\forall i \ \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))$

$(\forall i \ \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1)) \Rightarrow \mathrm{str}(a, n, 1) = \mathrm{str}(b, n, 1)$

И, применяя контрапозицию к краям этой короткой цепочки импликаций, получим:

$\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1) \Rightarrow \neg(a = b)$

К заключению импликации мы можем добавлять любой дизъюнктивный член по аксиоме III.1 раздела 4. Приложение. И по правилу силлогизма получим:

$\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1) \Rightarrow (\neg(a = b) \vee \neg(m = 0))$

Если вынести знак отрицания за скобки в заключении данной импликации, то получим (докажем) нужное нам предложение:

1.1. $\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1) \Rightarrow \neg(m = 0 \wedge a = b)$.

Кроме того, в соответствии с леммой (л.9.2) имеем:

$\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1) \Rightarrow$

$((\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1) \Rightarrow \neg(m = 0 \wedge a = b))$

$\Rightarrow [(\mathrm{str}(a, m, 1) \neq \mathrm{str}(b, m, 1) \wedge (\forall i < m : \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))) \Leftrightarrow \mathrm{C}(m, a, b)]$

$)$

Посылки можно переставлять местами, поэтому предложение 1.1 переставляем на первое место, и убираем его по правилу MP, так как оно истинно. Получим:

$\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1)$

$\Rightarrow [(\mathrm{str}(a,m,1) \neq \mathrm{str}(b,m,1) \wedge (\forall i < m : \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1))) \Leftrightarrow \mathrm{C}(m,a,b)]$

В силу леммы (л.9.1) и последней эквивалентности имеем (заменяя в лемме (л.9.1) формулу под квантором $\exists_1$ на эквивалентную формулу):

$\mathrm{str}(a,n,1) \neq \mathrm{str}(b,n,1) \Rightarrow \exists_1 m\, \mathrm{C}(m,a,b)$. Ветка 1 доказана.

Ветка 2.

Докажем $\exists_1 m\, \mathrm{C}(m,a,b)$ при условии

$a = b$

Очевидно, что

$a = b \Rightarrow \exists_1 m(m = 0 \wedge a = b)$

Теперь аналогично предложению 1.1 в ветке 1 получаем предложение:

2.1. $a = b \Rightarrow \neg(\mathrm{str}(a,m,1) \neq \mathrm{str}(b,m,1) \wedge (\forall i < m : \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1)))$

Действительно:

$a = b \Rightarrow (\forall i\ \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1))$

А далее по цепочке силлогизмов (многоточие в посылке – это заключение предыдущей импликации):

$\ldots \Rightarrow \mathrm{str}(a,m,1) = \mathrm{str}(b,m,1)$

$\ldots \Rightarrow \mathrm{str}(a,m,1) = \mathrm{str}(b,m,1) \vee \neg(\forall i < m : \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1))$

$\ldots \Rightarrow \neg(\mathrm{str}(a,m,1) \neq \mathrm{str}(b,m,1) \wedge (\forall i < m : \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1)))$

Истинность импликации 2.1 очевидна, так как неравенство элементов алфавита на местах m невозможно для равных строк. И данный вывод не требует правил вывода помимо MP, так как опирается на силлогизм и аксиому (а) $\forall x\, \mathrm{A}(x) \Rightarrow \mathrm{A}(a)$ из Раздела 4. Приложение.

Это предложение 2.1 истинно, и мы точно так же сокращаем его после применения леммы (л.9.2), как и в ветке 1. Получим:

$a = b \Rightarrow [(m = 0 \wedge a = b) \Leftrightarrow \mathrm{C}(m,a,b)]$

Поэтому из последней формулы и написанного выше предложения

$a = b \Rightarrow \exists_1 m(m = 0 \wedge a = b)$, получим (заменяя в данном предложении формулу под квантором $\exists_1$ на эквивалентную формулу):

$a = b \Rightarrow \exists_1 m\, \mathrm{C}(m,a,b)$. Ветка 2 доказана.

Объединяя ветки 1 и 2, получим:

$(\mathrm{str}(a,n,1) \neq \mathrm{str}(b,n,1) \vee a = b) \Rightarrow \exists_1 m C(m,a,b)$

По правилу вывода из раздела 4. Приложение:

(β) Если верно $\mathrm{B}(a) \Rightarrow U$, то верно $\exists x\, \mathrm{B}(x) \Rightarrow U$

Получим:

$(\exists n\ \mathrm{str}(a,n,1) \neq \mathrm{str}(b,n,1) \vee a = b) \Rightarrow \exists_1 m\, \mathrm{C}(m,a,b)$

Теперь, чтобы доказать, что $\exists_1 m\, \mathrm{C}(m,a,b)$, достаточно доказать посылку последней импликации и отбросить её по правилу MP. Итак, осталось доказать:

$\exists n(\mathrm{str}(a,n,1) \neq \mathrm{str}(b,n,1) \vee a = b)$

По пункту VI.9 из раздела 4. Приложение:

$\exists n(\mathrm{str}(a,n,1) \neq \mathrm{str}(b,n,1) \vee a = b) \Leftrightarrow (a = b \vee \exists n\ \mathrm{str}(a,n,1) \neq \mathrm{str}(b,n,1))$

По (8.1) $a = b \Leftrightarrow \forall i\ \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1)$ и по VI.7 из раздела 4. Приложение:

$a = b \Leftrightarrow \neg(\exists n\ \mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1))$, поэтому:

$\exists n(\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1) \vee a = b) \Leftrightarrow (a = b \vee \neg(a = b))$

Правая часть эквивалентности доказана из-за тавтологии

$A \vee \neg A$

Таким образом,

$\exists n(\mathrm{str}(a, n, 1) \neq \mathrm{str}(b, n, 1) \vee a = b)$ доказано, а с ним доказано и $\exists_1 m\, \mathrm{C}(m, a, b)$, в силу чего (9.1) является определением.

(с.9.1) $\mathrm{len}(a) \neq \ominus \Rightarrow [0 < \mathrm{CompIn}(a, b) \leq \mathrm{len}(a) \Leftrightarrow \forall u\, \mathrm{CompIn}(a \cdot u, b) = \mathrm{CompIn}(a, b)]$, и то же верно, если в этом следствии заменить $\mathrm{CompIn}(a, b)$ и $\mathrm{CompIn}(a \cdot u, b)$ на $\mathrm{CompIn}(b, a)$ и $\mathrm{CompIn}(b, a \cdot u)$ соответственно.

Пояснение: если строка a конечна ($\mathrm{len}(a) \neq \ominus$) и отличие от строки b есть ($0 < \mathrm{CompIn}(a, b)$) и находится в пределах строки a ($\mathrm{CompIn}(a, b) \leq \mathrm{len}(a)$), то это тот и только тот случай, когда для любой строки $a \cdot u$, начало которой равно конечной строке a, имеется первое отличие от строки b в той же позиции, в которой в строке a находится первое отличается от строки b.

Тут мы впервые доказываем возможность игнорировать в строке всё, кроме её ограниченного начала, но при этом получить интересующий нас результат для всей строки при определённых условиях.

Доказательство.

Предполагаем, что $\mathrm{len}(a) \neq \ominus$ и в условиях этой гипотезы дедукции нам для доказательства (с.9.1) достаточно доказать эквивалентность:

$0 < \mathrm{CompIn}(a, b) \leq \mathrm{len}(a) \Leftrightarrow \forall u\, \mathrm{CompIn}(a \cdot u, b) = \mathrm{CompIn}(a, b)$

Для доказательства данной эквивалентности надо доказать две импликации:

Ветка 1: $0 < \mathrm{CompIn}(a, b) \leq \mathrm{len}(a) \Rightarrow \forall u\, \mathrm{CompIn}(a \cdot u, b) = \mathrm{CompIn}(a, b)$

Ветка 2: $\forall u\, \mathrm{CompIn}(a \cdot u, b) = \mathrm{CompIn}(a, b) \Rightarrow 0 < \mathrm{CompIn}(a, b) \leq \mathrm{len}(a)$

Ветка 1.

Определение (9.1) для $\mathrm{CompIn}(a, b)$, когда $\mathrm{CompIn}(a, b) \neq 0$, и $\mathrm{CompIn}(a, b) \leq \mathrm{len}(a)$ (наши дедуктивные гипотезы):

$(\mathrm{str}(a, \mathrm{CompIn}(a, b), 1) \neq \mathrm{str}(b, \mathrm{CompIn}(a, b), 1) \wedge (\forall i\, i < \mathrm{CompIn}(a, b) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1)))$

Так как для всех i (под квантором общности) в данном предложении верно $i \leq \mathrm{len}(a)$, то будет верно в силу аксиомы (7.1) о результате конкатенации при конечной 1-й строке:

$\mathrm{str}(a \cdot u, i, 1) = \mathrm{str}(a, i, 1)$

И то же самое можно сказать про

$\mathrm{str}(a \cdot u, \mathrm{CompIn}(a, b), 1) = \mathrm{str}(a, \mathrm{CompIn}(a, b), 1)$

Поэтому, по аксиомам для равенства можно заменить в первом предложении Ветки 1 все $\mathrm{str}(a, i, 1)$ на $\mathrm{str}(a \cdot u, i, 1)$, а $\mathrm{str}(a, \mathrm{CompIn}(a, b), 1)$ можно заменить на $\mathrm{str}(a \cdot u, \mathrm{CompIn}(a, b), 1)$.

И тогда выяснится, что число $\mathrm{CompIn}(a, b)$ полностью удовлетворяет определению для $\mathrm{CompIn}(a \cdot u, b)$.

Таким образом, доказано:

$0 < \mathrm{CompIn}(a,b) \leq \mathrm{len}(a) \Rightarrow \mathrm{CompIn}(a \cdot u, b) = \mathrm{CompIn}(a,b)$

И по схеме вывода из Раздела 4. Приложение

(α) Если верно $U \Rightarrow \mathrm{B}(a)$, то верно $U \Rightarrow \forall x\, \mathrm{B}(x)$

Получим:

$0 < \mathrm{CompIn}(a,b) \leq \mathrm{len}(a) \Rightarrow \forall u\, \mathrm{CompIn}(a \cdot u, b) = \mathrm{CompIn}(a,b)$, ветка 1 доказана.

Ветка 2.

Возьмём контрапозицию, эквивалентную доказываемой импликации:

$[\mathrm{CompIn}(a,b) = 0 \vee \mathrm{len}(a) < \mathrm{CompIn}(a,b)] \Rightarrow \neg\forall u\, \mathrm{CompIn}(a \cdot u, b) = \mathrm{CompIn}(a,b)$

Правую часть перепишем в соответствии с п. VI.7 Раздела 4. Приложение. И разобьём доказательство полученной импликации на доказательство 2-х импликаций:

Ветка 2.1: $\mathrm{CompIn}(a,b) = 0 \Rightarrow \exists u\, \mathrm{CompIn}(a \cdot u, b) \neq \mathrm{CompIn}(a,b)$

Ветка 2.2: $\mathrm{len}(a) < \mathrm{CompIn}(a,b) \Rightarrow \exists u\, \mathrm{CompIn}(a \cdot u, b) \neq \mathrm{CompIn}(a,b)$

После того, как Ветки 2.1 и 2.2 будут доказаны, из них выводится исходная Ветку 2 в силу тавтологии:

$(A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow ((A \vee B) \Rightarrow C))$, п.III.3 Раздела 4. Приложение.

Ветка 2.1. $\mathrm{CompIn}(a,b) = 0$ – предположение дедукции.

Так как $\mathrm{CompIn}(a,b) = 0$, то из определения (9.1) получаем:

$a = b$

Положим:

$u = \mathrm{Chr}(0)$

Очевидно (легко выводится из $a = b$ и $\mathrm{len}(a) \neq \ominus$):

$\mathrm{CompIn}(a \cdot \mathrm{Chr}(0), b) \neq \mathrm{CompIn}(a,b)$

Так как левая часть неравенства ($\mathrm{CompIn}(a \cdot \mathrm{Chr}(0), b)$) имеет значение $\mathrm{len}(a)'$, а правая ($\mathrm{CompIn}(a,b)$) – имеет значение 0 (ноль).

Из аксиомы (b) Раздела 4. Приложение:

$\mathrm{CompIn}(a \cdot u, b) \neq \mathrm{CompIn}(a,b) \Rightarrow \exists u\, \mathrm{CompIn}(a \cdot u, b) \neq \mathrm{CompIn}(a,b)$

А, значит, верно, при подстановке в свободную (не под квантором) переменную u:

$\mathrm{CompIn}(a \cdot \mathrm{Chr}(0), b) \neq \mathrm{CompIn}(a,b) \Rightarrow \exists u\, \mathrm{CompIn}(a \cdot u, b) \neq \mathrm{CompIn}(a,b)$

Так как посылка импликации истинна, то получаем:

$\exists u\, \mathrm{CompIn}(a \cdot u, b) \neq \mathrm{CompIn}(a,b)$, вспоминая предположение дедукции ветки 2.1 – получаем её доказательство.

Ветка 2.2. $\mathrm{len}(a) < \mathrm{CompIn}(a,b)$ – предположение дедукции.

Так как $\mathrm{len}(a) < \mathrm{CompIn}(a,b)$, то из определения (9.1) получаем:

$\forall i \leq \mathrm{len}(a) : \mathrm{str}(a,i,1) = \mathrm{str}(b,i,1)$

Поэтому, из (т.8.5) получаем

$\mathrm{beg}(b, \mathrm{len}(a)) = a$

Положим:

$u = \mathrm{end}(b, \mathrm{len}(a)')$

Тогда по (1) получаем:

$b = a \cdot \mathrm{end}(b, \mathrm{len}(a)')$

Поэтому:

$\mathrm{CompIn}(a \cdot \mathrm{end}(b, \mathrm{len}(a)'), b) \neq \mathrm{CompIn}(a, b)$

Последнее неравенство истинное, потому что слева – 0 (ноль), а справа $\mathrm{CompIn}(a, b) > 0$.

Дальше вывод аналогичен ветке 2.1, поэтому ветка 2.2. доказана.

Следствие (с.9.1) доказано.

(9.2) Определение для $\mathrm{Comp}(a, b)$:

$\mathrm{Comp}(a, b) = \mathrm{Comp}(\mathrm{str}(a, \mathrm{CompIn}(a, b), 1), \mathrm{str}(b, \mathrm{CompIn}(a, b), 1))$

Очевидно, что это определение.

1. Если строки не равны, то результат $\mathrm{Comp}(a, b)$ совпадает с результатом сравнения элементов алфавита на том месте (в обоих строках), где это отличие возникло впервые от 1 места и до данного, Вопрос сводится к сравнению элементов алфавита в силу аксиомы (5.1) $\forall a \forall i \exists j \ \mathrm{str}(a, i, 1) = \mathrm{Chr}(j)$. А сравнение для элементов алфавита было определено в аксиоме (5.4).

2. Если строки равны, то результат сравнения будет равен нулю, потому что

$\mathrm{CompIn}(a, b) = 0$ для равных строк, в силу чего

$\mathrm{Comp}(\mathrm{str}(a, 0, 1), \mathrm{str}(b, 0, 1)) = \mathrm{Comp}(\ominus, \ominus) = 0$

В последней строке вывода были использованы определение (4.1) $\mathrm{str}(a, i, j) = \mathrm{beg}(\mathrm{end}(a, i), j)$ и аксиома (2.2) $\mathrm{end}(a, 0) = \ominus$, и ещё аксиома (2.4) $\mathrm{beg}(\ominus, i) = \ominus$.

Критерий локального сравнения данных:

(с.9.2) $\mathrm{len}(a) \neq \ominus \Rightarrow [0 < \mathrm{CompIn}(a, b) \leq \mathrm{len}(a) \Leftrightarrow \forall u \, \mathrm{Comp}(a \cdot u, b) = \mathrm{Comp}(a, b)]$, и то же верно, если в этом следствии заменить $\mathrm{CompIn}(a, b)$, $\mathrm{Comp}(a \cdot u, b)$ и $\mathrm{Comp}(a, b)$ на $\mathrm{CompIn}(b, a)$, $\mathrm{Comp}(b, a \cdot u)$ и $\mathrm{Comp}(b, a)$ соответственно.

Заметим, что в левой и правой частях эквивалентностей используются разные функции. В левой - $\mathrm{CompIn}(a, b)$, как в следствии (с.9.1), а в правой - $\mathrm{Comp}(a, b)$, которая определена в последней аксиоме (9.2).

Доказательство.

По аналогии с доказательством (с.9.1) рассмотрим две импликации при гипотезе дедукции $\mathrm{len}(a) \neq \ominus$.

Ветка 1.

Из доказательства Ветки 1 в (с.9.1) знаем, что когда $\mathrm{CompIn}(a, b) \neq 0$, и $\mathrm{CompIn}(a, b) \leq \mathrm{len}(a)$, то

$\mathrm{CompIn}(a \cdot u, b) = \mathrm{CompIn}(a, b)$

И из доказательства (с.9.1) для всех $i < \mathrm{len}(a)$ верно

$\mathrm{str}(a \cdot u, i, 1) = \mathrm{str}(a, i, 1)$

Поэтому из определения (9.2) – заменяя сразу $\mathrm{CompIn}(a \cdot u, b)$ и $\mathrm{str}(a \cdot u, \ldots, 1)$ на равные им $\mathrm{CompIn}(a, b)$ и $\mathrm{str}(a, \ldots, 1)$:

$\mathrm{Comp}(a \cdot u, b) = \mathrm{Comp}(\mathrm{str}(a, \mathrm{CompIn}(a, b), 1), \mathrm{str}(b, \mathrm{CompIn}(a, b), 1)) = \mathrm{Comp}(a, b)$

Поэтому Ветка 1 доказана.

Ветка 2.1: $\mathrm{CompIn}(a, b) = 0 \Rightarrow \exists u \, \mathrm{Comp}(a \cdot u, b) \neq \mathrm{Comp}(a, b)$

Как и в доказательстве Ветки 2.1 следствия (с.9.1) рассматриваем

$u = \mathrm{Chr}(0)$ и выводим заключение импликации из

$\mathrm{Comp}(a{\cdot}\mathrm{Chr}(0), b) = \mathrm{Comp}(\mathrm{str}(a{\cdot}\mathrm{Chr}(0), \mathrm{CompIn}(a{\cdot}\mathrm{Chr}(0), b), 1), \mathrm{str}(b, \mathrm{CompIn}(a{\cdot}\mathrm{Chr}(0), b), 1))$

$= \mathrm{Comp}(\mathrm{str}(a \cdot \mathrm{Chr}(0), \mathrm{len}(a)', 1), \mathrm{str}(b, \mathrm{len}(a)', 1)) = \mathrm{Comp}(\mathrm{Chr}(0), \ominus) = 2$

Результат 2 отличается от $\mathrm{Comp}(a, b) = 0$.

Завершение доказательства – по аналогии с Веткой 2.1 из доказательства для (с.9.1).

Ветка 2.2: $\mathrm{len}(a) < \mathrm{CompIn}(a, b) \Rightarrow \exists u\, \mathrm{Comp}(a \cdot u, b) \neq \mathrm{Comp}(a, b)$

В качестве предположения дедукции-2 возьмем не просто $\mathrm{len}(a) < \mathrm{CompIn}(a, b)$, но:

$\mathrm{CompIn}(a, b) \neq 0 \wedge \mathrm{len}(a) < \mathrm{CompIn}(a, b)$, в соответствии с теоремой VI.10 из раздела 4. Приложение.

Как и в Ветке 2.2 доказательства (с.9.1) рассматриваем

$u = \mathrm{end}(b, \mathrm{len}(a)')$ и выводим

$\mathrm{Comp}(a \cdot u, b) = 0$, так как мы добились равенства $a \cdot u = b$.

И этот результат 0 (ноль) отличается от $\mathrm{Comp}(a, b) \neq 0$ из предположения дедукции-2.

Завершение доказательства – по аналогии с Веткой 2.1

Следствие (с.9.2) доказано.

(л.9.3) Лемма о минимальной позиции совпадения в строке:

$what \neq \ominus \wedge \mathrm{len}(what) \neq \ominus \wedge \mathrm{str}(in, n, \mathrm{len}(what)) = what \Rightarrow$

$\exists_1 m(\mathrm{str}(in, m, \mathrm{len}(what)) = what \wedge (\forall i < m : \mathrm{str}(in, i, \mathrm{len}(what)) \neq what))$

Пояснение. Это лемма о том, что если в строке in имеется подстрока (не пустая и не бесконечная), равная строке $what$, то можно найти такую подстроку, равную $what$, которая начиналась бы в строке in раньше (на позиции с самым маленьким номером), чем любые другие такие строки.

Логика данной формулы идентична логике формулы (л.9.1) – меняется только способ сравнения: вместо поиска самого раннего отличия в данной позиции у данной строки ищется самое раннее совпадение начиная с данной позиции у данной строки. Доказательство аналогичное доказательству (л.9.1).

(9.3) Определение для $\mathrm{find}(in, what, 0)$:

$(what \neq \ominus \wedge \mathrm{len}(what) \neq \ominus \wedge \mathrm{str}(in, \mathrm{find}(in, what, 0), \mathrm{len}(what)) = what$

$\wedge \forall j (j < \mathrm{find}(in, what, 0) \Rightarrow \mathrm{str}(in, j, \mathrm{len}(what)) \neq what)$

$) \vee (\mathrm{find}(in, what, 0) = 0 \wedge (what = \ominus \vee \mathrm{len}(what) = \ominus \vee \forall j\ \mathrm{str}(in, j, \mathrm{len}(what)) \neq what))$

Пояснение.

Первый член дизъюнкции соответствует случаю, когда найдена позиция в строке in, с которой начинается такая подстрока в строке in, которая равна строке $what$. Притом раньше найденной позиции нет никакой другой позиции с таким же совпадением со строкой $what$, а строка $what$ не является ни пустой, ни бесконечной.

Второй член дизъюнкции – это случай равенства нулю: это случай, когда строка $what$ является либо пустой, либо бесконечной, либо ни на каких местах в строке in не начинается подстрока, равная строке $what$.

В принципе, можно было бы включить в состав находимых и варианты с бесконечной подстрокой, но не вижу в этом необходимости с практической точки зрения. Если же необходимость возникнет, то можно будет тогда определить функцию расширенного поиска бесконечной подстроки.

Сейчас же логика функции $\mathrm{find}(in, what, 0)$ почти совпадает с логикой функции $\mathrm{CompIn}(a, b)$. Даже проще, так как случай $what = \ominus$ сразу попадет в «прочие» для $\mathrm{find}(in, what, 0)$, в отличие от $b = \ominus$ для $\mathrm{CompIn}(a, b)$.

Так как логика формулы (9.3) аналогична логике формулы (9.1), как логика леммы (л.9.3) аналогична логике леммы (л.9.1), то:

Доказательство на основе леммы (л.9.3) того, что аксиома (9.3) является определением – проводится почти идентично тому (с учетом разницы в способе признания результата в данной строке в данной позиции – искомым), как доказывалось на основе леммы (л.9.1) то, что аксиома (9.1) является определением.

Поэтому считаем доказанным то, что аксиома (9.3) является определением для
$\mathrm{find}(in, what, 0)$.

(9.4) Определение для $\mathrm{find}(in, what, i)$:
$\mathrm{find}(in, what, i) = \mathrm{find}(\mathrm{end}(in, i'), what, 0)$

Очевидно, что аксиома (9.4) является определением для функции $\mathrm{find}(in, what, i)$.

От функции $\mathrm{find}(in, what, 0)$ она отличается тем, что ищет позицию совпадения со строкой $what$ в строке in не с первой позиции строки in, а только с (i') позиции включительно, пропустив первые i символов в строке in.

(9.5) Определение для $\mathrm{if}_0(a, b, c)$:
$(\mathrm{if}_0(a, b, c) = b \wedge a = 0) \vee (\mathrm{if}_0(a, b, c) = c \wedge a \neq 0)$

Пояснение. Если первый аргумент в данной функции равен нулю, то функция принимает значение второго аргумента. В противном случае функция принимает значение третьего аргумента.

Очевидно, что данная аксиома является определением, то есть, предложение:
$\exists_1 u(u = b \wedge a = 0) \vee (u = c \wedge a \neq 0)$ легко выводится.

(л.9.4) $a \neq a \cdot \mathrm{Chr}(0) \Leftrightarrow \mathrm{len}(a) \neq \ominus$

Действительно, если верно $a \neq a \cdot \mathrm{Chr}(0)$, то для бесконечной строки это не так, в силу применив к аксиоме о конкатенации с бесконечной строкой (7.2) контрапозиции и подстановки:
$a \neq a \cdot \mathrm{Chr}(0) \Rightarrow \mathrm{len}(a) \neq \ominus$

В обратную сторону эта импликация доказывается из (т.7.1) которая (напоминание):
$\mathrm{len}(a_1) \neq \ominus \wedge \ldots \wedge \mathrm{len}(a_n) \neq \ominus \Rightarrow \mathrm{len}(a_1 \cdot \ldots \cdot a_n) = \mathrm{len}(a_1) + \ldots + \mathrm{len}(a_n)$, так как $\mathrm{len}(a) \neq \ominus$:
$\mathrm{len}(a \cdot \mathrm{Chr}(0)) = \mathrm{len}(a) + \mathrm{len}(\mathrm{Chr}(0))$

Но из (т.7.2) которая (напоминание):
$i \leq l_{ChrLim} \Rightarrow \mathrm{len}(\mathrm{Chr}(i)) = 1$

Получаем:
$\mathrm{len}(\mathrm{Chr}(0)) = 1$, и находим:
$\mathrm{len}(a \cdot \mathrm{Chr}(0)) = \mathrm{len}(a)'$

Из арифметики известно – так как $\mathrm{len}(a)$ число – что:

$\mathrm{len}(a)' \neq \mathrm{len}(a)$

откуда

$\mathrm{len}(a \cdot \mathrm{Chr}(0)) \neq \mathrm{len}(a)$

А из теоремы (т.8.5)

$\mathrm{len}(a) \neq \ominus \Rightarrow [a = b \Leftrightarrow \forall i((\mathrm{len}(a) = \mathrm{len}(b) \wedge 0 < i \wedge i \leq \mathrm{len}(a)) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))]$

Получаем поэтому (из $\mathrm{len}(a \cdot \mathrm{Chr}(0)) \neq \mathrm{len}(a)$ и из (т.8.5), меняя обе стороны в эквивалентности на свои отрицания):

$a \neq a \cdot \mathrm{Chr}(0)$

Значит, эквивалентность

$a \neq a \cdot \mathrm{Chr}(0) \Leftrightarrow \mathrm{len}(a) \neq \ominus$

доказана по теореме дедукции и в «обратную» сторону.

Принципиальная необходимость следующих трёх определений будет понятна после трёх подразделов, которые следуют за текущим подразделом.

(9.6) Определение для функции $\mathrm{Ins}(a, i, b)$:

$\mathrm{Ins}(a, i, b) = \mathrm{beg}(a, \max(i, 1) - 1) \cdot b \cdot \mathrm{end}(a, \mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b) + i))$

Данная функция представляет собой композицию аксиоматизированных и определённых ранее или в арифметике функций, поэтому нет нужды в доказательстве того, что (9.6) – определение.

Пояснение. Функция $\mathrm{Ins}(a, i, b)$ «вставляет» строку b внутрь строки a, заменяя своими символами все те символы строки a, номера мест которых в строке a начинаются с i-го (включительно) и заканчиваются $i + \mathrm{len}(b) - 1$-м. По крайней мере, так происходит, когда значение переменной a имеет достаточно большой размер. То есть, когда в строке a на местах номер с i-го по $i + \mathrm{len}(b) - 1$ находятся символы.

Если же размер $\mathrm{len}(a) < i$, то строка b просто допишется (при помощи конкатенации) к концу строки a.

Если же размер $\mathrm{len}(a) \geq i$, но при этом вся строка b не может «влезть» в исходные пределы строки a (потому что $\mathrm{len}(a) < i + \mathrm{len}(b) - 1$), то от строки a останется только $\mathrm{beg}(a, i - 1)$, а дальше будет дописана строка b.

Но всё сказанное о значении функции $\mathrm{Ins}(a, i, b)$ верно лишь в том случае, если $i \neq 0$. Потому что если $i = 0$, то значением функции $\mathrm{Ins}(a, i, b)$ будет b.

Для чего нам нужна эта функция? Для того, чтобы наша теория была способна выразить такую работу алгоритма, при которой алгоритм изменяет лишь часть данных (то, что у нас в теории будет «в начале» переменной). И чтобы получить результат этой работы, алгоритму не требуется ни чтения «лишней» части данных, ни изменения этих «лишних» данных. И, как мы увидим дальше, такая выразительность не может быть достигнута ни рекурсивными функциями, ни в рамках арифметики.

Более формально говоря, нам необходимо сформулировать такие «корректные» условия для «локальной вставки», при которых будет верно:

$\forall u \, \mathrm{Ins}(a \cdot u, i, b) = \mathrm{Ins}(a, i, b) \cdot u$

При этом «хвост» u не должен «сдвинуться» в строке $\mathrm{Ins}(a,i,b)\cdot u$, если сравнивать его положение с положением в строке $a\cdot u$:

$\mathrm{len}(a)=\mathrm{len}(\mathrm{Ins}(a,i,b))$

Теперь мы приступим к реализации всей этой программы.

Определим следующую функцию проверки выполнения комплекса условий для локальной вставки:

(9.7) Определение для функции $\mathrm{IsIns}(q_a,i,q_b)$:

$(\mathrm{IsIns}(q_a,i,q_b)=1\wedge i\neq 0\wedge q_a\neq\ominus\wedge q_b\neq\ominus\wedge(q_b=0\vee q_a\geq i+q_b-1))$

$\vee(\mathrm{IsIns}(q_a,i,q_b)=0\wedge(i=0\vee q_a=\ominus\vee q_b=\ominus\vee(q_b\neq 0\wedge q_a<i+q_b-1)))$

Очевидно, что это определение, потому что первый член дизъюнкции описывает набор всех «корректных» условий (пока «корректность» не доказана – пишу кавычки), а второй член дизъюнкции описывает случай отрицания первого набора. И для «корректного» набора условий есть единственное значение функции – равное 1. А для любого «некорректного» набора тоже единственный результат функции – равный 0. И эти результаты не равны $(1\neq 0)$.

Но не только «вставку» новой строки b в старую строку a можно сделать так, что при любом продолжении старой строки a строкой u не было никаких изменений в строке u ни в содержании, ни в положении строки u внутри $a\cdot u$ после изменения:

$\forall u\,\mathrm{Ins}(a\cdot u,i,b)=\mathrm{Ins}(a,i,b)\cdot u$

Точно так же и получать значения из строки можно независимо от того, началом в какой строке $a\cdot u$ является строка a:

$\forall u\ \mathrm{str}(a\cdot u,i,j)=\mathrm{str}(a,i,j)$

И для вычисления того, когда данное выражение истинно, мы определим соответствующую функцию.

(9.8) Определение для функции $\mathrm{IsStr}(q_a,i,j)$:

$(\mathrm{IsStr}(q_a,i,j)=1\wedge q_a\neq\ominus\wedge(i=0\vee j=0\vee q_a\geq i+j-1))$

$\vee(\mathrm{IsStr}(q_a,i,j)=0\wedge(q_a=\ominus\vee(i\neq 0\wedge j\neq 0\wedge q_a<i+j-1)))$

Очевидно, что это определение, по тем же причинам, что были написаны для определения (9.7).

Пояснение. Разница между извлечением подстроки и вставкой в строку состоит в том, что при вставке с нулевого символа – всё заменяется новым значением. А при извлечении с нулевого места в качестве результата получается пустая строка. И такое значение результата не зависит от того, какая строка u продолжает строку a в строке $a\cdot u$. Поэтому данная функция $\mathrm{IsStr}(q_a,i,j)$ возвращает 1 (в отличие от нуля функции $\mathrm{IsIns}(q_a,i,q_b)$ при аналогичных условиях), если подстроку пытаются получить с нулевого места строки a (или строки $a\cdot u$). А в остальном функции $\mathrm{IsIns}(q_a,i,q_b)$ и $\mathrm{IsStr}(q_a,i,j)$ работают схожим образом.

Только функция $\mathrm{IsIns}(q_a,i,q_b)$ делает дополнительную проверку по аргументу q_b – не соответствует ли его значение строке с бесконечной длиной, а в функции $\mathrm{IsStr}(q_a,i,j)$ изначально используется j для извлечения конечной строки длины j (или меньше, если в строке с длиной q_a не хватит символов).

(9.9) Определение для $\mathrm{ss}(i)$ и $\mathrm{ss_k}(i)$:

$\mathrm{len}(\mathrm{ss}(i)) = i \wedge \forall j \leq i : \mathrm{str}(\mathrm{ss}(i), j, 1) = \mathrm{Chr}(0)$. Функция $\mathrm{ss_k}(i)$ определяется так же, но только вместо $\mathrm{Chr}(0)$ используется $\mathrm{Chr}(k)$, где $k \leq l_{ChrLim}$.

То, что (9.9) является определением – очевидно. Формально существование $\mathrm{ss}(i)$ можно вывести по индукции из $\mathrm{ss}(0) = \ominus$ и $\mathrm{ss}(i') = \mathrm{ss}(i) \cdot \mathrm{Chr}(0)$. А единственность этого существования следует из (т.8.5):

$\mathrm{len}(a) \neq \ominus \Rightarrow [a = b \Leftrightarrow \forall i((\mathrm{len}(a) = \mathrm{len}(b) \wedge 0 < i \wedge i \leq \mathrm{len}(a)) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))]$

VI. Алгоритмы, не сводимые к (частично) рекурсивным функциям и арифметике

За неимением теории строк, теория первого порядка, в которой «представимы» алгоритмы в современных книгах по теории алгоритмов – это арифметика, а представимость алгоритмов в арифметике – это представимость частично рекурсивных функций (ЧРФ далее в этом подразделе). Но множество ЧРФ не включает в себя все необходимые для исследования в теории алгоритмов алгоритмы.

Дело в том, что примитивно рекурсивные функции (ПРФ далее в этом подразделе) используют только такие базовые функции, и такие операторы подстановки и примитивной рекурсии, которые имеют дело только с целостными значениями переменных. Ничего в этом плане не меняет и оператор минимизации – расширяющий множество примитивно рекурсивных функций до множества ЧРФ.

Напомню базовые функции и способы создания из них всех остальных функций данного множества, взятые из Википедии (на начало дня 25.02.2020):

Начало цитаты

К числу базовых ПРФ относятся функции следующих трёх видов:

1. Нулевая функция 0 — функция без аргументов, всегда возвращающая 0.

2. Функция следования S одного переменного, сопоставляющая любому натуральному числу непосредственно следующее за ним натуральное число $x+1$.

3. Функции I_n^m, где $0 < m \leq n$, от n переменных, сопоставляющие любому упорядоченному набору $x_1, \ldots, x_n$ натуральных чисел число x_m из этого набора.

Операторы подстановки и примитивной рекурсии определяются следующим образом:

Оператор суперпозиции (иногда — оператор подстановки). Пусть f — функция от m переменных, а $g_1, \ldots, g_m$ - упорядоченный набор функций от n переменных каждая. Тогда результатом суперпозиции функций g_k в функцию f называется функция h от n переменных, сопоставляющая любому упорядоченному набору $x_1, \ldots, x_n$ натуральных чисел число

$\mathrm{h}(x_1, \ldots, x_n) = \mathrm{f}(\mathrm{g_1}(x_1, \ldots, x_n), \ldots, \mathrm{g_m}(x_1, \ldots, x_n))$.

Оператор примитивной рекурсии. Пусть f — функция от n переменных, а g – функция от $n+2$ переменных. Тогда результатом применения оператора примитивной рекурсии к паре функций f и g называется функция h от $n+1$ переменной вида

$\mathrm{h}(x_1, \ldots, x_n, 0) = \mathrm{f}(x_1, \ldots, x_n)$,

$\mathrm{h}(x_1, \ldots, x_n, y+1) = \mathrm{g}(x_1, \ldots, x_n, y, \mathrm{h}(x_1, \ldots, x_n, y))$.

В данном определении переменную y можно понимать как счётчик итераций, f – как исходную функцию в начале итерационного процесса, выдающего некую последовательность функций n переменных, начинающуюся с f, и g – как оператор, принимающий на вход n переменных $x_1, \ldots, x_n$, номер шага итерации (y), функцию $\mathrm{h}(x_1, \ldots, x_n, y)$ на данном шаге итерации, и возвращающий функцию на следующем шаге итерации.

. . .

ЧРФ определяется аналогично ПРФ, только к двум операторам суперпозиции и примитивной рекурсии добавляется ещё третий оператор – минимизации аргумента.

Оператор минимизации аргумента. Пусть f – функция от n натуральных переменных. Тогда результатом применения оператора минимума аргумента к функции f называется функция h от $n-1$ переменной, задаваемая следующим определением:

$\mathrm{h}(x_1,\ldots,x_{n-1},0) = min\, y$, при условии $\mathrm{f}(x_1,\ldots,n-1,y) = 0$

То есть функция h возвращает минимальное значение последнего аргумента функции f, при котором её значение равно 0.

...

Общерекурсивная функция – частично рекурсивная функция, определённая для всех значений аргументов.

Конец цитаты

Как видим, никакая базовая функция не может работать с какой-то частью своего входного аргумента, не зная его «хвоста». И любая производная от базовых функций функция на базе указанных выше правил – тоже не может оказаться функцией, использующей свой аргумент частично, не «читая» его целиком. Действительно:

Нулевая функция не имеет аргументов, Функции I_n^m использует только готовые аргументы, никак не получая из аргумента часть его значения, функция следования – увеличивает аргумент на 1. Чтобы это сделать – необходимо прочитать аргумент целиком.

Итак, базовые функции работают только, используя (читая, в частности) аргумент целиком.

Теперь разберем методы построения новых функций из базовых функций.

Оператор суперпозиции никак не меняет того, как составляющие его функции работают с аргументами, и сводится к ним в плане работы с аргументами. Оператор примитивной рекурсии – уменьшает аргумент при своем построении на 1. Но для получения значения, которое меньше входного аргумента на 1 надо прочитать входной аргумент целиком.

Таким образом, ПРФ могут работать только с целыми значениями входных аргументов. Потому что они строятся на основе базовых функций, которые работают с целостными значениями входных аргументов. А методы построения ПРФ сохраняют тот же метод работы со значениями входных аргументов – читая и используя эти значения целиком.

Меняет ли что-то в этой ситуации оператор минимизации? В смысле входных аргументов он всего лишь сокращает их число.

Таким образом, ЧРФ работают со значениями входных аргументов только целостно – используя (читая, в частности) все входные данные в каждом используемом ими аргументе. «Используемые» аргументы не следует путать с переданными функции аргументами: функция I_n^m, например, использует только один аргумент, но тоже целиком.

Это значит, что в рамках множества ЧРФ невозможно отличить алгоритм, использующий только первый символ в своём входном аргументе от алгоритма, который работает со всем своим входным аргументом. Поэтому решать вопрос об эффективности (скорости) работы алгоритмов, находясь в рамках ЧРФ невозможно в общем случае:

Ведь разница между размерами одного символа (или любого ограниченного их числа) и возможным размером входного аргумента, не может быть ограничена не то, что каким-либо полиномом

от этого ограниченного числа, но вообще никакой функцией.

Вот в теории строк 1-я же аксиома сообщает о возможности разбиения строки в позиции с произвольным номером i с выделением начальной части $\mathrm{beg}(a,i)$:

(1) $a = \mathrm{beg}(a,i) \cdot \mathrm{end}(a,i')$

Мало того, что строка a в теории строк может быть представлена как

$a = b \cdot Unknown$

Но и работать со строкой $a = b \cdot Unknown$ в теории строк можно – при определенных условиях – так, словно вместо всей строки a есть только её начало b. Эти условия видны в следствии (с.9.2) для работы со строкой при помощи функции $\mathrm{Comp}()$, эти условия представлены функциями $\mathrm{IsIns}(q_a,i,q_b)$ и $\mathrm{IsStr}(q_a,i,j)$, как будет доказано в следующем подразделе.

Но помимо теории строк – ещё и в реальном мире у нас есть возможность «локальной» работы со строкой. Например: я могу прочитать только начало книги и решить, что она мне не годится. И для принятия такого решения мне нет нужды читать книгу до конца, потому что начало книги несёт такую информацию, которая никак не меняется от дальнейшего текста. И время на чтение этого начала я трачу одинаково вне зависимости от того – целая книга у меня в руках, или только вырванный из книги блок её первых страниц.

Но для того, чтобы в теории я мог воспользоваться возможностями «локальной работы» со строками из реального мира, мне нужно иметь в теории соответствующие инструменты. А таких инструментов нет в формализме частично рекурсивных функций.

Проблема ЧРФ при работе с входными данными чрезмерно (сколь угодно «чрезмерно») большого размера, которые не используются алгоритмом в своей работе, кроме некоторой начальной части этих данных, порождает формулировки вроде:

«Подтверждающий сертификат должен иметь размер, ограниченный полиномом от размера проверяемого слова» и «Если среди сертификатов допустимого размера нет сертификата, который является подтверждающим для данного слова, то данное слово не принадлежит языку».

Возникает вопрос: а кто/что проверяет – является ли допустимым размер у проверяемого сертификата? Разве проверка такого рода не является вычислением? И разве время на эту проверку не столь же уместно учитывать в качестве времени работы алгоритма проверки, как и время после того, когда недопустимые по своим размерам сертификаты отброшены?

Разумеется, время на проверку размеров сертификатов учитывать надо. И правильная формулировка состоит в том, что время работы алгоритма проверки не должно зависеть от размера чрезмерного по своим размерам сертификата. Но контроль размера и выбраковку сертификатов с чрезмерно большими размерами алгоритм проверки осуществляет самостоятельно.

И выбраковка чрезмерно больших сертификатов должна осуществляться за время, полиномиальное от критического («граничного») размера сертификата – при чтении первого символа, находящегося вне допустимых – для размера входных данных – пределов. А вовсе не в результате обработки всего «входа».

Но в рамках ЧРФ решить подобную задачу невозможно. Однако одной заменой ЧРФ на что-то более выразительное для теории алгоритмов обойтись не удастся, как мы увидим чуть ниже в

данном подразделе – арифметика тоже не обладает необходимой для теории алгоритмов выразительностью.

Поэтому необходимо создать такую теорию (теорию строк, например) и такую модель исполнения алгоритмов (2-я статья, которая будет после данной), которая устраняла бы неразрешимую в рамках ЧРФ проблему работы с «входными данными», которые используются алгоритмами лишь в незначительной начальной части этих данных.

К данному подразделу мы уже в достаточной степени построили теорию строк для того, чтобы доказать, что арифметика (не только теория Пеано, но и её любые возможные арифметические расширения) недостаточно выразительна для решения вопросов теории алгоритмов в общем случае.

(м.9.1) В теории строк с односимвольным алфавитом ($l_{ChrLim} = 0$) невозможно в общем случае представить работу алгоритмов со строками многосимвольного алфавита таким образом, чтобы размер логического вывода для результата работы такого алгоритма был в полиномиальных (или любых иных) пределах относительно времени работы этого алгоритма.

Доказательство.

Пусть у нас есть алгоритм проверки русского правописания, который должен отвергать (возвращая в качестве результата своей работы ноль, например) предложенный для проверки текст сразу, как только обнаружит ошибку в этом тексте. Пусть на вход данному алгоритму подан роман, первое же слово в котором начинается на «Ъ».

Разумеется, наш алгоритм отвергнет данный роман как неграмотно написанный по первой же его букве. Потому что в русском языке нет слов, которые начинаются на «Ъ». Замечу, что тут речь идёт о работе алгоритма со строкой, составленной на базе многосимвольного алфавита. И для простоты будем считать, что на любую другую букву русское слово начинаться может, что проще реального русского языка, где есть ещё буква «Ь».

Вопрос: можно ли в односимвольной теории строк построить такое представление для данного алгоритма проверки (сопоставив как-то многосимвольным строкам строки из символов односимвольного алфавита), что логический вывод данного результата (доказательство истинности результата, равного нулю) имел бы размер, не выходящий за некие полиномиальные ограничения относительно времени работы данного алгоритма?

Если бы это было так, то в логическом выводе не мог бы присутствовать весь роман, вообще говоря. Потому что размер романа может быть сколь угодно велик по сравнению с «Ъ» и временем работы алгоритма проверки.

Если же предположить, что в логическом выводе теории строк с односимвольным алфавитом присутствует лишь часть строки, то эта часть состоит из одних и тех же символов – будем считать «0», например. А размер всей строки, которая соответствует роману – неизвестен.

Допустим, логический вывод дает нужный нам результат 0 (отвергая роман как ошибочный) на основании подстроки «000...0» длиной в фиксированное количество n символов «0». Количество символов, представляющих весь роман – неизвестно. Это значит, что про роман, который представлен строкой, в которой есть фиксированное число n символов «0» заранее известно, что он

начинается на «Ъ». Я тут упрощаю (это напоминание), потому что при реальном русском языке ошибка будет и при «Ь», но разбираемая модель понятна.

А это означает, что все романы, которые представлены строками односимвольного алфавита длиной от n символов начинаются на символ «Ъ». А это означает, что все романы, которые не начинаются на символ «Ъ» должны соответствовать конечному числу вариантов, которые могут выразить строки (из символа односимвольного алфавита) длиной меньше, чем фиксированное число n символов «0».

Но таких романов – бесконечное количество, а вариантов разных строк, длина которых меньше n – конечное количество. Поэтому случай из предыдущего абзаца невозможен.

Полученное противоречие доказывает, что предположение о том, что:

В односимвольной теории строк логический вывод о том, что многосимвольный текст начинается на некий «запрещённый» символ может обойтись лишь частью строки, которая содержится в строке односимвольного алфавита, соответствующей данному тексту многосимвольного алфавита –

Является ошибочным.

Поскольку размер текста в сравнении с размером одного символа может быть как угодно велик, то мы доказали, что размер логического вывода и время на его проверку в односимвольной теории строк может быть сколь угодно велики в сравнении со временем работы алгоритма проверки. Метатеорема доказана.

(м.9.2) В арифметике невозможно в общем случае представить работу алгоритмов со строками многосимвольного алфавита таким образом, чтобы размер логического вывода для результата работы такого алгоритма был в полиномиальных (или любых иных) пределах относительно времени работы этого алгоритма.

Доказательство.

Выразительность арифметики не превосходит выразительности теории строк с односимвольным алфавитом, потому что арифметика является частью теории строк. Поэтому то, что невозможно логически вывести в теории строк с односимвольным алфавитом – невозможно вывести и в арифметике. Теорема доказана.

Как видим, благодаря теории строк удаётся понять недостаточную выразительность арифметики для того, чтобы пытаться решать в её рамках вопросы теории алгоритмов. И это понимание возникает за счёт «взгляда со стороны» на саму теорию строк, а уже поняв разницу между разными её вариантами – удаётся понять и в чём недостаточная выразительность арифметики для теории алгоритмов. И, разумеется, приведённые доказательства метатеорем не являются формальными и не могут быть формализованы в рамках теории строк – потому что это информация не теории строк, а про теорию строк.

Из метатеоремы (м.9.2) следует, что гёделевы номера – которые определяются в арифметике в качестве способа представить тексты логических формул, логических выводов, функций арифметики и т.п. – являются недостаточно выразительными для задач теории алгоритмов, как и сама арифметика.

А почему нельзя считать нужным представлением формулу, выражающую то, что нужная строка начинается на «Ъ»?

Просто потому, что мы в реальности не можем сказать – есть ли «Ъ» в многосимвольной строке, соответствующей данной «односимвольный» строке – не прочитав данную строку односимвольного алфавита до конца. А раз это невозможно в реальности, то невозможно и в теории – если эта теория непротиворечива и, если соответствующие ей функции мы строим в нашей реальности в возможных в этой реальности интерпретациях.

Но в любом случае мы опираемся на реальность в доказательстве метатеорем. В частности – не можем себе представить, как можно было бы решить вопрос с «Ъ» в начале, не прочитав всю строку из символа односимвольного алфавита.

В следующем разделе мы разберем связь между теорией строк и теорией Пеано (или любым расширением теории Пеано). И там будут представлен и вариант аксиоматизации для «односимвольной» связи строк и числе.

Для арифметики, получается, совершенно закономерно иметь интерпретацию, в которой необходимо читать число до конца. Потому что то, как «зашифровано» число – это никак не задано в самой арифметике. И поэтому ЧРФ никак не «портят» арифметику тем, что работают с входным аргументом только целиком – потому что арифметика «поступает» точно так же.

А в чём разница с многосимвольным алфавитом? В том, что в любой строке данной длины n из символов многосимвольного алфавита есть ещё варианты, кроме одного данного. И «клинча» - «тогда и остальные такие же» - не возникает. А вот при односимвольном алфавите логический вывод, сделанный на основании части длинной n от всей строки (произвольного размера), будет «таким же» для любых строк, имеющих длину больше или равную n.

Кстати, даже «частичное чтение» не позволило бы в арифметике сделать вывод, что число больше или равно числу прочитанных «символов». Действительно, ведь на базе арифметики можно аксиоматизировать следующую связь между числами и строками при односимвольном алфавите:

Если n – чётное число, но не степень 2, то число записано в виде строки из символов «0» длиной $n+1$: «00...0». Если n – нечётное число, то число n записано как строка из символов «0»: «00...0». И количество символов «0» в этой строке (длина строки) будет 2^n+1. А вот число 2^n будет записана тоже как строка из «0», но длина строки в этой записи будет $n+1$.

И тогда нет возможности в общем случае по чтению нескольких начальных символов утверждать, что число больше некоторого числа – в общем случае. Потому что оно может быть равно нечётному числу, хотя символов ты уже прочитал больше, чем данное нечётное число, но прочитал их пока меньше, чем до конца.

Заметим, что в реальности строки обладают свойством доступности для частичного чтения. Теперь недостаточная выразительность арифметики стала обоснованной в результате доказательства метатеоремы (м.9.2).

Уже упоминалось (в подразделе IV раздела 1), что вычислимость на машине Тьюринга в современных учебниках обычно рассматривается в духе стандартной интерпретации, то есть, когда «входные аргументы» представлены последовательностью «счётных палочек», что экспоненциаль-

но больше по своим размерам, чем позиционное представление чисел. И представимость алгоритмов машины Тьюринга в арифметике доказывается не напрямую, а через ЧРФ.

Тезис Чёрча понимался сейчас обычно как возможность свести произвольное вычисление (которое завершается за конечное время) к некоторой ЧРФ, чтобы в итоге вычисления получить правильный результат – независимо от того времени, которое потребуется для этого – если время конечно. И независимо от размера логического вывода для доказательства того, что будет получен правильный результат – если размер логического вывода конечный.

И тезис Чёрча оказывается ошибочным, если понимать его как возможность свести произвольное вычисление к некоторой ЧРФ, без потери важных для теории алгоритмов свойств вычисления.

Что же теперь будет построено вместо ЧРФ? В планах – в статье номер 2 – будет подробно рассмотрена «машина компьютерных алгоритмов» (МКА далее), и в ней будет реализована возможность частичного чтения входного аргумента. И представимость функций в теории строк будет доказываться уже непосредственно в отношении МКА, без «посредника» вроде рекурсивных функций. При этом речь должна идти о полиномиальной представимости – чтобы время работы алгоритма и размер текста доказательства были в полиномиальных пределах друг относительно друга.

Можно ли вместо МКА взять машину Тьюринга? Возможно – но это весьма неудобно. И если в рамках прежнего построения «представимости» сопоставление между работой машины Тьюринга и теорией Пеано шло через ЧРФ и касалось только ЧРФ, то заменяя ЧРФ на МКА, мы только упростим ситуацию. К тому же будет сделано необходимое исправление – замена модели ЧРФ, обладающей приматом целостности (даже если это словарные ЧРФ, а не числовые) на модель МКА – которая свободна от этого ограничения.

VII. Критерий локального изменения данных

Теперь мы сформулируем и докажем теоремы, которые нам потребуются для доказательства «полиномиальной представимости» машины компьютерных алгоритмов в теории строк. Теория строк излагается в данной (1-й) статье, машина компьютерных алгоритмов строится в следующей (2-й) статье. Доказательство «полиномиальной представимости» будет дано в 3-й статье, а некоторые нюансы (оговоренные в 3-й статье) будут уточнены в 4-й статье.

(т.9.1) $\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus \Rightarrow \mathrm{str}(a \cdot b \cdot u, \mathrm{len}(a)', \mathrm{len}(b)) = b$

Доказательство при дедуктивном предположении $\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus$.

Ветка 1. Дедуктивное предположение-2: $\mathrm{len}(b) \neq 0$

Тогда из-за (т.4.2) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{str}(a, i, j) = \mathrm{end}(\mathrm{beg}(a, i + j - 1), i)$ получим:

1.1. $\mathrm{str}(a \cdot b \cdot u, \mathrm{len}(a)', \mathrm{len}(b)) = \mathrm{end}(\mathrm{beg}(a \cdot b \cdot u, \mathrm{len}(a) + \mathrm{len}(b)), \mathrm{len}(a)')$

Так как $\mathrm{len}(a \cdot b) = \mathrm{len}(a) + \mathrm{len}(b)$ из-за (т.7.1), и в силу (т.8.6) $\mathrm{len}(a) \neq \ominus \Rightarrow a = \mathrm{beg}(a \cdot u, \mathrm{len}(a))$ получаем:

$\mathrm{beg}((a \cdot b) \cdot u, \mathrm{len}(a) + \mathrm{len}(b)) = (a \cdot b)$

Поэтому из п.1.1 и предыдущего равенства получим:

1.2. $\mathrm{str}(a \cdot b \cdot u, \mathrm{len}(a)', \mathrm{len}(b)) = \mathrm{end}((a \cdot b), \mathrm{len}(a)')$

Но $\mathrm{end}((a \cdot b), \mathrm{len}(a)') = b$, из-за (т.8.7) $\mathrm{len}(a) \neq \ominus \Rightarrow u = \mathrm{end}(a \cdot u, \mathrm{len}(a)')$, поэтому

1.3. $\mathrm{str}(a \cdot b \cdot u, \mathrm{len}(a)', \mathrm{len}(b)) = b$

Ветка 1 доказана.

Ветка 2. Дедуктивное предположение-2: $\mathrm{len}(b) = 0$

Тогда в силу определения $\mathrm{str}()$ и из-за (2.1) $\mathrm{beg}(a, 0) = \ominus$ получим:

2.1. $\mathrm{str}(a \cdot b \cdot u, \mathrm{len}(a)', \mathrm{len}(b)) = \mathrm{str}(a \cdot b \cdot u, \mathrm{len}(a)', 0) = \ominus$

С другой стороны, из-за (т.8.8) $a = \ominus \Leftrightarrow \mathrm{len}(a) = 0$ получим:

2.2. $b = \ominus$

Из пунктов 2.1 и 2.2 получаем:

$\mathrm{str}(a \cdot b \cdot u, \mathrm{len}(a)', \mathrm{len}(b)) = b$

Ветка 2 для случая $\mathrm{len}(b) = 0$ тоже доказана. Вместе с веткой 1 для случая $\mathrm{len}(b) \neq 0$, все варианты рассмотрены.

Теорема (т.9.1) доказана.

(т.9.2) $\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{end}(a \cdot b \cdot u, \mathrm{len}(a)') = b \cdot u$

Доказательство при дедуктивном предположении $\mathrm{len}(a) \neq \ominus$.

Из (т.8.7) $\mathrm{len}(a) \neq \ominus \Rightarrow u = \mathrm{end}(a \cdot u, \mathrm{len}(a)')$ получим, используя ассоциативность конкатенации:

$\mathrm{end}(a \cdot (b \cdot u), \mathrm{len}(a)') = b \cdot u$

Теорема (т.9.2) доказана.

(т.9.3) $\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1 \Rightarrow \mathrm{len}(\mathrm{Ins}(a, i, b)) = \mathrm{len}(a)$

Доказательство при дедуктивном предположении $\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1$.

Из определения (9.7) и дедуктивного предположения получаем:

$i \neq 0 \wedge \mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus \wedge (\mathrm{len}(b) = 0 \vee \mathrm{len}(a) \geq i + \mathrm{len}(b) - 1)$

Теперь просто вычислим, чему равна функция $\mathrm{Ins}(a,i,b)$ при истинности предыдущего утверждения:

$\mathrm{Ins}(a,i,b) = \mathrm{beg}(a, \max(i,1)-1) \cdot b \cdot \mathrm{end}(a, \mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b)+i))$, по определению (9.6)

В силу дедуктивного предположения $(\max(i,1)-1) = i-1$ и в силу ещё и того, что $\mathrm{len}(b) \neq \ominus$ (а, значит, $\mathrm{len}(\mathrm{len}(b)) \neq 0$ из-за (с.8.8)) верно $\min(\mathrm{len}(\mathrm{len}(b)), i) > 0$. Из последнего неравенства и определения if_0 получаем $\mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b)+i) = \mathrm{len}(b)+i$. Поэтому:

$\mathrm{Ins}(a,i,b) = \mathrm{beg}(a, i-1) \cdot b \cdot \mathrm{end}(a, \mathrm{len}(b)+i)$

Если $\mathrm{len}(b) = 0$, то $b = \ominus$ из-за (т.8.8) и тогда его можно исключить из конкатенации в силу её ассоциативности и (т.8.3) $a = a \cdot \ominus = \ominus \cdot a$. Поэтому тогда:

$\mathrm{Ins}(a,i,b) = \mathrm{beg}(a, i-1) \cdot \ominus \cdot \mathrm{end}(a, 0+i) = \mathrm{beg}(a, i-1) \cdot \mathrm{end}(a, i)$

Но по (1) имеем:

$a = \mathrm{beg}(a, i-1) \cdot \mathrm{end}(a, i)$

То есть, при $\mathrm{len}(b) = 0$ верно:

$\mathrm{Ins}(a,i,b) = a$

$\mathrm{len}(\mathrm{Ins}(a,i,b))) = \mathrm{len}(a)$

Теперь рассмотрим $\mathrm{len}(b) \neq 0$, тогда из дедуктивного предположения верно:

$\mathrm{len}(a) \geq i + \mathrm{len}(b) - 1$

Чтобы посчитать $\mathrm{len}(\mathrm{Ins}(a,i,b))$, будем считать длины каждого члена конкатенации в равенстве:

$\mathrm{Ins}(a,i,b) = \mathrm{beg}(a, i-1) \cdot b \cdot \mathrm{end}(a, \mathrm{len}(b)+i)$

А что их сумма будет равна длине их конкатенации обеспечено теоремой (т.7.1) которая (напоминание):

$\mathrm{len}(a_1) \neq \ominus \wedge \ldots \wedge \mathrm{len}(a_n) \neq \ominus \Rightarrow \mathrm{len}(a_1 \cdot \ldots \cdot a_n) = \mathrm{len}(a_1) + \ldots + \mathrm{len}(a_n)$

Поэтому:

$\mathrm{len}(\mathrm{Ins}(a,i,b)) = \mathrm{len}(\mathrm{beg}(a, i-1)) + \mathrm{len}(b) + \mathrm{len}(\mathrm{end}(a, \mathrm{len}(b)+i))$

Из-за теоремы (т.6.7) которая (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge i \leq \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{beg}(a,i)) = i \wedge (\mathrm{len}(\mathrm{end}(a,i)) = \mathrm{len}(a) - i + 1 \vee i = 0)$

Получаем (так как $(i-1) \leq \mathrm{len}(a)$):

$\mathrm{len}(\mathrm{beg}(a, i-1)) = i-1$

Слагаемое $\mathrm{len}(b)$ не требует вычисления, осталось найти $\mathrm{len}(\mathrm{end}(a, \mathrm{len}(b)+i)$

Случай 1.

Для вычисления данного слагаемого нам может пригодится либо упомянутая теорем (6.7) в части:

$\mathrm{len}(\mathrm{end}(a,i)) = \mathrm{len}(a) - i + 1$, но только если $i \leq \mathrm{len}(a)$ в теореме и $\mathrm{len}(b) + i \leq \mathrm{len}(a)$ в нашем конкретном случае. Тогда:

$\mathrm{len}(\mathrm{end}(a, \mathrm{len}(b)+i) = \mathrm{len}(a) - (\mathrm{len}(b)+i) + 1 = \mathrm{len}(a) - \mathrm{len}(b) - i + 1$

$\mathrm{len}(\mathrm{Ins}(a,i,b)) = (i-1) + \mathrm{len}(b) + (\mathrm{len}(a) - \mathrm{len}(b) - i + 1) = \mathrm{len}(a)$

Для Случая 1 теорема доказана.

Случай 2.

Либо для вычисления $\mathrm{len}(\mathrm{end}(a, \mathrm{len}(b)+i)$ нам пригодится теорема (6.9) которая (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(a) < i \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = \mathrm{len}(a) \wedge (\mathrm{len}(\mathrm{end}(a, i)) = 0)$

Нас интересует следующая часть теоремы (6.9):

$\mathrm{len}(\mathrm{end}(a, i)) = 0$, но только если $\mathrm{len}(a) < i$ в теореме и $\mathrm{len}(a) < \mathrm{len}(b) + i$ в нашем конкретном случае. Тогда вместе с условием $\mathrm{len}(a) \geq i + \mathrm{len}(b) - 1$ из дедуктивного предположения имеем:

$\mathrm{len}(a) \geq i + \mathrm{len}(b) - 1 \wedge \mathrm{len}(a) < \mathrm{len}(b) + i$

Перепишем для большей наглядности так:

$\mathrm{len}(b) + i - 1 \leq \mathrm{len}(a) < \mathrm{len}(b) + i$

В арифметике из этого выводится:

$\mathrm{len}(a) = \mathrm{len}(b) + i - 1$

$\mathrm{len}(b) = \mathrm{len}(a) - i + 1$

Теперь можем посчитать всю сумму длин:

$\mathrm{len}(\mathrm{Ins}(a, i, b)) = (i - 1) + \mathrm{len}(b) + 0 = i - 1 + \mathrm{len}(a) - i + 1 = \mathrm{len}(a)$

Для Случая 2 теорема тоже доказана.

Теорема доказана.

(т.9.4) Критерий локальной вставки:

$(\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus) \Rightarrow$

$(\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1 \Leftrightarrow \forall u\, \mathrm{Ins}(a \cdot u, i, b) = \mathrm{Ins}(a, i, b) \cdot u)$

Для доказательства этой эквивалентности нужно доказать, что в условиях 1-го дедуктивного предположения:

$\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus$ истинны 2 импликации:

$\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1 \Rightarrow \forall u\, \mathrm{Ins}(a \cdot u, i, b) = \mathrm{Ins}(a, i, b) \cdot u)$ – первая импликация;

$\forall u\, \mathrm{Ins}(a \cdot u, i, b) = \mathrm{Ins}(a, i, b) \cdot u) \Rightarrow \mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1$ – вторая импликация.

Для доказательства 1-й импликации делаем 2-е дедуктивное предположение:

$\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1$

Из определения (9.7) и 2-го дедуктивного предположения получаем:

$i \neq 0 \wedge \mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus \wedge (\mathrm{len}(b) = 0 \vee \mathrm{len}(a) \geq i + \mathrm{len}(b) - 1)$

Теперь просто вычислим, чему равна функция $\mathrm{Ins}(a \cdot u, i, b)$ из заключения 1-й импликации при истинности предыдущего утверждения:

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{beg}(a \cdot u, \max(i, 1) - 1) \cdot b \cdot \mathrm{end}(a \cdot u, \mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b) + i))$, по определению (9.6)

В силу 2-го дедуктивного предположения $\max(i, 1) - 1 = i - 1$ и в силу ещё и того, что $\mathrm{len}(b) \neq \ominus$ (из 1-го дедуктивного предположения) верно $\min(\mathrm{len}(\mathrm{len}(b)), i) > 0$ (так как $\mathrm{len}(b) \neq \ominus \Leftrightarrow \mathrm{len}(\mathrm{len}(b)) \neq 0$ из-за (с.8.8)) Из последнего неравенства и определения if_0 получаем $\mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, i) = \mathrm{len}(b) + i$. Поэтому:

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{beg}(a \cdot u, i - 1) \cdot b \cdot \mathrm{end}(a \cdot u, \mathrm{len}(b) + i)$, но:

$\mathrm{beg}(a \cdot u, i - 1) = \mathrm{str}(a \cdot u, 1, i - 1)$, из-за (т.4.6),

Поэтому:

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{str}(a \cdot u, 1, i - 1) \cdot b \cdot \mathrm{end}(a \cdot u, \mathrm{len}(b) + i)$

Ветка 1 для 1-й импликации. Случай $\mathrm{len}(b) = 0$.

Если $\mathrm{len}(b) = 0$, то $b = \ominus$ из-за (т.8.8) и тогда его можно исключить из конкатенации в силу её ассоциативности и (т.8.3) $a = a \cdot \ominus = \ominus \cdot a$. Поэтому тогда:

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{str}(a \cdot u, 1, i-1) \cdot \mathrm{end}(a \cdot u, 0 + i)$

Так $\mathrm{str}(a \cdot u, 1, i-1) = \mathrm{beg}(a \cdot u, i-1)$ из-за (т.4.6) $\mathrm{str}(a, 1, j) = \mathrm{beg}(a, j)$, то

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{beg}(a \cdot u, i-1) \cdot \mathrm{end}(a \cdot u, i)$

В силу (1) поэтому при $\mathrm{len}(b) = 0$ получаем:

$\mathrm{Ins}(a \cdot u, i, b) = a \cdot u$

Так как

$\mathrm{Ins}(a, i, b) = \mathrm{Ins}(a \cdot \ominus, i, b) = a \cdot \ominus = a$

То

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{Ins}(a, i, b) \cdot u$

Для ветки 1 импликация 1 доказана.

Ветка 2 для 1-й импликации. Случай $\mathrm{len}(b) > 0$. Из этого и из 2-го дедуктивного предположения получается $\mathrm{len}(a) \geq i + \mathrm{len}(b) - 1$.

У нас доказано при 1-м и 2-м предположениях дедукции

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{str}(a \cdot u, 1, i-1) \cdot b \cdot \mathrm{end}(a \cdot u, \mathrm{len}(b) + i)$

Но $i - 1 \leq \mathrm{len}(a)$, так как выше мы видели, что $\mathrm{len}(b) + i - 1 \leq \mathrm{len}(a)$. Поэтому:

$\mathrm{str}(a \cdot u, 1, i-1) = \mathrm{str}(a, 1, i-1)$, что выводится из (т.8.6) $\mathrm{len}(a) \neq \ominus \Rightarrow a = \mathrm{beg}(a \cdot u, \mathrm{len}(a))$, после того, как разбить a в соответствии с (1) $a = \mathrm{beg}(a, i-1) \cdot \mathrm{end}(a, i)$ и воспользоваться теоремой (т.6.7) чтобы доказать $\mathrm{len}(\mathrm{beg}(a, i-1)) = i - 1$, подобно тому, как мы делали чуть выше.

Таким образом:

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{str}(a, 1, i-1) \cdot b \cdot \mathrm{end}(a \cdot u, \mathrm{len}(b) + i)$

У нас остался только один член конкатенации, зависящий от u. Поэтому теперь надо доказать, что:

$\mathrm{end}(a \cdot u, \mathrm{len}(b) + i) = \mathrm{end}(a, \mathrm{len}(b) + i) \cdot u$

Запишем на основании аксиомы (1) переменную a в виде:

$a = \mathrm{beg}(a, \mathrm{len}(b) + i - 1) \cdot \mathrm{end}(a, \mathrm{len}(b) + i)$

Тогда

$\mathrm{end}(a \cdot u, \mathrm{len}(b) + i) = \mathrm{end}(\mathrm{beg}(a, \mathrm{len}(b) + i - 1) \cdot \mathrm{end}(a, \mathrm{len}(b) + i) \cdot u, \mathrm{len}(b) + i)$

В силу 2-го дедуктивного и предположения Ветки 2 верно $\mathrm{len}(a) \geq i + \mathrm{len}(b) - 1$, поэтому

$\mathrm{len}(b) + i - 1 \leq \mathrm{len}(a)$

Поэтому в силу (т.6.7) которая (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge i \leq \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = i \wedge (\mathrm{len}(\mathrm{end}(a, i)) = \mathrm{len}(a) - i + 1 \vee i = 0)$

Имеем:

$\mathrm{len}(\mathrm{beg}(a, \mathrm{len}(b) + i - 1)) = \mathrm{len}(b) + i - 1$

А поэтому из теоремы (т.9.2) которая (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus \Rightarrow \mathrm{end}(a \cdot b \cdot u, \mathrm{len}(a)') = b \cdot u$

Получим

$\mathrm{end}(\mathrm{beg}(a, \mathrm{len}(b) + i - 1) \cdot \mathrm{end}(a, \mathrm{len}(b) + i) \cdot u, \mathrm{len}(b) + i) = \mathrm{end}(a, \mathrm{len}(b) + i) \cdot u$

Таким образом, мы получили:

$\mathrm{end}(a \cdot u, \mathrm{len}(b) + i) = \mathrm{end}(a, \mathrm{len}(b) + i) \cdot u$

И теперь мы можем переписать предложение

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{str}(a, 1, i - 1) \cdot b \cdot \mathrm{end}(a \cdot u, \mathrm{len}(b) + i)$

Таким образом:

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{str}(a, 1, i - 1) \cdot b \cdot \mathrm{end}(a, \mathrm{len}(b) + i) \cdot u$

Если рассмотреть случай $u = \ominus$, то получим:

$\mathrm{Ins}(a, i, b) = \mathrm{Ins}(a \cdot \ominus, i, b) = \mathrm{str}(a, 1, i - 1) \cdot b \cdot \mathrm{end}(a, \mathrm{len}(b) + i)$

Из последних 2 формул получаем:

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{Ins}(a, i, b) \cdot u$

Ветка 2 для импликации 1 доказана.

Чтобы перейти к квантору общности, надо для результата этих 2х веток (сделав из каждой импликацию по теореме дедукции, соединив обе эти импликации в одну и отбросив дизъюнкцию частных посылок по MP) воспользоваться правилом вывода:

(α) Если верно $U \Rightarrow \mathrm{B}(a)$, то верно $U \Rightarrow \forall x\, \mathrm{B}(x)$

После чего импликация 1 доказана.

Теперь надо доказать в рамках $\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus$ 1-го дедуктивного предположения 2-ю импликацию:

$\forall u\, \mathrm{Ins}(a \cdot u, i, b) = \mathrm{Ins}(a, i, b) \cdot u \Rightarrow \mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1$

Перепишем её в эквивалентном виде при помощи контрапозиции, превращая квантор всеобщности в квантор существования при помощи теоремы VI.7 из раздела 4. Приложение:

$\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) \neq 1 \Rightarrow \exists u\, \mathrm{Ins}(a \cdot u, i, b) \neq \mathrm{Ins}(a, i, b) \cdot u)$

Очевидно, что

$\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) \neq 1 \Leftrightarrow \mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 0$

Поэтому перепишем 2ю импликацию так:

$\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 0 \Rightarrow \exists u\, \mathrm{Ins}(a \cdot u, i, b) \neq \mathrm{Ins}(a, i, b) \cdot u)$

Для доказательства 2-й импликации делаем 2-е дедуктивное предположение:

$\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 0$

Из определения (9.7) и 2-го дедуктивного предположения получаем «рабочий вариант» 2-го дедуктивного предположения:

$i = 0 \vee \mathrm{len}(a) = \ominus \vee \mathrm{len}(b) = \ominus \vee (\mathrm{len}(b) \neq 0 \wedge \mathrm{len}(a) < i + \mathrm{len}(b) - 1)$

Теперь нам нужно доказать, что из каждого члена дизъюнкции приведённого рабочего варианта 2-го дизъюнктивного предположения следует нужное нам заключение

$\exists u\, \mathrm{Ins}(a \cdot u, i, b) \neq \mathrm{Ins}(a, i, b) \cdot u)$

А после этого мы сможем объединить все импликации с разными посылками и одинаковым заключением в импликацию, где посылка будет дизъюнкцией при помощи

III.3 $(A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \vee B \Rightarrow C))$, из раздела 4. Приложение.

Поэтому доказываем последовательно для каждой такой посылки нужное нам заключение

Ветка 1. $i = 0$

Положим

$u = \mathrm{Chr}(0)$, тогда из определения (9.6)

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{beg}(a \cdot u, \max(i, 1) - 1) \cdot b \cdot \mathrm{end}(a \cdot u, \mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b) + i))$

Подстановкой соответствующего $i = 0$ получим для двух членов конкатенации:

$\mathrm{beg}(a \cdot u, \max(i, 1) - 1) = \mathrm{beg}(a \cdot u, \max(0, 1) - 1) = \mathrm{beg}(a \cdot u, 0) = \ominus$

$\mathrm{end}(a \cdot u, \mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b) + i)) = \mathrm{end}(a \cdot u, \mathrm{if}_0(0, 0, \mathrm{len}(b) + i)) = \ominus$

Поэтому:

$\mathrm{Ins}(a \cdot u, i, b) = \ominus \cdot b \cdot \ominus = b$

Что касается $\mathrm{Ins}(a, i, b) \cdot u$, то по аналогичному выводу получим (подставляя $u = \mathrm{Chr}(0)$):

$\mathrm{Ins}(a, i, b) \cdot u = b \cdot u = b \cdot \mathrm{Chr}(0)$

Очевидно, что длины строк, получающиеся из 2-х последних формул не равны:

$\mathrm{len}(\mathrm{Ins}(a \cdot u, i, b)) \neq \mathrm{len}(\mathrm{Ins}(a, i, b) \cdot u)$

А не равны они из-за теоремы (т.7.1) которая (напоминание):

$\mathrm{len}(a_1) \neq \ominus \wedge \ldots \wedge \mathrm{len}(a_n) \neq \ominus \Rightarrow \mathrm{len}(a_1 \cdot \ldots \cdot a_n) = \mathrm{len}(a_1) + \ldots + \mathrm{len}(a_n)$

Поэтому

$\mathrm{len}(b) < \mathrm{len}(b \cdot \mathrm{Chr}(0)) = \mathrm{len}(b) + \mathrm{len}(\mathrm{Chr}(0)) = \mathrm{len}(b) + 1$

Кстати, про $\mathrm{Chr}(0)$ мы можем заведомо утверждать, что $\mathrm{Chr}(0) \neq \ominus$, а поэтому и длина его равна 1 из-за теоремы (т.7.2) которая (напоминание):

$i \leq l_{ChrLim} \Rightarrow \mathrm{len}(\mathrm{Chr}(i)) = 1$:

Ведь для случая $i = 0$ заведомо выполнено $i \leq l_{ChrLim}$, каким бы натуральным числом l_{ChrLim} ни был.

А в силу того, что длины строк не равны – не равны и сами строки из-за теоремы (т.8.5) которая (напоминание):

$\mathrm{len}(a) \neq \ominus \Rightarrow [a = b \Leftrightarrow \forall i((\mathrm{len}(a) = \mathrm{len}(b) \wedge 0 < i \wedge i \leq \mathrm{len}(a)) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))]$

Поэтому для $u = \mathrm{Chr}(0)$ верно:

$\mathrm{Ins}(a \cdot u, i, b) \neq \mathrm{Ins}(a, i, b) \cdot u$

То есть, верно – после подстановки – следующее неравенство:

$\mathrm{Ins}(a \cdot b \cdot \mathrm{Chr}(0), i, b) \neq \mathrm{Ins}(a, i, b) \cdot b \cdot \mathrm{Chr}(0)$

В силу аксиомы логики предикатов из раздела 4. Приложение:

(b) $\mathrm{A}(a) \Rightarrow \exists x\, \mathrm{A}(x)$

Верно:

$\mathrm{Ins}(a \cdot u, i, b) \neq \mathrm{Ins}(a, i, b) \cdot u \Rightarrow \exists u\, \mathrm{Ins}(a \cdot u, i, b) \neq \mathrm{Ins}(a, i, b) \cdot u)$

И мы можем подставить на место свободных переменных конкретное значение $u = \mathrm{Chr}(0)$:

$\mathrm{Ins}(a \cdot \mathrm{Chr}(0), i, b) \neq \mathrm{Ins}(a, i, b) \cdot \mathrm{Chr}(0) \Rightarrow \exists u\, \mathrm{Ins}(a \cdot u, i, b) \neq \mathrm{Ins}(a, i, b) \cdot u)$

Посылка данной импликации доказана чуть выше, поэтому её можно отбросить по правилу вывода MP. И мы получаем:

$\exists u\, \mathrm{Ins}(a \cdot u, i, b) \neq \mathrm{Ins}(a, i, b) \cdot u)$

Что и требовалось доказать для Ветки 1.

Ветка 2. $\mathrm{len}(a) = \ominus \vee \mathrm{len}(b) = \ominus$

В этой ветке 2-е дедуктивное предположение является точным логическим отрицанием 1-го с его $\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus$. А из противоречия доказано всё. Поэтому для Ветки 2 заключение импликации 2 тоже доказано.

Ветка 3. $(\mathrm{len}(b) \neq 0 \wedge \mathrm{len}(a) < i + \mathrm{len}(b) - 1)$

А точнее, мы рассмотрим Ветку 3 в таком «рабочем виде»:

$i \neq 0 \wedge \mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus \wedge (\mathrm{len}(b) \neq 0 \wedge \mathrm{len}(a) < i + \mathrm{len}(b) - 1)$

Поясню такую «метаморфозу».

У нас 2-е дедуктивное предположение:

$i = 0 \vee \mathrm{len}(a) = \ominus \vee \mathrm{len}(b) = \ominus \vee (\mathrm{len}(b) \neq 0 \wedge \mathrm{len}(a) < i + \mathrm{len}(b) - 1)$

Если записать его в виде:

$A \vee B$

Где член дизъюнкции A – уже рассмотрен в «Ветка 1» и «Ветка 2», а сейчас мы рассматриваем в качестве посылки член дизъюнкции B. И «вдруг» подменяем его на:

$(\neg A \wedge B)$

Это возможно в том случае, если 2-е дедуктивное предположение в виде $A \vee B$ эквивалентно следующему:

$A \vee (\neg A \wedge B)$

А оно действительно эквивалентно в силу теоремы VI.10 $(A \vee B) \Leftrightarrow (A \vee (\neg A \wedge B))$ из раздела 4. Приложение.

Поэтому наша «подмена» дедуктивного предположения-2 Ветки 3:

$\mathrm{len}(b) \neq 0 \wedge \mathrm{len}(a) < i + \mathrm{len}(b) - 1$

на «Рабочую посылку 3»:

$i \neq 0 \wedge \mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus \wedge (\mathrm{len}(b) \neq 0 \wedge \mathrm{len}(a) < i + \mathrm{len}(b) - 1)$

является корректной.

Из рабочей посылки 3 нам нужно вывести заключение 2-й импликации:

$\exists u\, \mathrm{Ins}(a \cdot u, i, b) \neq \mathrm{Ins}(a, i, b) \cdot u)$

Для этого положим:

$u = \mathrm{ss}(i + \mathrm{len}(b) - 1 - \mathrm{len}(a))$

В качестве аргумента в $\mathrm{ss}()$ тут – натуральное число, не равное нулю, в силу

$\mathrm{len}(a) < i + \mathrm{len}(b) - 1$, из рабочей посылки 3.

Теперь просто вычислим размер $\mathrm{Ins}(a \cdot u, i, b)$ для которого верно:

$\mathrm{Ins}(a \cdot u, i, b) = \mathrm{Ins}(a \cdot \mathrm{ss}(i + \mathrm{len}(b) - 1 - \mathrm{len}(a)), i, b)$

Так как:

$\mathrm{len}(a \cdot \mathrm{ss}(i + \mathrm{len}(b) - 1 - \mathrm{len}(a))) = \mathrm{len}(a) + \mathrm{len}(\mathrm{ss}(i + \mathrm{len}(b) - 1 - \mathrm{len}(a)))$

То учтём, что по определению $\mathrm{ss}(i)$ верно: $\mathrm{len}(\mathrm{ss}(i)) = i$, поэтому:

$\mathrm{len}(a \cdot \mathrm{ss}(i + \mathrm{len}(b) - 1 - \mathrm{len}(a))) = \mathrm{len}(a) + i + \mathrm{len}(b) - 1 - \mathrm{len}(a) = i + \mathrm{len}(b) - 1$

А это значит, что наша реализация

$a \cdot u = a \cdot \mathrm{ss}(i + \mathrm{len}(b) - 1 - \mathrm{len}(a))$

удовлетворяет условию:

$\mathrm{len}(a \cdot u) \leq i + \mathrm{len}(b) - 1$

Кроме того, верно:

$\mathrm{len}(a \cdot u) \neq \ominus$

И выполнено:

$\mathrm{IsIns}(a \cdot u, i, b) = 1$

А поэтому воспользуемся теоремой (т.9.3), которая (напоминание):

$\mathrm{IsIns}(\mathrm{len}(a), i, \mathrm{len}(b)) = 1 \Rightarrow \mathrm{len}(\mathrm{Ins}(a, i, b))) = \mathrm{len}(a)$

Из неё мы непосредственно получаем:

$\mathrm{len}(\mathrm{Ins}(a \cdot u, i, b)) = \mathrm{len}(a \cdot u) = i + \mathrm{len}(b) - 1$, при $u = \mathrm{ss}(i + \mathrm{len}(b) - 1 - \mathrm{len}(a))$.

А теперь посчитаем $\mathrm{len}(\mathrm{Ins}(a, i, b) \cdot u)$:

$\mathrm{len}(\mathrm{Ins}(a, i, b) \cdot u) = \mathrm{len}(\mathrm{Ins}(a, i, b)) + \mathrm{len}(u)$

Из определения (9.6) для $\mathrm{Ins}(a, i, b)$:

$\mathrm{Ins}(a, i, b) = \mathrm{beg}(a, \max(i, 1) - 1) \cdot b \cdot \mathrm{end}(a, \mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b) + i))$

Для $\mathrm{beg}(a, \max(i, 1) - 1)$ верно:

$\mathrm{len}(\mathrm{beg}(a, \max(i, 1) - 1)) = \mathrm{len}(\mathrm{beg}(a, i - 1)) = \min(\mathrm{len}(a), i - 1)$

Действительно, $\max(i, 1) = i$, так как в Рабочей посылке 3 выполнено $i \neq 0$, а последняя строка в цепочке равенств – из следствия (с.6.9), которая (напоминание):

$\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = \min(i, \mathrm{len}(a)) \wedge (\mathrm{len}(\mathrm{end}(a, i)) = \max(\mathrm{len}(a), i + 1) - i + 1 \vee i = 0)$

Для $\mathrm{end}(a, \mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b) + i))$ верно:

$\mathrm{len}(\mathrm{end}(a, \mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b) + i))) = \mathrm{len}(\mathrm{end}(a, \mathrm{len}(b) + i))$

Действительно, $\min(\mathrm{len}(\mathrm{len}(b)), i) \neq 0$, так как в Рабочей посылке 3 выполнено $i \neq 0$, $\mathrm{len}(b) \neq 0$, $\mathrm{len}(b) \neq \ominus$. Из $\mathrm{len}(b) \neq \ominus$ следует $\mathrm{len}(\mathrm{len}(b)) \neq 0$ в силу (с.8.8) $\mathrm{len}(a) = \ominus \Leftrightarrow \mathrm{len}(\mathrm{len}(a)) = 0$. А поэтому $\mathrm{if}_0(\min(\mathrm{len}(\mathrm{len}(b)), i), 0, \mathrm{len}(b) + i)$ равно $\mathrm{len}(b) + i$.

Таким образом:

$\mathrm{len}(\mathrm{Ins}(a, i, b)) = \min(\mathrm{len}(a), i - 1) + \mathrm{len}(b) + \mathrm{len}(\mathrm{end}(a, \mathrm{len}(b) + i))$

Так как в Рабочей посылке 3 верно $\mathrm{len}(a) < i + \mathrm{len}(b) - 1$, то тем более $\mathrm{len}(a) < i + \mathrm{len}(b)$.

Поэтому в силу теоремы (т.6.9), которая (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(a) < i \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = \mathrm{len}(a) \wedge (\mathrm{len}(\mathrm{end}(a, i)) = 0)$

Получаем:

$\mathrm{len}(\mathrm{end}(a, \mathrm{len}(b) + i)) = 0$

Значит:

$\mathrm{len}(\mathrm{Ins}(a, i, b)) = \min(\mathrm{len}(a), i - 1) + \mathrm{len}(b) + 0 = \min(\mathrm{len}(a), i - 1) + \mathrm{len}(b)$

Вспоминая, что мы рассматриваем $u = \mathrm{ss}(i + \mathrm{len}(b) - 1 - \mathrm{len}(a))$ где $\mathrm{len}(u) = i + \mathrm{len}(b) - 1 - \mathrm{len}(a)$ получим:

$\mathrm{len}(\mathrm{Ins}(a, i, b) \cdot u) = \min(\mathrm{len}(a), i - 1) + \mathrm{len}(b) + \mathrm{len}(u) = \min(\mathrm{len}(a), i - 1) + \mathrm{len}(b) + i + \mathrm{len}(b) - 1 - \mathrm{len}(a)$

$\mathrm{len}(\mathrm{Ins}(a, i, b) \cdot u) = \min(\mathrm{len}(a), i - 1) + i - 1 - \mathrm{len}(a) + 2 \times \mathrm{len}(b)$

И сравним это с полученным выше:

$\text{len}(\text{Ins}(a \cdot u, i, b)) = i + \text{len}(b) - 1$

Тогда:

$\text{len}(\text{Ins}(a, i, b) \cdot u) - \text{len}(\text{Ins}(a \cdot u, i, b)) = [\min(\text{len}(a), i-1) + i - 1 - \text{len}(a) + 2 \times \text{len}(b)] - [i + \text{len}(b) - 1]$

$\text{len}(\text{Ins}(a \cdot u, i, b)) - \text{len}(\text{Ins}(a, i, b) \cdot u) = \min(\text{len}(a), i - 1) + \text{len}(b) - \text{len}(a)$

Так как в Рабочей посылке 3 верно $\text{len}(a) < i + \text{len}(b) - 1$, то результат последнего выражения больше 0, так как если $\min(\text{len}(a), i - 1) = i - 1$, то $\text{len}(a)$ вычитается от $i + \text{len}(b) - 1$, которая больше $\text{len}(a)$. А если $\min(\text{len}(a), i - 1) = \text{len}(a)$, то результат выражения равен $\text{len}(b)$, а $\text{len}(b) \neq 0$ по Рабочей посылке 3.

Таким образом:

$\text{len}(\text{Ins}(a \cdot u, i, b)) \neq \text{len}(\text{Ins}(a, i, b) \cdot u)$

Значит, по теореме (т.8.5), которая (напоминание)::

$\text{len}(a) \neq \ominus \Rightarrow [a = b \Leftrightarrow \forall i((\text{len}(a) = \text{len}(b) \wedge 0 < i \wedge i \leq \text{len}(a)) \Rightarrow \text{str}(a, i, 1) = \text{str}(b, i, 1))]$

Получаем:

$\text{Ins}(a \cdot u, i, b) \neq \text{Ins}(a, i, b) \cdot u$, при $u = \text{ss}(i + \text{len}(b) - 1 - \text{len}(a))$.

От этого конкретного значения u переходим к нужному нам заключению:

$\exists u \, \text{Ins}(a \cdot u, i, b) \neq \text{Ins}(a, i, b) \cdot u)$

Для перехода от предпоследнего предложения к последнему предложению используем тот же способ, который использовался при рассмотрении Ветки 1.

Все варианты посылок рассмотрены, поэтому импликация 2 тоже доказана.

Теорема доказана.

VIII. Критерий локального извлечении данных

Критерий локального извлечения данных:

(т.9.5) $\mathrm{len}(a) \neq \ominus \Rightarrow (\mathrm{IsStr}(\mathrm{len}(a), i, j) = 1 \Leftrightarrow \forall u\ \mathrm{str}(a \cdot u, i, j) = \mathrm{str}(a, i, j))$

Для доказательства этой эквивалентности нужно доказать, что в условиях 1-го дедуктивного предположения:

$\mathrm{len}(a) \neq \ominus$ истинны 2 импликации:

$\mathrm{IsStr}(\mathrm{len}(a), i, j) = 1 \Rightarrow \forall u\ \mathrm{str}(a \cdot u, i, j) = \mathrm{str}(a, i, j)$ – первая импликация;

$\forall u\ \mathrm{str}(a \cdot u, i, j) = \mathrm{str}(a, i, j) \Rightarrow \mathrm{IsStr}(\mathrm{len}(a), i, j) = 1$ – вторая импликация.

Для доказательства 1-й импликации делаем 2-е дедуктивное предположение:

$\mathrm{IsStr}(\mathrm{len}(a), i, j) = 1$

Из определения (9.8) и 2-го дедуктивного предположения получаем:

$\mathrm{len}(a) \neq \ominus \wedge (i = 0 \vee j = 0 \vee \mathrm{len}(a) \geq i + j - 1)$

Ветка 1. $i = 0$

Тогда из определения $\mathrm{str}()$ и из-за (2.2) $\mathrm{end}(a, 0) = \ominus$ и (2.4) $\mathrm{beg}(\ominus, q) = \ominus$ получим:

$\mathrm{str}(a \cdot u, i, j) = \mathrm{str}(a \cdot u, 0, j) = \mathrm{beg}(\mathrm{end}(a \cdot u, 0), j) = \ominus$

Аналогично:

$\mathrm{str}(a, i, j) = \ominus$

Поэтому

$\mathrm{str}(a \cdot u, i, j) = \mathrm{str}(a, i, j)$

Первая импликация для Ветки 1 доказана из последнего равенства в виде:

$\mathrm{IsStr}(\mathrm{len}(a), i, j) = 1 \Rightarrow \forall u\ \mathrm{str}(a \cdot u, i, j) = \mathrm{str}(a, i, j))$

А переход к квантору общности делается по аксиоме из Раздела 4. Приложение:

(α) Если верно $U \Rightarrow \mathrm{B}(a)$, то верно $U \Rightarrow \forall x\, \mathrm{B}(x)$

Ветка 2. $j = 0$

Тогда из определения $\mathrm{str}()$ и из-за (2.1) $\mathrm{beg}(a, 0) = \ominus$ получим:

$\mathrm{str}(a \cdot u, i, j) = \mathrm{str}(a \cdot u, i, 0) = \mathrm{beg}(\mathrm{end}(a \cdot u, i), 0) = \ominus$

Аналогично:

$\mathrm{str}(a, i, j) = \ominus$

Поэтому

$\mathrm{str}(a \cdot u, i, j) = \mathrm{str}(a, i, j)$

Завершение доказательства Ветки 2 с квантором общности – как для Ветки 1.

Ветка 3. $(i \neq 0 \wedge j \neq 0 \wedge \mathrm{len}(a) \geq i + j - 1)$

Почему рассмотренные Ветки 1 и 2 $(i = 0 \vee j = 0)$ превратились в своё отрицание, и добавились как член конъюнкции к оставшемуся для рассмотрения варианту – см. теорему VI.10 $(A \vee B) \Leftrightarrow (A \vee (\neg A \wedge B))$ в разделе 4. Приложение.

Теперь, чтобы вычислить, чему равна функция $\mathrm{str}(a \cdot u, i, j)$, представим переменную a следующим образом в соответствии с аксиомой (1) и с учётом $i \neq 0$ получим:

$a = \mathrm{beg}(a, i - 1) \cdot \mathrm{end}(a, i)$

$\mathrm{end}(a, i) = \mathrm{beg}(\mathrm{end}(a, i), j) \cdot \mathrm{end}(\mathrm{end}(a, i), j')$

На основании (т.4.6) $\mathrm{str}(a,1,j) = \mathrm{beg}(a,j)$ получаем:

$\mathrm{beg}(a,i-1) = \mathrm{str}(a,1,i-1)$

На основании определения (4.1) для $\mathrm{str}()$ получаем:

$\mathrm{beg}(\mathrm{end}(a,i),j) = \mathrm{str}(a,i,j)$

На основании (т.3.2) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{end}(a,i+j-1) = \mathrm{end}(\mathrm{end}(a,i),j)$ получаем:

$\mathrm{end}(\mathrm{end}(a,i),j') = \mathrm{end}(a,i+j)$

Теперь можно переписать a следующим образом:

$a = \mathrm{str}(a,1,i-1) \cdot \mathrm{str}(a,i,j) \cdot \mathrm{end}(a,i+j)$

На основании следствия (с.6.7) которое (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge i \neq 0 \wedge i+j-1 \leq \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{str}(a,i,j)) = j$

Получаем с учётом того, что $\mathrm{len}(a) \geq i+j-1$:

$\mathrm{len}(\mathrm{str}(a,1,i-1)) = i-1$

$\mathrm{len}(\mathrm{str}(a,i,j)) = j$

На основании трёх последних предложений (не считая (с.6.7)), ассоциативности конкатенации и теоремы (т.9.1) которая (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(b) \neq \ominus \Rightarrow \mathrm{str}(a \cdot b \cdot u, \mathrm{len}(a)', \mathrm{len}(b)) = b$

Получаем:

$\mathrm{str}(a \cdot u,i,j) = \mathrm{str}(\mathrm{str}(a,1,i-1) \cdot \mathrm{str}(a,i,j) \cdot \mathrm{end}(a,i+j) \cdot u,i,j) = \mathrm{str}(a,i,j)$

Завершение доказательства Ветки 3 – как для Ветки 1.

Первая импликация доказана.

Теперь надо доказать вторую импликацию:

$\forall u\ \mathrm{str}(a \cdot u,i,j) = \mathrm{str}(a,i,j) \Rightarrow \mathrm{IsStr}(\mathrm{len}(a),i,j) = 1$

Перепишем её в эквивалентном виде при помощи контрапозиции, превращая квантор общности в квантор существования при помощи теоремы VI.7 из раздела 4. Приложение:

$\mathrm{IsStr}(\mathrm{len}(a),i,j) \neq 1 \Rightarrow \exists u\ \mathrm{str}(a \cdot u,i,j) \neq \mathrm{str}(a,i,j)$

Очевидно, что

$\mathrm{IsStr}(\mathrm{len}(a),i,j) \neq 1 \Leftrightarrow \mathrm{IsStr}(\mathrm{len}(a),i,j) = 0$

Поэтому перепишем 2ю импликацию так:

$\mathrm{IsStr}(\mathrm{len}(a),i,j) = 0 \Rightarrow \exists u\ \mathrm{str}(a \cdot u,i,j) \neq \mathrm{str}(a,i,j)$

Для доказательства 2-й импликации делаем 2-е дедуктивное предположение:

$\mathrm{IsStr}(\mathrm{len}(a),i,j) = 0$

Из определения (9.8) и 2-го дедуктивного предположения получаем «рабочий вариант» 2-го дизъюнктивного предположения:

$\mathrm{len}(a) = \ominus \vee (i \neq 0 \wedge j \neq 0 \wedge \mathrm{len}(a) < i+j-1)$

Член дизъюнкции $\mathrm{len}(a) = \ominus$ в данном предложении является отрицанием 1-го дедуктивного предположения, поэтому удаляем его из дизъюнкции. И тогда первое и второе дедуктивные предположения дают общее дедуктивное предположение:

$\mathrm{len}(a) \neq \ominus \wedge i \neq 0 \wedge j \neq 0 \wedge \mathrm{len}(a) < i+j-1$

Из общего дедуктивного предположения нам нужно вывести заключение 2-й импликации:

$\exists u\ \mathrm{str}(a \cdot u, i, j) \neq \mathrm{str}(a, i, j)$

Для этого положим:

$u = \mathrm{ss}(i + j - 1 - \mathrm{len}(a))$

Тогда, учитывая, что по определению $\mathrm{ss}(i)$ верно: $\mathrm{len}(\mathrm{ss}(i)) = i$, получим:

$\mathrm{len}(a \cdot u) = \mathrm{len}(a) + \mathrm{len}(u) = \mathrm{len}(a) + \mathrm{len}(\mathrm{ss}(i + j - 1 - \mathrm{len}(a)))$

$\mathrm{len}(a \cdot u) = \mathrm{len}(a) + i + j - 1 - \mathrm{len}(a) = i + j - 1$

На основании следствия (с.6.7) которое (напоминание):

$\mathrm{len}(a) \neq \ominus \wedge i \neq 0 \wedge i + j - 1 \leq \mathrm{len}(a) \Rightarrow \mathrm{len}(\mathrm{str}(a, i, j)) = j$

Получаем с учётом того, что $\mathrm{len}(a \cdot u) \geq i + j - 1$:

$\mathrm{len}(\mathrm{str}(a \cdot u, i, j)) = j$

С другой стороны, из-за (т.4.2) $(i \neq 0 \wedge j \neq 0) \Rightarrow \mathrm{str}(a, i, j) = \mathrm{end}(\mathrm{beg}(a, i + j - 1), i)$ получим:

$\mathrm{str}(a, i, j) = \mathrm{end}(\mathrm{beg}(a, i + j - 1), i)$

На основании (т.8.2) $\mathrm{len}(a) \neq \ominus \Rightarrow a = \mathrm{str}(a, 1, \mathrm{len}(a))$ и (т.4.6) $\mathrm{str}(a, 1, j) = \mathrm{beg}(a, j)$ получаем:

$\mathrm{str}(a, i, j) = \mathrm{end}(\mathrm{beg}(\mathrm{beg}(a, \mathrm{len}(a)), i + j - 1), i)$

На основании (т.3.1) $\mathrm{beg}(a, \min(i, j)) = \mathrm{beg}(\mathrm{beg}(a, i), j)$ и $\mathrm{len}(a) < i + j - 1$ из общего дедуктивного предположения получаем:

$\mathrm{str}(a, i, j) = \mathrm{end}(\mathrm{beg}(a, \mathrm{len}(a)), i)$

Используя снова (т.4.6) $\mathrm{str}(a, 1, j) = \mathrm{beg}(a, j)$ и (т.8.2) $\mathrm{len}(a) \neq \ominus \Rightarrow a = \mathrm{str}(a, 1, \mathrm{len}(a))$ получаем:

$\mathrm{str}(a, i, j) = \mathrm{end}(a, i)$

Тогда на основании $i \neq 0$ из общего дедуктивного предположения и следствия (с.6.9), которая (напоминание):

$\mathrm{len}(a) \neq \ominus \Rightarrow \mathrm{len}(\mathrm{beg}(a, i)) = \min(i, \mathrm{len}(a)) \wedge (\mathrm{len}(\mathrm{end}(a, i)) = \max(\mathrm{len}(a), i - 1) - i + 1 \vee i = 0)$

Получаем:

$\mathrm{len}(\mathrm{str}(a, i, j)) = \mathrm{len}(\mathrm{end}(a, i)) = \max(\mathrm{len}(a), i - 1) - i + 1$

и надо его сравнить с полученным ранее

$\mathrm{len}(\mathrm{str}(a \cdot u, i, j)) = j$

При том, что $\mathrm{len}(a) < i + j - 1$ из общего дедуктивного предположения.

Перепишем последнее неравенство так:

$\mathrm{len}(a) - (i - 1) < j$

Левая часть $\mathrm{len}(a) - (i - 1)$ может быть и отрицательной, но тогда

$\max(\mathrm{len}(a), i - 1) - i + 1 = 0 < j$

А если $\mathrm{len}(a) - (i - 1)$ не отрицательна, то

$\max(\mathrm{len}(a), i - 1) - i + 1 = \mathrm{len}(a) - (i - 1) < j$

То есть, в любом случае:

$\max(\mathrm{len}(a), i - 1) - i + 1 < j$

$\mathrm{len}(\mathrm{str}(a, i, j)) < \mathrm{len}(\mathrm{str}(a \cdot u, i, j))$

$\mathrm{len}(\mathrm{str}(a, i, j)) \neq \mathrm{len}(\mathrm{str}(a \cdot u, i, j))$

Тогда на основании теоремы (т.8.5), которая (напоминание):

$\mathrm{len}(a) \neq \ominus \Rightarrow [a = b \Leftrightarrow \forall i((\mathrm{len}(a) = \mathrm{len}(b) \wedge 0 < i \wedge i \leq \mathrm{len}(a)) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))]$

Получаем:

$\mathrm{str}(a, i, j) \neq \mathrm{str}(a \cdot u, i, j)$

Поэтому для $u = \mathrm{ss}(i + j - 1 - \mathrm{len}(a))$ верно:

$\mathrm{str}(a, i, j) \neq \mathrm{str}(a \cdot u, i, j)$

То есть, верно – после подстановки – следующее неравенство:

$\mathrm{str}(a, i, j) \neq \mathrm{str}(a \cdot u = \mathrm{ss}(i + j - 1 - \mathrm{len}(a)), i, j)$

В силу аксиомы логики предикатов из раздела 4. Приложение:

(b) $\mathrm{A}(a) \Rightarrow \exists x\, \mathrm{A}(x)$

Верно:

$\mathrm{str}(a, i, j) \neq \mathrm{str}(a \cdot u, i, j) \Rightarrow \exists u\ \mathrm{str}(a, i, j) \neq \mathrm{str}(a \cdot u, i, j)$

И мы можем подставить на место свободных переменных конкретное значение $u = \mathrm{ss}(i + j - 1 - \mathrm{len}(a))$:

$\mathrm{str}(a, i, j) \neq \mathrm{str}(a \cdot \mathrm{ss}(i + j - 1 - \mathrm{len}(a)), i, j) \Rightarrow \exists u\ \mathrm{str}(a, i, j) \neq \mathrm{str}(a \cdot u, i, j)$

Посылка данной импликации доказана чуть выше, поэтому её можно отбросить по правилу вывода MP. И мы получаем:

$\exists u\ \mathrm{str}(a, i, j) \neq \mathrm{str}(a \cdot u, i, j)$

Что и требовалось доказать из общего дедуктивного предположения, поэтому импликация 2 тоже доказана.

Теорема доказана.

3 Числа в теории строк

То, что в теории строк мы опираемся в «конструировании» объектов не только на ноль, создаёт свойства для строк сверх тех свойств, которые есть у чисел в теории Пеано. У строк теперь есть собственная структура «цепочка символов». Вопрос – как эту же структуру придать числам, которые используются в теории строк? Это будет та структура, которой нет у чисел арифметики (арифметики Пеано, в частности), но есть у чисел в позиционном представлении.

Что касается чисел, которые использованы в аксиомах теории строк, то они подчиняются обычным свойствам арифметики, и тут мы имеем (в неявном виде пока) ещё аксиомы теории Пеано или её арифметического расширения. Запишем явным образом аксиомы теории Q, достаточной для представления рекурсивных функций, а затем и её расширение до теории Пеано.

Дж. Булос, Р. Джеффри «Вычислимость и логика» Глава 14. Представимость в Q. Аксиомы теории Q.

(Q_1) $\forall i \forall j (i' = j' \Rightarrow i = j)$
(Q_2) $\forall i \, 0 \neq i'$
(Q_3) $\forall i (i \neq 0 \Rightarrow \exists j \, i = j')$
(Q_4) $\forall i \, i + 0 = i$
(Q_5) $\forall i \forall j (i + j' = (i + j)')$
(Q_6) $\forall i \, i \times 0 = 0$
(Q_7) $\forall i \forall j (i \times (j') = (i \times j) + i)$

Замечу, что для знака умножения использован символ «$\times$», потому что символ «$\cdot$» уже занят для обозначения конкатенации.

Глава 15 Неразрешимость, неопределимость и полнота. В той же книге. Добавление аксиом индукции к теории Q и переход к теории Пеано – «теории Z».

$((\mathrm{A}(0) \wedge \forall i (\mathrm{A}(i) \Rightarrow \mathrm{A}(i'))) \Rightarrow \forall i (\mathrm{A}(i)))$ – все формулы такого вида на языке теории строк, но с навешиванием на такие формулы квантора общности $\forall i$.

Разумеется, для теории строк формула $\mathrm{A}(i)$ понимается как формула, которая может зависеть и от разных строковых аргументов, которые не являются числами.

При этом аксиому Q_3 можно исключить, так как она выводится из аксиомы индукции и аксиомы Q_2.

До данного момента мы разделяли между собой просто строки и строки, которые являются ещё и числами. Вот аксиомы Пеано – это аксиомы, которые выполнены для не просто строк, но для строк со свойствами чисел. Теперь, сохраняя те же условности обозначения для строк, имеющих свойства чисел (для переменных со значением числа мы используем оговоренные ранее буквы) зададим связь между алфавитом для строк и числами.

Чтобы придать числам свойства строк, достаточно задать строку, которая будет 0 (нулём) и задать операцию добавления единицы (функцию следования). После этого остальные арифметические операции легко выводятся на основании свойств строк и аксиом арифметики Пеано.

Вот нужная для «превращения» некоторых строк ещё и в числа аксиома, которую можно при

желании разбить на отдельные аксиомы – отдельная аксиома из каждого члена следующей конъюнкции:

$(0 \leq l_{NumBeg} < l_{NumLim} \leq l_{ChrLim})$

$\wedge 0 = \mathrm{Chr}(l_{NumBeg})$

$\wedge (l_{NumBeg} \leq j < l_{NumLim} \Rightarrow (\mathrm{Chr}(j))' = \mathrm{Chr}(j'))$

$\wedge \mathrm{Chr}(l_{NumLim})' = (\mathrm{Chr}(l_{NumBeg})') \cdot \mathrm{Chr}(l_{NumBeg})$

$\wedge ((k \neq 0 \wedge l_{NumBeg} \leq j < l_{NumLim}) \Rightarrow (k \cdot \mathrm{Chr}(j))' = k \cdot \mathrm{Chr}(j'))$

$\wedge (k \neq 0 \Rightarrow (k \cdot \mathrm{Chr}(l_{NumLim}))' = (k') \cdot \mathrm{Chr}(l_{NumBeg}))$

Пояснения:

1. Мы задаём «алфавит в алфавите» - для цифр.

2. Первая цифра в этом алфавите – символ «0». Вот его мы и объявляем числом 0 (ноль) в этой части из приведённой аксиомы:

$0 = \mathrm{Chr}(l_{NumBeg})$

Дальше идёт логика добавления 1 в разных случаях.

3. Сначала рассматриваем случай, когда у нас число состоит из любой цифры, кроме последней цифры «алфавита цифр»:

$l_{NumBeg} \leq j < l_{NumLim} \Rightarrow (\mathrm{Chr}(j))' = \mathrm{Chr}(j')$

При увеличении такого числа на 1 происходит просто замена прежней цифры на следующую. Например, для десятичной системы:

$5' = 6$ или $5 + 1 = 6$

4. Затем рассматриваем случай, когда у нас число состоит из последней цифры «алфавита цифр». В десятичной системе это – 9:

$\mathrm{Chr}(l_{NumLim})' = (\mathrm{Chr}(l_{NumBeg})') \cdot \mathrm{Chr}(l_{NumBeg})$

На примере 9 это правило выглядит так:

$9' = 10$ или $9 + 1 = 10$

Заметим, что тут мы используем строковую операцию конкатенации, соединяя в правой части равенства следующую цифру после 0 (так как $1 = 0'$) и сам 0, который возник вместо 9.

5. Затем рассматриваем случай, когда последняя цифра – не 9 (если разбирать на примере десятичной системы) и она не единственная в числе (усложнение пункта 3). Тогда последняя цифра просто заменяется на следующую в алфавите цифру, когда число увеличено на 1:

$23345' = 23346$

В данном примере $k =$ «2334» и он соединён (конкатенация) с цифрой «5». Условие

$k \neq 0$

понятно, потому что числа у нас не имеют вида «05», например. Поэтому разбираемая часть аксиомы имеет вид:

$(k \neq 0 \wedge l_{NumBeg} \leq j < l_{NumLim}) \Rightarrow (k \cdot \mathrm{Chr}(j))' = k \cdot \mathrm{Chr}(j')$

6. И в заключение рассматриваем случай, когда последняя цифра в числе является последней цифрой ещё и в алфавите цифр:

$(k \neq 0 \Rightarrow (k \cdot \mathrm{Chr}(l_{NumLim}))' = (k') \cdot \mathrm{Chr}(l_{NumBeg}))$

Тогда последняя цифра заменяется на 0 (ноль), а вся предыдущая часть числа заменяется на число, которое больше этой предыдущей части на 1. На примере десятичного числа:

$123799' = 123800$

Видно, что число у нас из двух соединённых строк – «12379» и «9». При увеличении такого числа на 1 получается соединение из 2 новых строк – «12380» и «0».

Этой аксиоматизации достаточно для задания всех чисел – от 0 и до бесконечности. Из этой аксиомы и остальных аксиом для строк и арифметики можно уже вывести правила сложения, умножения, вычитания и деления «в столбик» и любые другие правила для операций с числами в позиционном представлении, записанных при помощи строк.

Замечу, что в аксиоме использованы для обозначения чисел те буквы, которые были оговорены для этого в начале предыдущего раздела: i, j, k, l, m, n. Неявно это означает, что в каждой аксиоме имеется посылка о том, что используемый на месте числа объект – число (а не просто строка). Например, полная запись аксиомы (8.1) из предыдущего раздела будет:

$a = b \Leftrightarrow (\forall i\, \mathrm{Num}(i) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1))$

Где одноместная предикатная буква $\mathrm{Num}(i)$ как раз «проверяет», является ли данная строка числом. Но раз мы условились об обозначении числа буквой i, то считаем это условие (про то, что тут число) исполненным и посылку с $\mathrm{Num}(i)$ пропускаем:

$a = b \Leftrightarrow \forall i\ \mathrm{str}(a, i, 1) = \mathrm{str}(b, i, 1)$

Это обычная практика – как, например, в теории множеств принято изображать множества строчными, а классы – прописными буквами, а посылки с проверками на то, что объект – множество – типа $\mathrm{M}(x)$ – тогда не пишут.

Тем не менее, одноместная предикатная буква $\mathrm{Num}(i)$ о проверке строки на то, что она ещё и число – имеется в теории. И в связи с аксиомой о связи чисел и строк можно вывести:

$\mathrm{Num}(a) \Leftrightarrow$

$\mathrm{len}(a) \neq \ominus \wedge (\mathrm{str}(a, 1, 1) = \mathrm{Chr}(l_{NumBeg}) \Rightarrow a = \mathrm{str}(a, 1, 1))$

$\wedge \forall i \leq \mathrm{len}(a) : l_{NumBeg} \leq \mathrm{Ach}(\mathrm{str}(a, i, 1)) \leq l_{NumLim}$

Пояснение: при вышеизложенной аксиоматизации о связи чисел и строк получается, что число представляет собой конечную строку из одних цифр, притом цифра «0» (ноль) может стоять на первом месте только в числе 0 (ноль).

Но представленная аксиоматизация для связи чисел и строк – совершенно не единственно возможная. Мы вполне можем написать аксиомы для представления чисел в духе «стандартной интерпретации». Вот так:

Аксиоматизация нуля:

$0 = \mathrm{Chr}(0)$

Аксиоматизация увеличения на 1:

$i' = i \cdot \mathrm{Chr}(0)$

Пояснение: Символы с номером 0 (ноль) мы используем как «счётные палочки». И их количество минус 1 и есть само число. А «кучи» этих «счётных палочек» мы организуем при помощи конкатенации.

При такой аксиоматизации можно вывести:

$i + j = \mathrm{end}(i \cdot j, 0'')$

$i - j = \mathrm{end}(i, \mathrm{len}(j))$, для $i > j$

Ну, а случай

$i \times j$

Окажется конкатенацией из $\mathrm{len}(j) - 1$ одинаковых строк $\mathrm{end}(i, 0'')$ и конкатенация $\mathrm{Chr}(0)$ сверх того:

$\mathrm{end}(i, 0'') \cdot \mathrm{end}(i, 0'') \ldots \mathrm{end}(i, 0'') \cdot \mathrm{end}(i, 0'') \cdot \mathrm{Chr}(0)$

Ну, или конкатенацией из $\mathrm{len}(i) - 1$ строк $\mathrm{end}(j, 0'')$ и конкатенация $\mathrm{Chr}(0)$ сверх того:

$\mathrm{end}(j, 0'') \cdot \mathrm{end}(j, 0'') \ldots \mathrm{end}(j, 0'') \cdot \mathrm{end}(j, 0'') \cdot \mathrm{Chr}(0)$

Напоминаю, что для знака умножения использован символ «$\times$», потому что символ «$\cdot$» уже занят для обозначения конкатенации.

При таких аксиомах связи чисел и строк будет верно:

$\mathrm{Num}(a) \Leftrightarrow \mathrm{len}(a) \neq \ominus \wedge \mathrm{len}(a) \neq 0 \wedge (\forall i\, 1 \leq i \leq \mathrm{len}(a) \Rightarrow \mathrm{str}(a, i, 1) = \mathrm{Chr}(0))$

Как видим, одни и те же аксиомы арифметики допускают представление чисел и в таком виде, когда размер представления числа равен величине числа. Это совершенно не годится для правильной оценки размеров данных и времени работы с ними при рассмотрении алгоритмов, работающих со строками и числами в позиционном представлении. Впрочем, это уже было доказано в предыдущем разделе.

4 Приложение. Логика теорий первого порядка с равенством

В данном разделе приведены аксиомы логики исчисления предикатов, аксиомы равенства и сведения о некоторых базовых теоремах логики. Использована аксиоматика из книги Д. Гильберт, П. Бернайс «Основания математики. Логические исчисления и формализация арифметики». Далее кратко «Основания математики».

Выбранная аксиоматика представляется наиболее интуитивно понятной и удобной для построения логических выводов среди известных мне. К тому же опора на общеизвестный классический труд в значительной мере страхует от каких-то незамеченных читателями неточностей.

«Основания математики», Глава III. Исчисление высказываний. Параграф 3. Дедуктивная логика высказываний. Подпараграф 2. Одна система исходных формул для дедуктивной логики высказываний: полнота этой системы.

I. Формулы для импликации:

1) $A \Rightarrow (B \Rightarrow A)$

2) $(A \Rightarrow (A \Rightarrow B)) \Rightarrow (A \Rightarrow B)$

3) $(A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$

II. Формулы для конъюнкции:

1) $A \wedge B \Rightarrow A$

2) $A \wedge B \Rightarrow B$

3) $(A \Rightarrow B) \Rightarrow ((A \Rightarrow C) \Rightarrow (A \Rightarrow B \wedge C))$

III. Формулы для дизъюнкции:

1) $A \Rightarrow A \vee B$

2) $B \Rightarrow A \vee B$

3) $(A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \vee B \Rightarrow C))$

IV. Формулы для эквивалентности:

1) $(A \Leftrightarrow B) \Rightarrow (A \Rightarrow B)$

2) $(A \Leftrightarrow B) \Rightarrow (B \Rightarrow A)$

3) $(A \Rightarrow B) \Rightarrow ((B \Rightarrow A) \Rightarrow (A \Leftrightarrow B)$

V. Формулы для отрицания:

1) $(A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A)$

2) $A \Rightarrow \neg\neg A$

3) $\neg\neg A \Rightarrow A$

«Основания математики», Глава IV. Формализация процесса вывода II: Исчисление предикатов. Параграф 2. Связанные переменные и правила для кванторов. Подпараграф 4. Окончательная формулировка правил исчисления предикатов; изображение форм категорических суждений; случай пустой индивидной области.

В качестве исходных формул разрешается брать тождественные формулы исчисления высказываний. К ним мы добавляем обе основные формулы:

(a) $\forall x\, \mathrm{A}(x) \Rightarrow \mathrm{A}(a)$

(b) $\mathrm{A}(a) \Rightarrow \exists x\, \mathrm{A}(x)$

Тут я от себя добавлю пояснение, что аксиома (b) – это бесконечное множество импликаций. И мы можем выбирать ту, которая удобна нам. Например, при доказательстве существования для (5.3) в подразделе II раздела 2 мы взяли как бы $\mathrm{A}(l'_{ChrLim})$, за счёт чего воспользовались правилом силлогизма и эта посылка исчезла из итоговой формулы. Но за счёт аксиомы (b) мы получаем возможность выбрать путь (значение переменной a), который позволяет достичь нужной цели.

В качестве схем вывода, позволяющих получить новые формулы из ранее полученных, у нас имеется, наша первоначальная схема заключения <MP>

Если верно U и верно $U \Rightarrow R$, то верно R.

И кроме того, две новые схемы:

(α) Если верно $U \Rightarrow \mathrm{B}(a)$, то верно $U \Rightarrow \forall x\, \mathrm{B}(x)$

и

(β) Если верно $\mathrm{B}(a) \Rightarrow U$, то верно $\exists x\, \mathrm{B}(x) \Rightarrow U$

Обе они могут применяться лишь тогда, когда в первой формуле схемы переменная a встречается лишь на местах, указанных в качестве аргумента, а x не входит в $\mathrm{B}(a)$.

«Основания математики», Глава V. Исчисление предикатов с равенством. Полнота одноместного исчисления предикатов:

(J_1) $a = a$

(J_2) $a = b \Rightarrow (\mathrm{A}(a) \Rightarrow \mathrm{A}(b))$

Где $\mathrm{A}(a)$ представляет собой любую логически корректную формулу, построенную из функциональных букв, предикатных букв теории, кванторов (в область действия которых не попадают a, b) и логических связок. При этом заменять a на b можно как везде, где написано a, так и в некоторых (произвольно выбранных) местах написания переменной a.

Теперь перечислим некоторые ключевые теоремы про логику первого порядка с равенством.

VI. Ключевые теоремы

1) Теорема дедукции

Простой вариант:

Если при добавлении к теории T утверждения A можно вывести утверждение B без использования правил вывода (α) и (β), то в теории T без добавления утверждения A можно вывести утверждение $A \Rightarrow B$.

Но мы использовали и более тонкий вариант этой теоремы:

Использовать правила вывода (α) и (β) можно, но только так, чтобы при этом не связывалась на одна свободная переменная утверждения A.

Или, записывая более формально:

Если есть вывод $T, A \vdash B$ без использования (α) и (β) в отношении свободных переменных утверждения A, то есть вывод $T \vdash A \Rightarrow B$.

2) Вложенные посылки эквивалентны конъюнкции посылок

Утверждение:

$A_1 \Rightarrow (A_2 \Rightarrow \ldots \Rightarrow (A_N \Rightarrow B)\ldots)$

Эквивалентно утверждению:

$(A_1 \wedge A_2 \wedge \ldots \wedge A_N) \Rightarrow B$

Поэтому последовательность посылок во вложенных импликациях можно менять произвольным образом.

3) Правило С. Смотри, например, Э. Мендельсон, «Введение в математическую логику», Глава 2. Теория первого порядка. Раздел 7. Правило С.

Если у нас в теории T имеется истинное утверждение вида $\exists x\, \mathrm{A}(x)$, то есть, имеется вывод:

$T \vdash \exists x\, \mathrm{A}(x))$

И имеется ещё такой вывод:

$T, \mathrm{A}(u) \vdash B$, без использования (α) и (β) по u,

То найдётся вывод:

$T \vdash B$

4) Определение равенства. Смотри, например, Э. Мендельсон, «Введение в математическую логику», Глава 2. Теории первого порядка. Раздел 8. Теория первого порядка с равенством. Предложение 2.26. <критерий того, что в теории первого порядка истинны аксиомы равенства>

Если в аксиомах теории есть только функциональные буквы и знак равенства, то аксиомы равенства (J_1) и (J_2) для неё будут выполнены тогда и только тогда, когда

a. Для каждой функциональной буквы $\mathrm{f_n}()$ с аргументами $x_1, \ldots x_m$ (где m – количество аргументов функции $\mathrm{f_n}(x_1, \ldots x_m)$) истинно:

$(x_1 = y_1 \wedge \ldots \wedge x_m = y_m) \Rightarrow (\mathrm{f_n}(x_1, \ldots x_m) = \mathrm{f_n}(y_1, \ldots y_m))$

b. Для знака равенства истинна рефлексивность:

$x = x$

c. Для знака равенства истинна транзитивность:

$x = y \Rightarrow (z = x \Rightarrow z = y)$

5) Введение новых функциональных букв и предметных констант. Смотри, например, Э. Мендельсон, «Введение в математическую логику», Глава 2. Теории первого порядка. Раздел 9. Введение новых функциональных букв и предметных констант. Следствие 3.3.

Если для формулы $\mathrm{A}(u, x_1, \ldots x_m)$ выполнено:

$\exists_1 u\, \mathrm{A}(u, x_1, \ldots x_m)$

То можно аксиоматизировать функцию $\mathrm{f}(x_1, \ldots x_m)$ следующей формулой:

$\mathrm{A}(\mathrm{f}(x_1, \ldots x_m), x_1, \ldots x_m)$

И добавление данной аксиомы к теории не добавляет никакой дополнительной логики, кроме нового обозначения. В частности, такая функция будет соответствовать аксиомам равенства, если данная теория до введения этого обозначения была теорией первого порядка с равенством.

Напоминание – обозначение «существует и единственно»

$\exists_1 u\, \mathrm{A}(u, x_1, \ldots x_m)$ означает выполнение 2 условий:

a. $\exists u\, \mathrm{A}(u, x_1, \ldots x_m)$

b. $\mathrm{A}(u_1, x_1, \ldots x_m) \wedge \mathrm{A}(u_2, x_1, \ldots x_m) \Rightarrow u_1 = u_2$

6) Эквивалентные формулы логики высказывания взаимозаменяемы внутри любых формул логики. Например, если доказана формула

$A \Rightarrow (\neg B \vee C)$

И при этом верно:

$B \Leftrightarrow D$

То будет доказана и формула:

$A \Rightarrow (\neg D \vee C)$

7) $\neg\forall n \neg \mathrm{A}(n) \Leftrightarrow \exists n\, \mathrm{A}(n)$

Доказательство.

Ветка 1

$\forall n \neg \mathrm{A}(n) \Rightarrow \neg\, \mathrm{A}(i)$, по (a) $\forall x\, \mathrm{A}(x) \Rightarrow \mathrm{A}(a)$

$\mathrm{A}(i) \Rightarrow \neg\forall n \neg \mathrm{A}(n)$, из предыдущего, контрапозиция

$\exists n\, \mathrm{A}(n) \Rightarrow \neg\forall n \neg \mathrm{A}(n)$, из предыдущего по (β) Если верно $\mathrm{B}(a) \Rightarrow U$, то верно $\exists x\, \mathrm{B}(x) \Rightarrow U$

Ветка 2

$\neg\, \mathrm{A}(i) \Rightarrow \exists n \neg \mathrm{A}(n)$, по (b) $\mathrm{A}(a) \Rightarrow \exists x\, \mathrm{A}(x)$

$\neg\exists n \neg \mathrm{A}(n) \Rightarrow \mathrm{A}(i)$, из предыдущего, контрапозиция

$\neg\exists n \neg \mathrm{A}(n) \Rightarrow \forall n\, \mathrm{A}(n)$, из предыдущего по (α) Если верно $U \Rightarrow \mathrm{B}(a)$, то верно $U \Rightarrow \forall x\, \mathrm{B}(x)$

$\neg\forall n\, \mathrm{A}(n) \Rightarrow \exists n \neg \mathrm{A}(n)$, из предыдущего, контрапозиция

$\neg\forall n \neg \mathrm{A}(n) \Rightarrow \exists n\, \mathrm{A}(n)$, из предыдущего, заменяя $\mathrm{A}(n)$ на $\neg\, \mathrm{A}(n)$ и $\neg(\neg\, \mathrm{A}(n))$ на $\mathrm{A}(n)$ в соответствии с п.6)

Соединяя ветку 1 и 2 получим:

$\neg\forall n \neg \mathrm{A}(n) \Leftrightarrow \exists n\, \mathrm{A}(n)$

8) $\forall i(B \vee \mathrm{A}(i)) \Leftrightarrow B \vee \forall i(\mathrm{A}(i))$

Доказательство

Ветка 1.

$\mathrm{A}(n) \Rightarrow \exists i\, \mathrm{A}(i)$

$B \wedge \mathrm{A}(n) \Rightarrow B \wedge \exists i\, \mathrm{A}(i)$

$\exists i(B \wedge \mathrm{A}(n)) \Rightarrow B \wedge \exists i\, \mathrm{A}(i)$

$\neg B \vee \neg\exists i\, \mathrm{A}(i) \Rightarrow \neg\exists i(B \wedge \mathrm{A}(n))$

$\neg B \vee \forall i \neg\, \mathrm{A}(i) \Rightarrow \forall i \neg(B \wedge \mathrm{A}(n))$

$\neg B \vee \forall i \neg\, \mathrm{A}(i) \Rightarrow \forall i(\neg B \vee \neg\, \mathrm{A}(n))$

$B \vee \forall i\, \mathrm{A}(i) \Rightarrow \forall i(B \vee \mathrm{A}(n))$

Ветка 2.

$\forall i(B \vee \mathrm{A}(i)) \Rightarrow (B \vee \mathrm{A}(n))$

$\forall i(B \vee \mathrm{A}(i)) \Rightarrow (\neg B \Rightarrow \mathrm{A}(n))$

$[\neg B \wedge \forall i(B \vee \mathrm{A}(i))] \Rightarrow \mathrm{A}(n)$

$[\neg B \wedge \forall i(B \vee \mathrm{A}(i))] \Rightarrow \forall i\, \mathrm{A}(i)$

$\forall i(B \vee \mathrm{A}(i)) \Rightarrow (\neg B \Rightarrow \forall i\, \mathrm{A}(i))$

$\forall i(B \vee \mathrm{A}(i)) \Rightarrow (B \vee \forall i\, \mathrm{A}(i))$

Доказано.

9) $\exists i(B \vee \mathrm{A}(i)) \Leftrightarrow B \vee \exists i\, \mathrm{A}(i)$

Доказательство.

Ветка 1.

$\forall i\, \mathrm{A}(i) \Rightarrow \mathrm{A}(n)$

$B \wedge \forall i\, \mathrm{A}(i) \Rightarrow B \wedge \mathrm{A}(n)$

$B \wedge \forall i\, \mathrm{A}(i) \Rightarrow \forall i(B \wedge \mathrm{A}(i))$

$\neg\forall i(B \wedge \mathrm{A}(i)) \Rightarrow \neg B \vee \neg\forall i\, \mathrm{A}(i)$

$\exists i\neg(B \wedge \mathrm{A}(i)) \Rightarrow \neg B \vee \exists i\neg\, \mathrm{A}(i)$

$\exists i(\neg B \vee \neg\, \mathrm{A}(i)) \Rightarrow \neg B \vee \exists i\neg\, \mathrm{A}(i)$

$\exists i(B \vee \mathrm{A}(i)) \Rightarrow B \vee \exists i\, \mathrm{A}(i)$

Ветка 2.

$\mathrm{A}(n) \Rightarrow B \vee \mathrm{A}(n)$

$B \Rightarrow B \vee \mathrm{A}(n)$

$B \vee \mathrm{A}(n) \Rightarrow \exists i(B \vee \mathrm{A}(i))$

$\mathrm{A}(n) \Rightarrow \exists i(B \vee \mathrm{A}(i))$

$B \Rightarrow \exists i(B \vee \mathrm{A}(i))$

$\exists i\, \mathrm{A}(u) \Rightarrow \exists i(B \vee \mathrm{A}(i))$

$B \vee \exists i\, \mathrm{A}(u) \Rightarrow \exists i(B \vee \mathrm{A}(i))$

Доказано.

10) $(A \vee B) \Leftrightarrow (A \vee (\neg A \wedge B))$

Пояснение.

Данная эквивалентность часто используется, когда доказываются импликации вида:

$(A_1 \vee A_2 \vee \ldots \vee A_n) \Rightarrow D$

Для доказательства тогда надо доказывать импликации вида:

$A_1 \Rightarrow D$

$A_2 \Rightarrow D$

. . .

$A_n \Rightarrow D$

А после этого использовать аксиому III.3) $(A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \vee B \Rightarrow C))$ несколько раз.

Но зачастую в приведённой цепочке импликаций вместо импликации $A_n \Rightarrow D$ доказывают импликацию:

$(\neg A_1 \wedge \neg A_2 \wedge \ldots \wedge A_n) \Rightarrow D$

И то, что стоит слева от A_n в последней импликации – это как раз $\neg A$, если считать, что $(A_1 \vee A_2 \vee \ldots A_{n-1})$ – это A, а A_n – это B из доказываемой нами формулы VI.10. Поэтому и происходит подмена этого B в импликации $B \Rightarrow D$ на $(\neg A \wedge B)$, что вместо исходной посылки:

$(A_1 \vee A_2 \vee \ldots \vee A_n)$

Можно заменить её на эквивалентную:

$(A_1 \vee A_2 \vee \ldots \vee (\neg A_1 \wedge \neg A_2 \wedge \ldots \wedge A_n))$

И такая замена оказывается возможной как раз благодаря данной теореме VI.10.

Данную импликацию VI.10 можно проверить по таблицам истинности, можно дать логический вывод, но поясню, почему эта эквивалентность истинная – третьим способом – через «булеву алгебру».

$A \vee B$ эквивалентно $A \vee (\neg A \wedge B)$ по следующей причине:

$B \Leftrightarrow (A \vee \neg A) \wedge B \Leftrightarrow (A \wedge B) \vee (\neg A \wedge B)$

Поэтому:

$(A \vee B) \Leftrightarrow A \vee (A \wedge B) \vee (\neg A \wedge B)$

Но так как для A тоже верно:

$A \Leftrightarrow (A \wedge B) \vee (A \wedge \neg B)$

То

$A \Leftrightarrow A \wedge (A \wedge B)$

Ведь $A \vee (A \wedge B)$ ничем не отличается от $(A \wedge B) \vee (A \wedge \neg B) \vee (A \wedge B)$, где последний член просто повторяет первый, а дизъюнкция первых двух – эквивалентна A.

А в силу последней эквивалентности верно:

$A \vee (A \wedge B) \vee (\neg A \wedge B) \Leftrightarrow A \vee (\neg A \wedge B)$

Откуда и получаем искомое:

$(A \vee B) \Leftrightarrow A \vee (\neg A \wedge B)$

Теорема доказана.

НАЧАЛО подраздела 5.

5 Приложение. Перспективы использования ТКС и пример – 2-я т. Гёделя о неполноте

С теорией «компьютерных» строк (ТКС) радикально упрощается исследование вопросов работы алгоритмов и расчёт эффективности этой работы. Теперь нет нужды рассматривать программный текст как нечто отличающееся от данных. Мы вполне можем (что я планирую сделать во 2-й статье) рассматривать программу как строку. Названия команд этой программы будут практически совпадать с названиями функций из ТКС, а сам программный текст будет размещён на одном из «служебных» лучей данных МКА. На другом «служебном луче» будет размещён номер текущей исполняемой команды. Мы можем завести ещё сколько угодно «служебных» лучей данных где сможем считать время работы, размер использованных данных и что угодно.

Текст программы мы сможем изменять в процессе работы программы, как и номер текущей команды. Программы становятся такими же данными, как и любые другие данные. И сама теория у нас теперь именно про строки, а теоремы и доказательства теории мы тоже можем исследовать как объекты самой теорий «компьютерных» строк.

Теперь можно будет радикально пересмотреть принципиальные формулировки об алгоритмах и их преобразованиях между собой. В качестве «входных данных» алгоритма теперь можно рассматривать и сам текст его программы как доступную для обработки и учёта при проверках строку. А это крайне важно для уточнения понятия сложности, потому что сложность относительна – зависит от того, кто/что рассматривает соответствующий вопрос.

Например, если я знаю код, открывающий сейф, то для меня вопрос его открытия – простой. Если я этого не знаю, то вопрос – сложный. И кто имеет дело с поставленным вопросом, можно будет видеть по тексту программы. В том числе и сама программа сможет «смотреть» на свой программный текст.

Логики уже почти век как используют подобные методы. Теперь такая возможность появляется и в теории алгоритмов.

Имея теорию строк с конкатенацией, возможностью поиска, взятия подстрок и т.п. можно гораздо проще в техническом отношении решать множество вопросов.

Например, докажем (схематично) 2-ю теорему Гёделя о неполноте. 1-я теорема Гёделя о неполноте в форме Россера доказывается легко и можно прочитать в любых учебниках логики для ВУЗов. А вот за доказательством 2-й теоремы Гёделя все известные мне книги отсылают к 2-му тому классических «Оснований математики» Гильберта и Бернайса, либо предлагают какие-то не типичные дополнения к формальной логике.

Для доказательства нам потребуются сведения о 1-й теореме Гёделя в форме Гёделя.

Есть утверждение Гёделя G о своей собственной недоказуемости в рамках теории Пеано. Мы будем формализовать его для теории строк (или любого непротиворечивого расширения теории строк). Легко доказывается следующая эквивалентность (в любом учебнике логики):

1) $\neg \mathrm{P}(\text{«}G\text{»}) \Leftrightarrow G$, где «$G$» обозначает текст утверждения Гёделя (рассуждаем не в терминах «гёделевых номеров», а в терминах строк или «текста» - не формально говоря).

В силу эквивалентности 1) в рамках нашей теории невозможно доказать утверждение G, потому что вместе с доказательством утверждения G будет доказано и утверждение $\mathrm{P}(G)$ – так как доказуемость тоже доказывается для доказанных утверждений. И текст доказательства доказуемости $\mathrm{P}(X)$ утверждения X строится по определенному алгоритму из текста доказательства самого утверждения X. А вместе с этим из эквивалентности 1) будет доказано $\neg\,\mathrm{P}(\text{«}G\text{»})$. Как видим, получается противоречие, если бы удалось доказать G. Поэтому:

2) G невозможно доказать в рамках нашей теории (в теории строк – для неё разбираем).

Так просто доказывается первая половина теоремы Гёделя в форме Гёделя. Для доказательства неполноты надо было бы доказать невозможность доказательства $\neg G$ и для этого потребовалось бы разбираться с нетривиальными вопросами вроде ω-непротиворечивости. Но нам для 2-ой теоремы Гёделя нужна только первая часть 1-й теоремы Гёделя о неполноте в форме Гёделя. А кто хочет «досмотреть» доказательство 1-й теоремы Гёделя о неполноте – может обратиться в учебники за 1-й т. Гёделя о неполноте в форме Россера. Никаких ω-непротиворечивостей разбирать не потребуется. Мы же здесь вернёмся к 2-й т. Гёделя.

Предикат доказуемости обычно определяется следующим образом:

$\mathrm{P}(y) : \exists x\ \mathrm{p}(y, x)$

Где $\mathrm{p}(y, x)$ – предикат проверки доказательства теоремы y (текст теоремы предикат p(y, x) получает в первом аргументе), а y – текст доказательства, который предикат $\mathrm{p}(y, x)$ получает во втором аргументе.

Но мы дадим иное определение для $\mathrm{P}(y)$ и иначе построим $\mathrm{p}(y, x)$ – это будет функция, а не предикат. И это даст нам возможность доказывать всякие равенства, а на их основе – через аксиомы равенств – получать необходимые для доказательства импликации.

Строим функцию проверки соответствия текста доказательства x тексту теоремы y с такими свойствами:

3.1) $\mathrm{p}(y, x) = 1$, если значением переменной x является текст логического вывода для теоремы, текст которой – в переменной y;

3.2) $\mathrm{p}(y, x) = 0$, если значением переменной x является текст, негодный в качестве логического вывода для утверждения, текст которого – в переменной y. Если текст в переменной y абсурдный, то никакой логический вывод для него не подойдёт, разумеется.

Такие алгоритмы проверки доказательств есть, и даже используются на практике. Поэтому никаких принципиальных трудностей построить функцию $\mathrm{p}(y, x)$ – нет.

Тогда:

3.3) $\mathrm{P}(y) : \exists x\ \mathrm{p}(y, x) = 1$

Что такое «текст вывода»? Это конкатенация «строк текста». «Строк текста» не в смысле «строк» как объектов теории строк, а как части большой строки («текста»), притом каждая такая часть заканчивается символом перевода строки, например. И каждая такая «строка текста» является текстом логической формулы (кроме символа перевода строки), которая логически следует из одной или 2-х предыдущих строк текста по одному из правил логического вывода.

Для правильного доказательства, при $\mathrm{p}(y, x) = 1$ будет верно:

3.4) $y = \mathrm{last}(x)$

Где функция $\mathrm{last}(x)$ просто возвращает последнюю «строку текста» из текста x. Для программистов очевидно, что это очень простая функция. То есть – текст утверждения в переменной y тоже заканчивается символом перевода строки.

Что касается «превращения» текста доказательства x для теоремы Y, текст которой находится в переменной y в текст доказательства для $P(\text{«}Y\text{»})$, где «Y» обозначает подстановку текста утверждения Y в предикат доказуемости, то и для этого введём функцию – пусть $\mathrm{pp}(x)$. Эта функция будет генерировать верное доказательство для $\mathrm{P}(\text{«}Y\text{»})$ только из того x, которое действительно доказывает y. Построить такую функцию можно и в том нет принципиальных проблем. Тогда будет доказано:

3.5) $\mathrm{p}(y, x) = \mathrm{p}(\mathrm{ArgTo1}(\text{« P}(y)\text{»}, y), \mathrm{pp}(x))$

Где $\mathrm{ArgTo1}(\text{« P}(y)\text{»}, y)$ подставляет в текст алгоритма проверки $\mathrm{P}(y)$ значение аргумента y. И у нас вместо формулы $\mathrm{P}(y)$ с переменной y получается текст функции $\mathrm{P}(\ldots)$ с конкретным значением вместо переменной. Один из простых вариантов подстановки, кстати, такой:

$\exists y(y = \ldots \wedge \mathrm{P}(y))$, где вместо многоточия – нужное значение.

После всех этих подготовительных шагов мы с лёгкостью докажем (но чуть ниже разберём «лёгкость»):

4) $\mathrm{P}(\text{«}G\text{»}) \Rightarrow \mathrm{P}(\text{« P}(\text{«}G\text{»}) \wedge \neg\,\mathrm{P}(\text{«}G\text{»})\text{»})$, откуда по контрапозиции получим:

$\neg\,\mathrm{P}(\text{« P}(\text{«}G\text{»}) \wedge \neg\,\mathrm{P}(\text{«}G\text{»})\text{»}) \Rightarrow \neg\,\mathrm{P}(\text{«}G\text{»})$, а теперь по эквивалентности 1):

5) $\neg\,\mathrm{P}(\text{« P}(\text{«}G\text{»}) \wedge \neg\,\mathrm{P}(\text{«}G\text{»})\text{»}) \Rightarrow G$

Последнее предложение 5) и есть 2-я теорема Гёделя о неполноте. Если будет доказана недоказуемость противоречия, то будет доказано и утверждение Гёделя. А его доказательство означает противоречивость теории, как мы видели выше.

Неважно, что в посылке импликации противоречие представлено «экзотическим»

$\mathrm{P}(\text{«}G\text{»}) \wedge \neg\,\mathrm{P}(\text{«}G\text{»})$

В противоречивой теории было бы за пару шагов доказано что угодно и это противоречие тоже. Поэтому его недоказуемость означала бы доказательство непротиворечивость теории. Но после такого доказательства оказывается доказанным утверждение G. А это приводит к противоречию. Говоря на «популярном» уровне:

Если система доказала свою правоту, то она противоречива (она заведомо ошибается).

Теперь посмотрим на «лёгкость» доказательства утверждения

4) $\mathrm{P}(\text{«}G\text{»}) \Rightarrow \mathrm{P}(\text{« P}(\text{«}G\text{»}) \wedge \neg\,\mathrm{P}(\text{«}G\text{»})\text{»})$

Доказывать его будем из следующего равенства:

4.1) $\mathrm{p}(\mathrm{ArgTo2}(\text{«}(x) \wedge (y)\text{»}, y_1, y_2), x_1 \cdot x_2 \cdot \mathrm{ProofAnd2}(y_1, y_2)) = \mathrm{p}(y_1, x_1) \times \mathrm{p}(y_2, x_2)$

Это равенство довольно очевидное и означает, что если у нас есть текст доказательства x_1 для утверждения Y_1 (его текст в y_1) и текст доказательства x_2 для утверждения Y_2 (его текст в y_2), то из этих текстов доказательств можно легко построить текст доказательства для конъюнкции этих утверждений:

$(Y_1) \wedge (Y_2)$

Текст доказательства для этой конъюнкции такой: это конкатенация текстов доказательств для Y_1 и Y_2, плюс конкатенация тех нескольких строк логического вывода, в которых из утверждений Y_1 и Y_2 (из предположения, что каждое из этих утверждений доказано в тексте выше) выводится их конкатенация $(Y_1) \wedge (Y_2)$.

Кстати,

$\mathrm{ProofAnd2}(y_1, y_2) = \mathrm{ProofAnd2}(\mathrm{last}(x_1), \mathrm{last}(x_2))$ в силу 3.4)

И введем обозначение

$\mathrm{ProveAnd2}(x_1, x_2) = x_1 \cdot x_2 \cdot \mathrm{ProofAnd2}(\mathrm{last}(x_1), \mathrm{last}(x_2))$

Конкретизируем 4.1) с учётом последнего обозначения:

4.2) $\mathrm{p}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, \mathrm{ProveAnd2}(x_1, x_2)) = \mathrm{p}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\text{»}, x_1) \times \mathrm{p}(\text{«}\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, x_2)$

Из пункта 3.5) получаем:

$\mathrm{p}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\text{»}, \mathrm{pp}(x)) = \mathrm{p}(\text{«}G\text{»}, x)$

Что же касается $\mathrm{p}(\text{«}\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, x_2)$, то имея доказательство для G, мы легко достраиваем текст этого доказательства до текста для доказательства $\neg\,\mathrm{P}(\text{«}G\text{»})$ из-за эквивалентности 1), так как для эквивалентности 1) тоже есть текст его доказательства. Назовём функцию для перехода от текста доказательства x для утверждения G к тексту доказательства для $\neg\,\mathrm{P}(\text{«}G\text{»})$ так:

$\mathrm{pg}(x)$. Тогда верно:

$\mathrm{p}(\text{«}\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, \mathrm{pg}(x)) = \mathrm{p}(\text{«}G\text{»}, x)$

Таким образом, из 4.2) и двух последних равенств получаем:

4.3) $\mathrm{p}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, \mathrm{ProveAnd2}(\mathrm{pp}(x), \mathrm{pg}(x))) = \mathrm{p}(\text{«}G\text{»}, x)^2 = \mathrm{p}(\text{«}G\text{»}, x)$

Сократим $\mathrm{ProveAnd2}(\mathrm{pp}(x), \mathrm{pg}(x))$ до обозначения $\mathrm{f}(x)$ и получим:

4.4) $\mathrm{p}(\text{«}G\text{»}, x) = \mathrm{p}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, \mathrm{f}(x))$

Из последнего равенства и свойств равенства получаем:

$\mathrm{p}(\text{«}G\text{»}, x) = 1 \Rightarrow \mathrm{p}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, \mathrm{f}(x)) = 1$

Используя VI.12 $\mathrm{Q}(\mathrm{f}(x)) \Rightarrow \exists y\,\mathrm{Q}(y)$ из раздела 4. Приложение:

$\mathrm{p}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, \mathrm{f}(x)) = 1 \Rightarrow \exists y(\mathrm{p}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, y) = 1)$

Из двух последних формул и силлогизма:

$\mathrm{p}(\text{«}G\text{»}, x) = 1 \Rightarrow \exists y(\mathrm{p}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}, y) = 1)$, по определению 3.3) получаем:

$\mathrm{p}(\text{«}G\text{»}, x) = 1 \Rightarrow \mathrm{P}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»})$, Из ($\beta$) Если верно $\mathrm{B}(a) \Rightarrow U$, то верно $\exists x\,\mathrm{B}(x) \Rightarrow U$ получаем:

$\exists x(\mathrm{p}(\text{«}G\text{»}, x) = 1) \Rightarrow \mathrm{P}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»})$, по определению 3.3) получаем:

4) $\mathrm{P}(\text{«}G\text{»}) \Rightarrow \mathrm{P}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»})$

Пункт 4 подробно рассмотрен и доказан. Дальнейшее доказательство до

5) $\neg\,\mathrm{P}(\text{«}\,\mathrm{P}(\text{«}G\text{»})\wedge\neg\,\mathrm{P}(\text{«}G\text{»})\text{»}) \Rightarrow G$

проведено выше. 2-я теорема Гёделя о неполноте доказана.

Пункт 4) – самый сложный. Нужно доказать именно импликацию, а не то, что если доказано одно, тогда доказано и другое. Потому что нам нужна контрапозиция, а воспользоваться ею можно только с импликацией. Из-за проблем в получении импликации пункта 4) и отсылают к доказательству во 2-й том «Оснований математики» Гёделя и Бернайса.

Но используя функцию проверки доказательства, вместо предиката, и технику работу со строками, мы смогли относительно просто вывести нужную импликацию.

Список литературы

[1] *Дж. Булос, Р. Джеффри.* Вычислимость и логика. Мир, М. 1994

[2] *Э. Мендельсон.* Введение в математическую логику. Наука, М. 1984

[3] *Д. Гильберт, П. Бернайс.* Основания математики. Логические. исчисления и формализация арифметики. Наука, М. 1979

[4] *В.В. Ященко (редактор), Ю.В. Нестеренко (глава 4) и др.* Введение в криптографию. МЦНМО, М. 2012

[5] *А.А. Бухштаб.* Теория чисел. Издательство «Просвещение», М. 1966

[6] *А.И. Мальцев.* Алгоритмы и рекурсивные функции. Наука, М. 1986

[7] *В.И. Игошин.* Теория алгоритмов. Инфра-М, М. 2013

[8] *А.С. Герасимов.* Курс математической логики и теории вычислимости. Издательство «Лань», Санкт-Петербург 2014

[9] *Л.П. Жильцова, Т.Г. Смирнова.* Основы теории автоматов и формальных языков в примерах и задачах: учебно-методическое пособие. ННГУ, Нижний Новгород 2017

[10] *А.Е. Пентус, М.Р. Пентус.* Теория формальных языков: Учебное пособие ЦПИ при механико-математическом ф-те МГУ. М. 2014

[11] *М. Клайн.* Математика. Утрата определённости. Мир, М. 1984

[12] *Н.Н. Непейвода.* Прикладная логика. Новосибирский центр по математическим наукам, Новосибирск 1996

[13] *Albert Visser.* Growing Commas. A Study of Sequentiality and Concatenation Notre Dame Journal of Formal Logic, Notre Dame, USA. 2009

[14] *John Corcoran; William Frank; Michael Maloney.* String Theory. The Journal of Symbolic Logic, Vol. 39, No. 4, Storrs, USA. 1974