32. Алгоритмически неразрешимые проблемы. Примеры.

**Алгоритмически неразрешимые проблемы -** задачи, для которых нет эффективного алгоритма (дающего результат и не бесконечного)

Неразрешимость связана с невозможностью реализовать алгоритм решения некоторой(конкретной?) задачи на машине тьюринга. Либо с проблемой массовости. Когда алгоритм не может быть применим к классу задач, но применим к конкретной задаче из класса. Либо наоборот. Алгоритм применим к классу задач, кроме каких то случаев. Происходит нарушение свойства массовости алгоритма. Алгоритм можно составить для любой задачи, но для класса задач такой алгоритм придумать не удается.

**Теорема.** Проблем распознавания самоприменимости алгоритма является неразрешимой.

Рассмотрим вариант универсальной МТ, когда на ленте изображен ее собственный шифр в алфавите машины. Естестевенно считать, что если МТ обрабатывает этот шифр – она самоприменима, иначе – несамоприменима.

Предположим, что такая МТ, анализирующая собственный шифр и выносящая решение о самоприменимости, существует. Тогда в этой МТ всякий самоприменимый шифр перерабатывается в некоторый символ «вэ» (положительный ответ), а всякий несамоприменимый шифр в символ «тэ» (отрицательный ответ).

При таком допущении можно построить такую МТ’, которая перерабатывает несамоприменимые шифры «тэ», но неприменима к саприменимым шифрам. Это достигается введением такого изменения таблицы соотвествия, когда после появления символа «вэ» вместо остановки МТ’ бесконечно повторяла бы «вэ», «вэ», «вэ», …

Итак, МТ’ применима ко всякому несамоприменимому шифру и неприменима к самоприменимым шифрам. Это приводит к противоречию.

1. Пусть МТ’ самоприменима, то есть она применима к собственному шифру МТ’ и останавливается, но тогда она несамоприменима, так как вырабатывает «тэ».
2. Пусть МТ’ несамоприменима, что означает ее применимость к её собственному шифру МТ’, то есть она самоприменима.

Указанное противоречие и есть доказательство теоремы, так как оно опровергает утверждение о существовании МТ.

Из этой теоремы вытекает алгоритмическая неразрешимость более широкой проблемы – проблемы распознавания применимости, заключающейся в необходимости установить применимость любой МТ к любой заданной конфигурации.

Определение. Ассоциативным исчислением называется совокупность всех слов в некотором алфавите А с конечной системой подстановок.

Чтобы задать ассоциативное исчисление достаточно указать алфавит А и систему подстановок. Подстановки бывают вида « p< - >q » или «p - >q», где p, q – слова в алфавите А. Неориентированная подстановка позволяет замену в прямом и обратном направлениях, ориентированная – лишь в прямом. Допустим, задан алфавит А = (a, b, c) и одна подстановка «ab<->bcb», а также p = abcbcbab. Заменим вхождения ab: bcbcbcbbcb, а затем используем подстановку в обратном направлении: abcbcbab->aabcbab->aaabab.

Если слово r может быть получено из слова s применением лишь одной подстановки, то говорят, что r и s смежные слова. Если слово r может быть получено из слова s и r применением конечного числа подстановок, то говорят, что смежные слова образуют «дедуктивную цепочку от s к r». Наконец, слова s и r называются «эквивалентными», если существует дедуктивная цепочка от s и r.

Для отношения эквивалентности слов справедливо следующее.

1. Рефлексивность
2. Симметричность
3. Транзитивность

Проблема эквивалентности в ассоциативном исчислении состоит в необходимости определения для любых 2 слов в данном ассоциативном исчислении эквивалентны они или нет.

**Определение.** Логика является полной, если в рамках её можно доказать истинность или ложность каждого утверждения.

**Определение.** Логика является непротиворечивой, если она свободна о противоречий (например, в ней нельзя получить одновременно истинное и ложное утверждения).

**Теорема.** Каждая адекватная непротиворечивая арифметическая логика неполна.

Математический смысл теоремы в том, что в арифметической логике существуют истинные утверждения о целых положительных числах, которые нельзя вывести и доказать средствами этой логики.

**Алгоритмически неразрешимая задача -** задача, для которой не существует алгоритма, который получив входные данные, останавливался и выдавал бы ответ.

**Пример** проблемы - проблема останова. Проблема определения завершаема работа машины или нет. Ведь если машина не остановилась, это не значит что она не остановится потом. Может быть мы мало ждали.

Для конкретного алгоритма можно провести анализ и узнать, завершаема ее работа или нет. Но нет универсального алгоритма который определял завершаемость любого алгоритма.

В проблеме останова мы же можем взять и провести анализ самостоятельно. Применяя разные подходы. И в конце определить остановится машина или нет. По сути мы же делаем некую последовательность действий по некоторым правилам. Это уже некий алгоритм. Но для каждого случая он свой.  Проблема поиска наименьшего корня. Мы вводили функцию наименьшего корня в МБР. Для конкретной функции мы можем провести анализ, но единого универсального алгоритма для всех функций не существует.