Модели вычислений

В.А. Захаров, Р.И. Подловченко

Лекция 5.

1. Универсальные машины Тьюринга
2. Неразрешимость проблемы останова
3. Сводимость алгоритмических проблем
4. Теорема Райса
5. Кодирования МТ

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Как было показано на предыдущей лекции, одни МТ (например, 1-ленточные МТ) могут моделировать работу других МТ (например, n-ленточных).

Но 80 лет назад К. Гедель, А. Тьюринг, Э. Пост, С. Клини, А. Черч обнаружили, что некоторые модели вычислений (включая машины Тьюринга) обладают удивительной способностью:

можно иметь всего лишь одну программу, которая позволяет моделировать работу всех программ вычислительной модели.

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Как было показано на предыдущей лекции, одни МТ (например, 1-ленточные МТ) могут моделировать работу других МТ (например, n-ленточных).

Но 80 лет назад К. Гедель, А. Тьюринг, Э. Пост, С. Клини, А. Черч обнаружили, что некоторые модели вычислений (включая машины Тьюринга) обладают удивительной способностью:

можно иметь всего лишь одну программу, которая позволяет моделировать работу всех программ вычислительной модели.

Такие программы называются **универсальными**.

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Каждая МТ $\mathcal{M} = (\Sigma, \Gamma, Q, q_1, q_0, q_{-1}, T)$ полностью определяется отношением перехода T и тройкой особо выделенных состояний q_1, q_0, q_{-1}. Отношение переходов можно представить конечным списком команд вида $K = (q', a : b, q'', D)$. Наименования всех состояний МТ $\mathcal{M}$, а также символы смещение считывающей головки МТ L, R можно закодировать (например, используя блочное кодирование) словами в алфавите МТ Σ.

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Каждая МТ $\mathcal{M} = (\Sigma, \Gamma, Q, q_1, q_0, q_{-1}, T)$ полностью определяется отношением перехода T и тройкой особо выделенных состояний q_1, q_0, q_{-1}. Отношение переходов можно представить конечным списком команд вида $K = (q', a : b, q'', D)$. Наименования всех состояний МТ $\mathcal{M}$, а также символы смещение считывающей головки МТ L, R можно закодировать (например, используя блочное кодирование) словами в алфавите МТ Σ.

Тогда описанием каждой команды $K = (q', a : b, q'', D)$ будет слово

$$code(K) = code(q')\ a\ b\ code(q'')code(D),$$

а описанием МТ $\mathcal{M}$ — слово

$$code(\mathcal{M}) = code(K_1)code(K_2)\ldots code(K_N)code(q_1)code(q_0)code(q_{-1}).$$

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Универсальной машиной Тьюринга (УМТ) называется МТ $\mathcal{U}$, обладающая следующим свойством:

для любой МТ $\mathcal{M}$ и для любого слова w УМТ $\mathcal{U}$, имея на входной ленте начальную конфигурацию $q_1 code(\mathcal{M})\#w$, проводит вычисление, которое

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Универсальной машиной Тьюринга (УМТ) называется МТ $\mathcal{U}$, обладающая следующим свойством:

для любой МТ $\mathcal{M}$ и для любого слова w УМТ $\mathcal{U}$, имея на входной ленте начальную конфигурацию $q_1 code(\mathcal{M})\#w$, проводит вычисление, которое

- завершается в допускающем состоянии тогда и только тогда, когда МТ $\mathcal{M}$ допускает слово w;

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Универсальной машиной Тьюринга (УМТ) называется МТ $\mathcal{U}$, обладающая следующим свойством:

для любой МТ $\mathcal{M}$ и для любого слова w УМТ $\mathcal{U}$, имея на входной ленте начальную конфигурацию $q_1 code(\mathcal{M})\#w$, проводит вычисление, которое

- завершается в допускающем состоянии тогда и только тогда, когда МТ $\mathcal{M}$ допускает слово w;
- завершается в отвергающем состоянии тогда и только тогда, когда МТ $\mathcal{M}$ отвергает слово w.

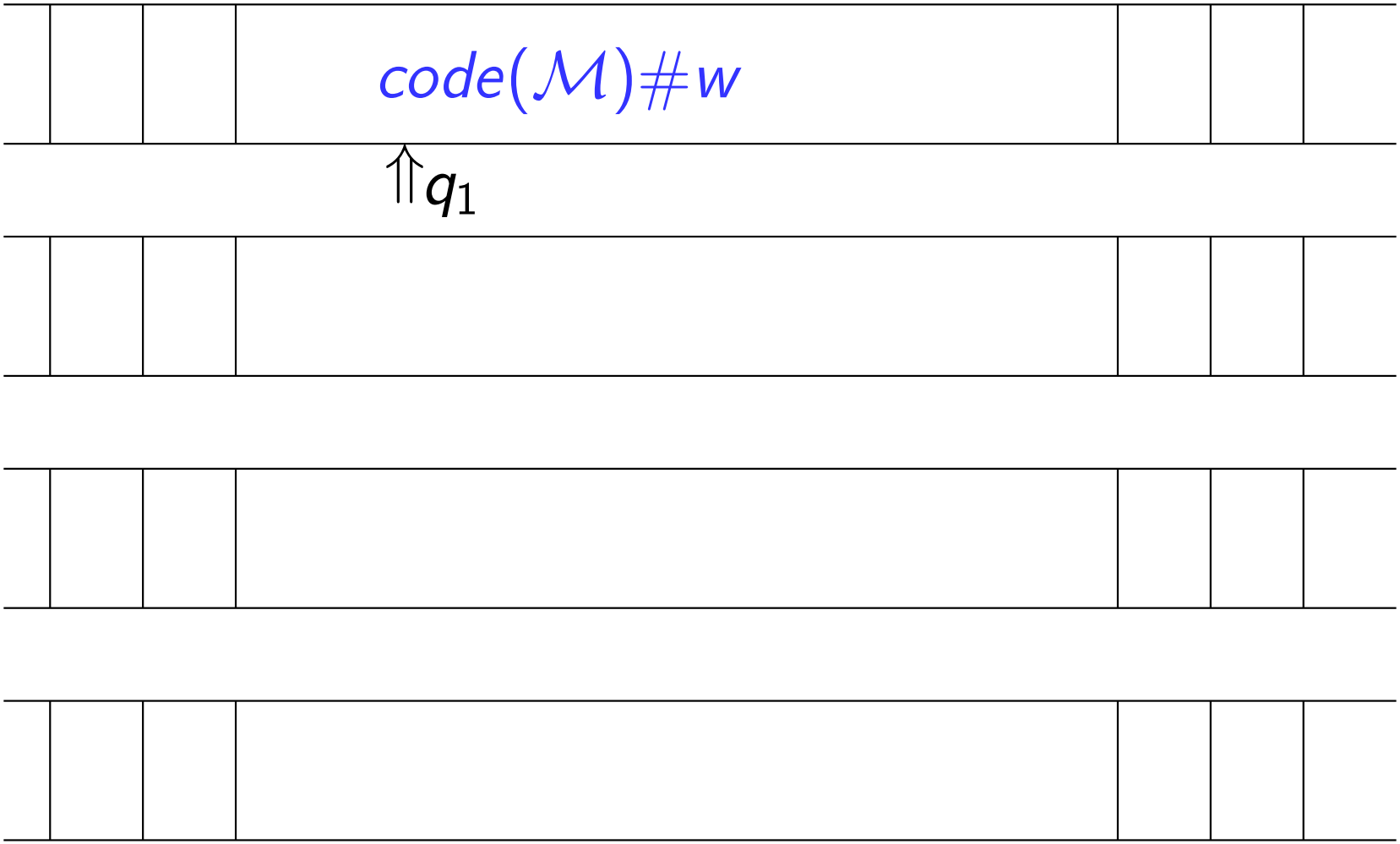
УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Теорема 5.1. Универсальные машины Тьюринга существуют.

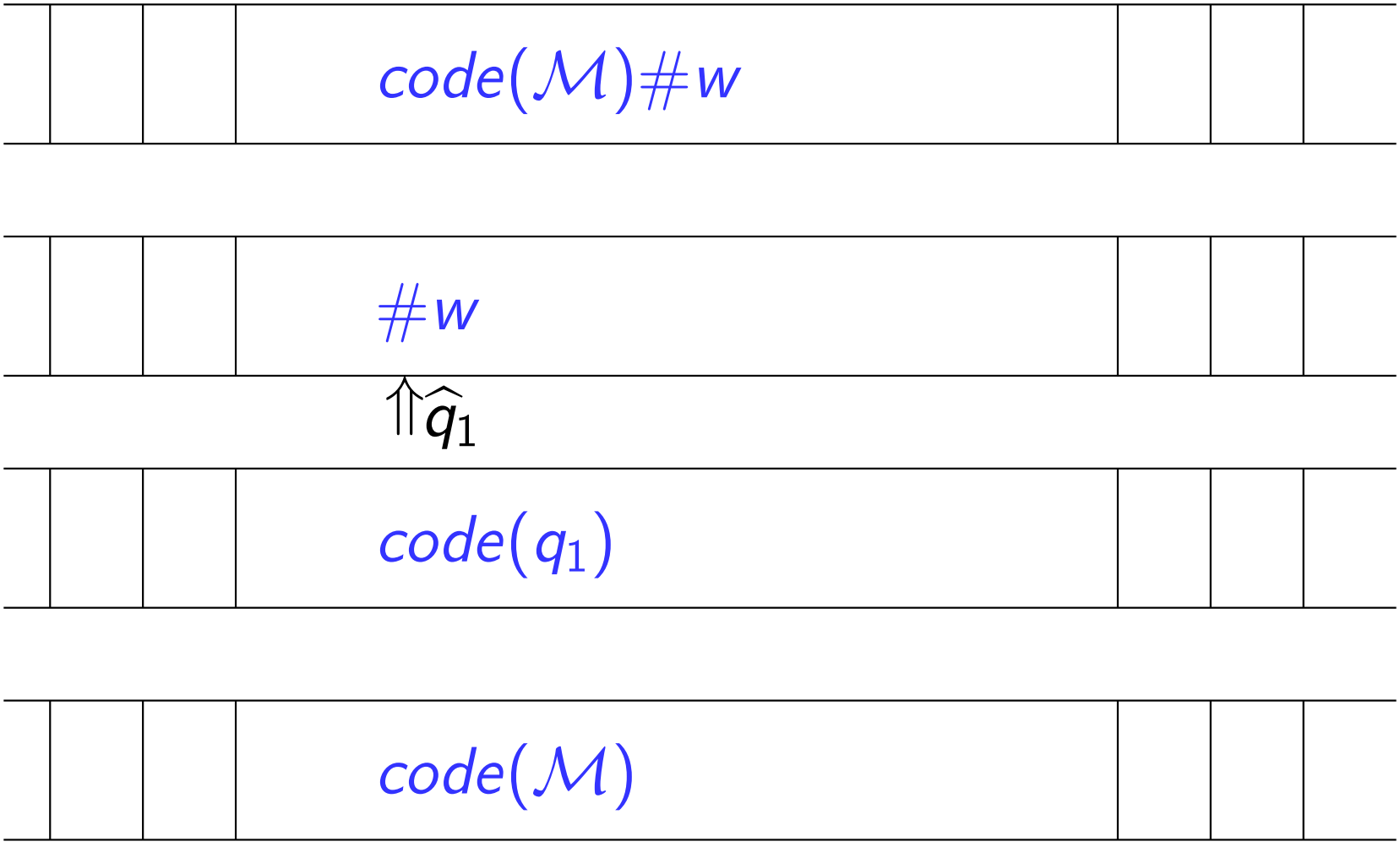
УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Теорема 5.1. Универсальные машины Тьюринга существуют.

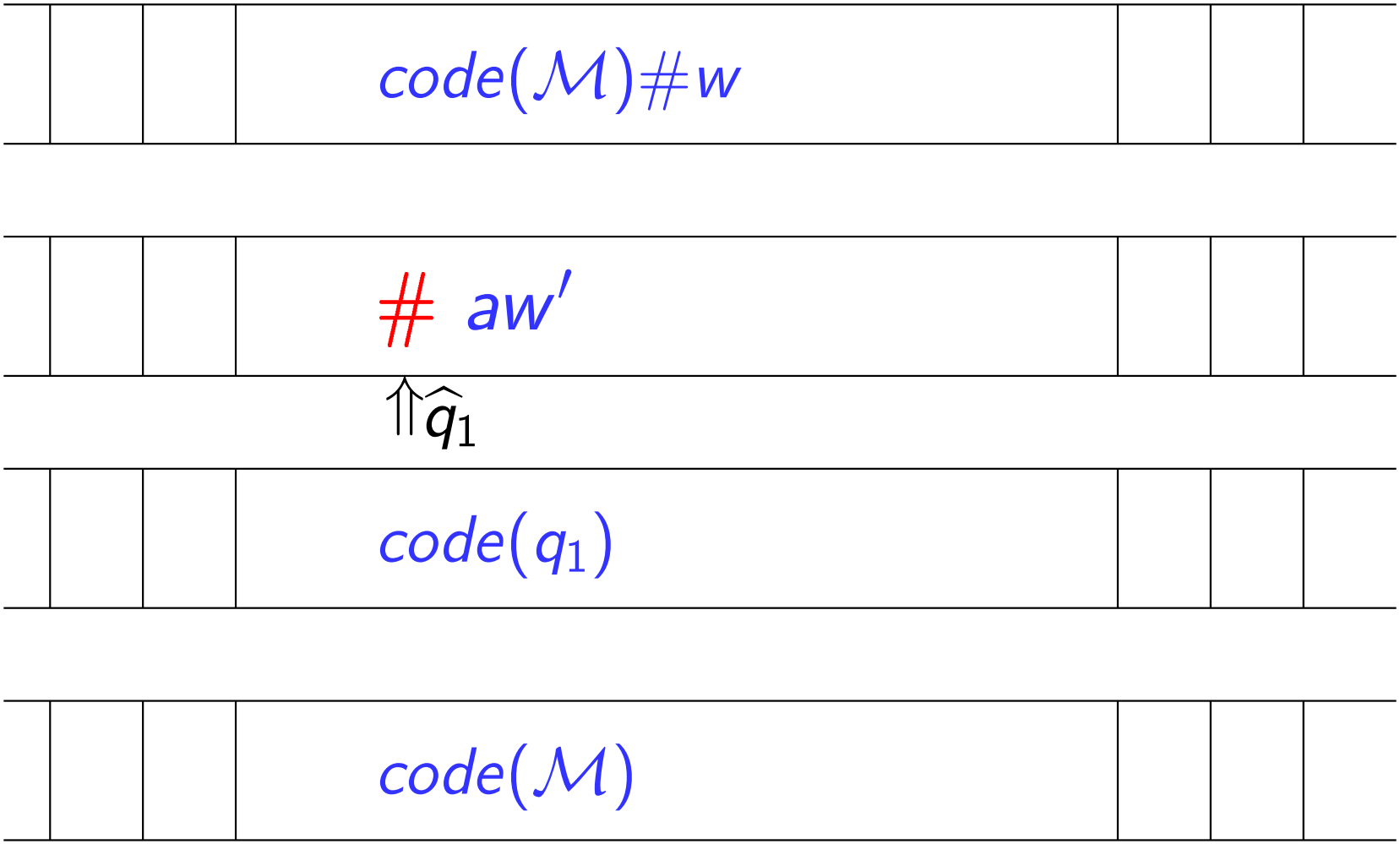
Доказательство. Рассмотрим 4-ленточную МТ $\mathcal{U}$, которая работает по следующему сценарию.



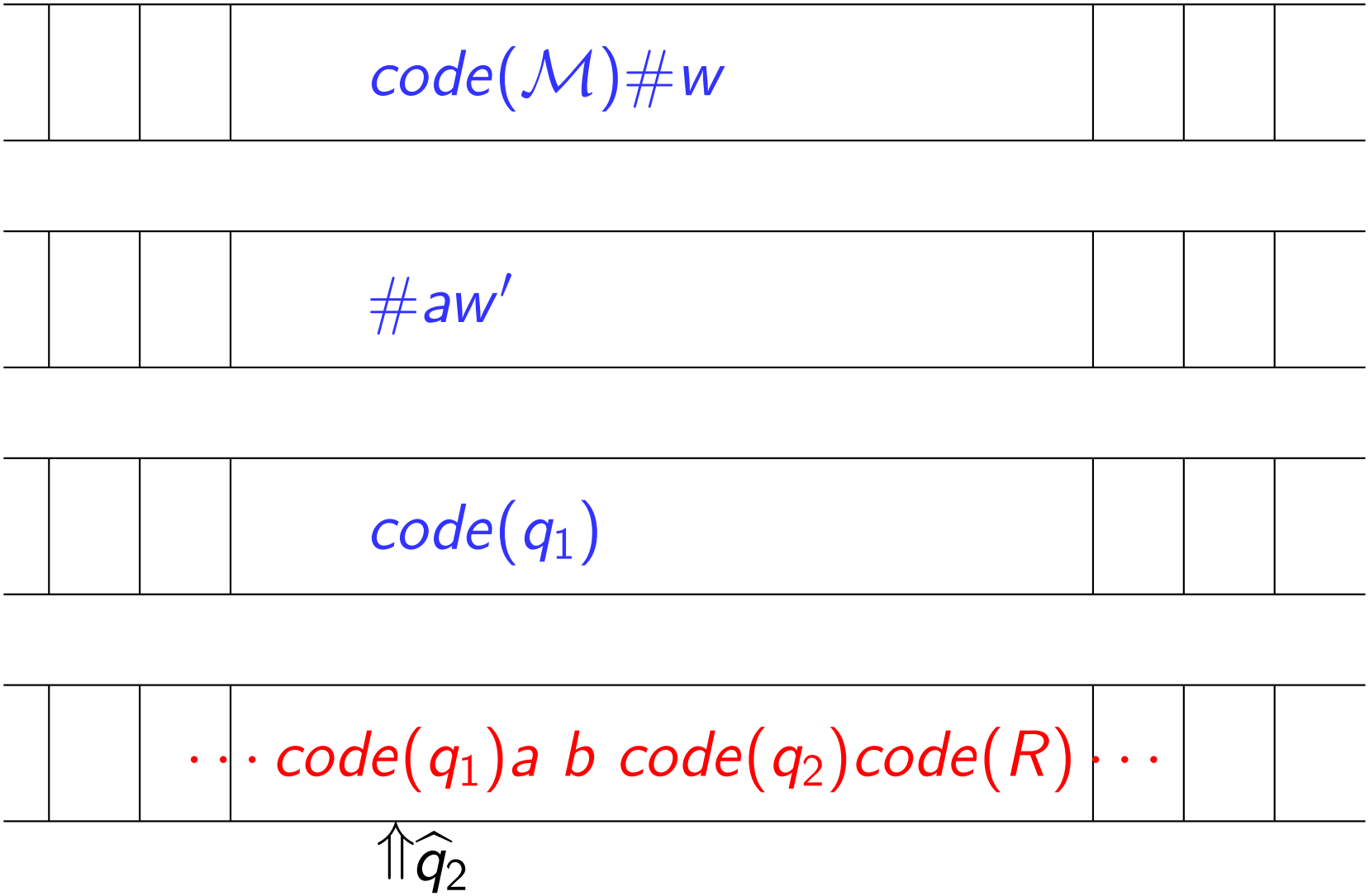
Получив на входной ленте начальную конфигурацию $q_1 code(\mathcal{M})\#w$,



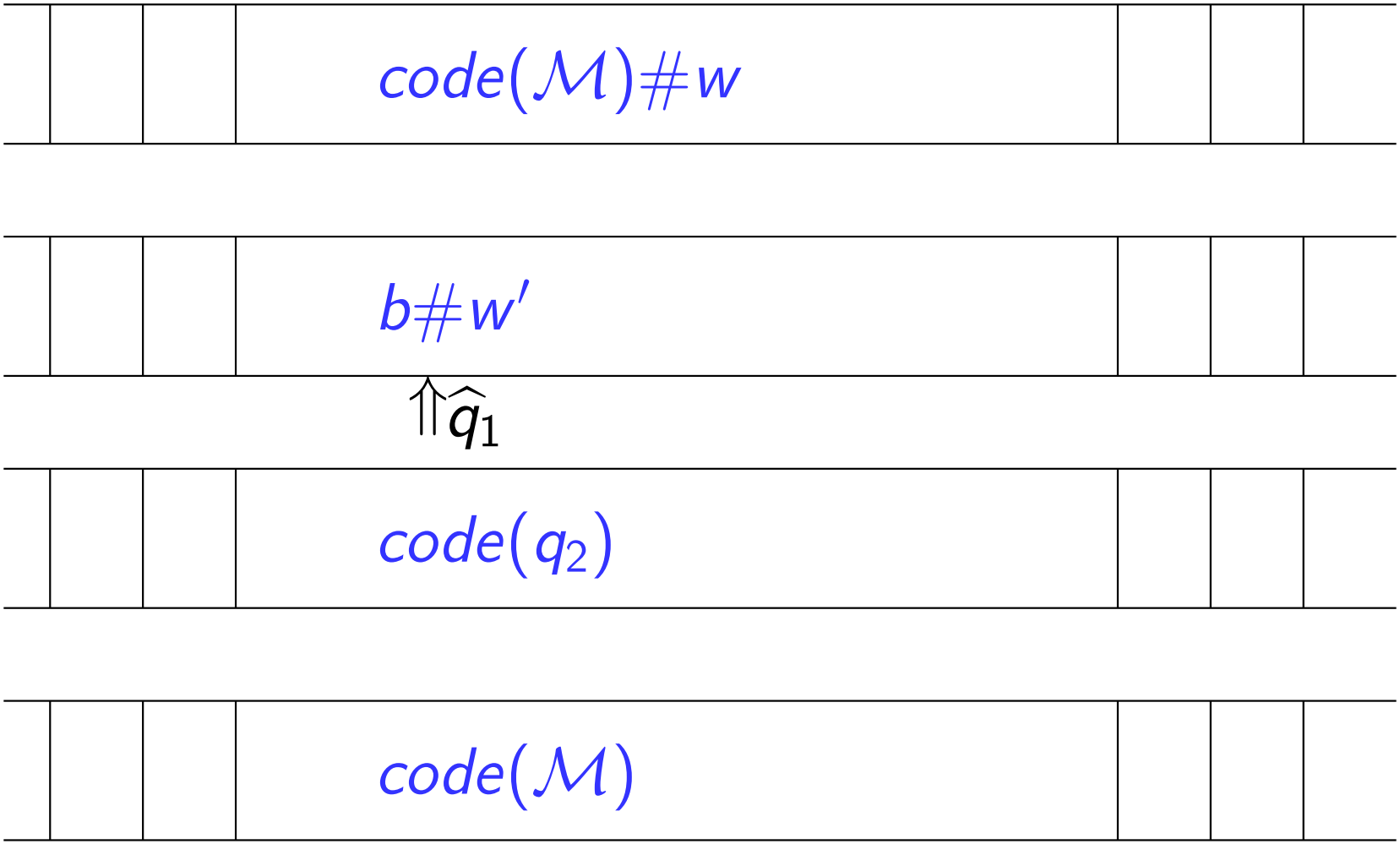
Получив на входной ленте начальную конфигурацию $q_1 code(\mathcal{M})\#w$, МТ $\mathcal{U}$, переписывет данные на рабочие ленты и далее повторяет одни и те же действия:



Используя разделитель $\#$ как маркер обозреваемой ячейки, запоминает букву справа от маркера,



Используя эту букву и запись кода состояния $code(q_1)$, находит код соответствующей команды,



и моделирует эффект применения этой команды.

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Если очередное слово $code(q)$ на второй рабочей ленте оказывается кодом

- допускающего состояния МТ $\mathcal{M}$, то МТ $\mathcal{U}$ также переходит в допускющее состояние;
- отвергающего состояния МТ $\mathcal{M}$, то МТ $\mathcal{U}$ также переходит в отвергающее состояние.

QED

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Если очередное слово $code(q)$ на второй рабочей ленте оказывается кодом

- допускающего состояния МТ $\mathcal{M}$, то МТ $\mathcal{U}$ также переходит в допускющее состояние;
- отвергающего состояния МТ $\mathcal{M}$, то МТ $\mathcal{U}$ также переходит в отвергающее состояние.

QED

А какого размера могут УМТ?

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Если очередное слово $code(q)$ на второй рабочей ленте оказывается кодом

- допускающего состояния МТ $\mathcal{M}$, то МТ $\mathcal{U}$ также переходит в допускющее состояние;
- отвергающего состояния МТ $\mathcal{M}$, то МТ $\mathcal{U}$ также переходит в отвергающее состояние.

QED

А какого размера могут УМТ?

Ю.В. Рогожину удалось построить УМТ с 22 командами [рекорд!]

УНИВЕРСАЛЬНЫЕ МАШИНЫ ТЬЮРИНГА

Вначале о практической пользе УМТ.

УМТ — математическая модель универсального программируемого вычислительного устройства.

Эту модель использовали Том Флауерс, Макс Ньюман и их коллеги при разработке первого программируемого компьютера Colossus, чтобы провести успешный криптоанализ германской системы шифрования **Lorenz**
Эту модель использовал Джон фон Нейман при разработке концепции устройства универсального компьютера.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Обсудим теоретическое значение УМТ.

Покажем, как идеи, лежащие в основе УМТ можно использовать для построения примера нерекурсивного языка.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Обсудим теоретическое значение УМТ.

Покажем, как идеи, лежащие в основе УМТ можно использовать для построения примера нерекурсивного языка.

Назовем МТ $\mathcal{M}$ самоприменимой, если $code(\mathcal{M}) \in L(\mathcal{M})$.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Обсудим теоретическое значение УМТ.

Покажем, как идеи, лежащие в основе УМТ можно использовать для построения примера нерекурсивного языка.

Назовем МТ $\mathcal{M}$ самоприменимой, если $code(\mathcal{M}) \in L(\mathcal{M})$.

Теорема 5.2. Язык

$$S = \{code(\mathcal{M}) : code(\mathcal{M}) \in L(\mathcal{M})\}$$

рекурсивно перечислим, но нерекурсивен.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Доказательство. Рекурсивная перечислимость языка S очевидна.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Доказательство. Рекурсивная перечислимость языка S очевидна.

Предположим, что язык S рекурсивен. Тогда есть тотальная МТ $\mathcal{M}_S$, распознающая S.

Модифицируем МТ $\mathcal{M}_S$: поменяем ролями допускющее состояние q_0 (теперь оно будет отвергающим) и отвергающее состояние q_{-1} (теперь оно будет допускающим).

Образовавшуюся в результате этой замены МТ назовем $\overline{\mathcal{M}_S}$; она также является тотальной.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А теперь зададимся вопросом: принадлежит ли слово $code(\overline{\mathcal{M}_S})$ языку S?

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А теперь зададимся вопросом: принадлежит ли слово $code(\overline{\mathcal{M}_S})$ языку S?

Если $code(\overline{\mathcal{M}_S}) \in S$, то по определению языка S верно $code(\overline{\mathcal{M}_S}) \in L(\overline{\mathcal{M}_S})$.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А теперь зададимся вопросом: принадлежит ли слово $code(\overline{\mathcal{M}_S})$ языку S?

Если $code(\overline{\mathcal{M}_S}) \in S$, то по определению языка S верно $code(\overline{\mathcal{M}_S}) \in L(\overline{\mathcal{M}_S})$.

Но МТ $\overline{\mathcal{M}_S}$ допускает те слова, которые отвергаются МТ $\mathcal{M}_S$. Значит, $code(\overline{\mathcal{M}_S}) \notin L(\mathcal{M}_S)$.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А теперь зададимся вопросом: принадлежит ли слово $code(\overline{M_S})$ языку S?

Если $code(\overline{M_S}) \in S$, то по определению языка S верно $code(\overline{M_S}) \in L(\overline{M_S})$.

Но МТ $\overline{M_S}$ допускает те слова, которые отвергаются МТ M_S. Значит, $code(\overline{M_S}) \notin L(M_S)$.

Но по определению МТ M_S верно $S = L(M_S)$. Отсюда следует, что $code(\overline{M_S}) \notin S$.

Получили противоречие:
$code(\overline{M_S}) \in S \Rightarrow code(\overline{M_S}) \notin S$!

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А что будет, если предположить, что $code(\overline{\mathcal{M}_S}) \notin S$?

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А что будет, если предположить, что $code(\overline{M_S}) \notin S$?

Тогда по определению языка S верно $code(\overline{M_S}) \notin L(\overline{M_S})$.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А что будет, если предположить, что $code(\overline{M_S}) \notin S$?

Тогда по определению языка S верно $code(\overline{M_S}) \notin L(\overline{M_S})$.

Но МТ $\overline{M_S}$ отвергает те слова, которые допускаются МТ M_S. Значит, $code(\overline{M_S}) \in L(M_S)$.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А что будет, если предположить, что $code(\overline{M_S}) \notin S$?

Тогда по определению языка S верно $code(\overline{M_S}) \notin L(\overline{M_S})$.

Но МТ $\overline{M_S}$ отвергает те слова, которые допускаются МТ M_S. Значит, $code(\overline{M_S}) \in L(M_S)$.

Но по определению МТ M_S верно $S = L(M_S)$.
Отсюда следует, что $code(\overline{M_S}) \in S$.

И снова противоречие:
$code(\overline{M_S}) \notin S \Rightarrow code(\overline{M_S}) \in S$!

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Итак, обе возможности $code(\overline{\mathcal{M}_S}) \in S$ и $code(\overline{\mathcal{M}_S}) \notin S$ оказываются несостоятельными.

Что это означает?

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Итак, обе возможности $code(\overline{\mathcal{M}_S}) \in S$ и $code(\overline{\mathcal{M}_S}) \notin S$ оказываются несостоятельными. Что это означает?

Это означает, что такой МТ $\overline{\mathcal{M}_S}$ (а значит, и МТ $\mathcal{M}_S$) существовать не может!

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Итак, обе возможности $code(\overline{\mathcal{M}_S}) \in S$ и $code(\overline{\mathcal{M}_S}) \notin S$ оказываются несостоятельными. Что это означает?

Это означает, что такой МТ $\overline{\mathcal{M}_S}$ (а значит, и МТ $\mathcal{M}_S$) существовать не может!

Но возможностью своего существования МТ $\mathcal{M}_S$ обязана предположению о рекурсивности языка S

.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Итак, обе возможности $code(\overline{\mathcal{M}_S}) \in S$ и $code(\overline{\mathcal{M}_S}) \notin S$ оказываются несостоятельными. Что это означает?

Это означает, что такой МТ $\overline{\mathcal{M}_S}$ (а значит, и МТ $\mathcal{M}_S$) существовать не может!

Но возможностью своего существования МТ $\mathcal{M}_S$ обязана предположению о рекурсивности языка S

.

Значит, язык S нерекурсивен!

QED

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Язык S соответствует массовой алгоритмической проблеме: по описанию МТ выяснить, является ли эта МТ самоприменимой.

Рекурсивная перечислимость языка S означает, что есть алгоритм, проверяющий правильность решения задачи о самоприменимости МТ.

Нерекурсивность языка S означает, что самого алгоритма решения проблемы самоприменимости МТ не существует.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Язык S соответствует массовой алгоритмической проблеме: по описанию МТ выяснить, является ли эта МТ самоприменимой.

Рекурсивная перечислимость языка S означает, что есть алгоритм, проверяющий правильность решения задачи о самоприменимости МТ.

Нерекурсивность языка S означает, что самого алгоритма решения проблемы самоприменимости МТ не существует.

Проблема самоприменимости МТ — это пример задачи, показывающей, что хороший экзаменатор может быть плохим студентом!

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А какой математический парадокс напоминает проблема самоприменимости МТ?

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А какой математический парадокс напоминает проблема самоприменимости МТ?

Парадокс Рассела (задачу о парикмахере): есть ли такой город, в котором парикмахер бреет всех тех жителей города, которые не бреются сами?

Этот парадокс имеет дело с тем же самым противоречием, которое возникает при решении проблемы самоприменимости МТ: а кто бреет самого парикмахера?

Мы увидим, что парадокс Рассела имеет далеко идущие последствия для программирования.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А есть ли такие проблемы, которые не только не имеют алгоритма решения, но даже не имеют алгоритма проверки правильности решений?

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А есть ли такие проблемы, которые не только не имеют алгоритма решения, но даже не имеют алгоритма проверки правильности решений?

Это проблема **НЕсамоприменимости МТ**: выяснить, верно ли,что заданная МТ не допускает свой собственный код.

Почему она не является полуразрешимой?

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

А есть ли такие проблемы, которые не только не имеют алгоритма решения, но даже не имеют алгоритма проверки правильности решений?

Это проблема **НЕсамоприменимости МТ**: выяснить, верно ли,что заданная МТ не допускает свой собственный код.

Почему она не является полуразрешимой?

Потому что полуразрешимость обеих проблем — самоприменимости и несамоприменимости — означала бы, что обе они являются разрешимыми (см. Теорему 4.4).

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Проблема самоприменимости кажется весьма искуственной: ну кому придет в голову подавать на вход программы ее же собственный код?

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Проблема самоприменимости кажется весьма искуственной: ну кому придет в голову подавать на вход программы ее же собственный код?

1) Это не совсем так: например, на вход компилятора можно (и вполне разумно) подавать саму же программу компиляции.

ПРОБЛЕМА САМОПРИМЕНИМОСТИ МАШИН ТЬЮРИНГА

Проблема самоприменимости кажется весьма искуственной: ну кому придет в голову подавать на вход программы ее же собственный код?

1) Это не совсем так: например, на вход компилятора можно (и вполне разумно) подавать саму же программу компиляции.

2) Есть очень много «естественных» задач программирования, для которых невозможно построить алгоритмов их решения. Это можно доказать опираясь на тот факт, что проблема самоприменимости МТ неразрешима.

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Будем говорить, что МТ $\mathcal{M}$ останавливается на пустой ленте, если $\varepsilon \in L(\mathcal{M})$, т.е. вычисление МТ $\mathcal{M}$, начинающееся конфигурацией, в которой на ленте записано пустое входное слово, завершается спустя конечно число тактов работы в допускающем состоянии.

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Будем говорить, что МТ $\mathcal{M}$ останавливается на пустой ленте, если $\varepsilon \in L(\mathcal{M})$, т.е. вычисление МТ $\mathcal{M}$, начинающееся конфигурацией, в которой на ленте записано пустое входное слово, завершается спустя конечно число тактов работы в допускающем состоянии.

Можно ли построить алгоритм проверки свойства останова МТ на пустой ленте?

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Будем говорить, что МТ $\mathcal{M}$ останавливается на пустой ленте, если $\varepsilon \in L(\mathcal{M})$, т.е. вычисление МТ $\mathcal{M}$, начинающееся конфигурацией, в которой на ленте записано пустое входное слово, завершается спустя конечно число тактов работы в допускающем состоянии.

Можно ли построить алгоритм проверки свойства останова МТ на пустой ленте?

Теорема 5.3. Язык $K = \{code(\mathcal{M}) : \varepsilon \in L(\mathcal{M})\}$ нерекурсивен.

ПРОБЛЕМА ОСТАНОВА

Доказательство. Покажем, что из неразрешимости проблемы самоприменимости МТ следует неразрешимость проблемы останова. Доказательство проводится «от противного».

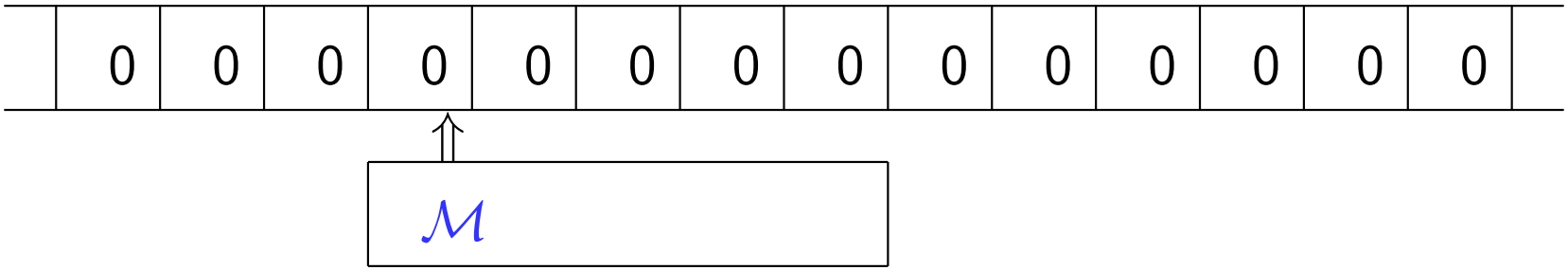
ПРОБЛЕМА ОСТАНОВА

Доказательство. Покажем, что из неразрешимости проблемы самоприменимости МТ следует неразрешимость проблемы останова. Доказательство проводится «от противного».

Допустим, что существует такая тотальная МТ $\mathcal{M}_K$, что $K = L(\mathcal{M}_K)$.

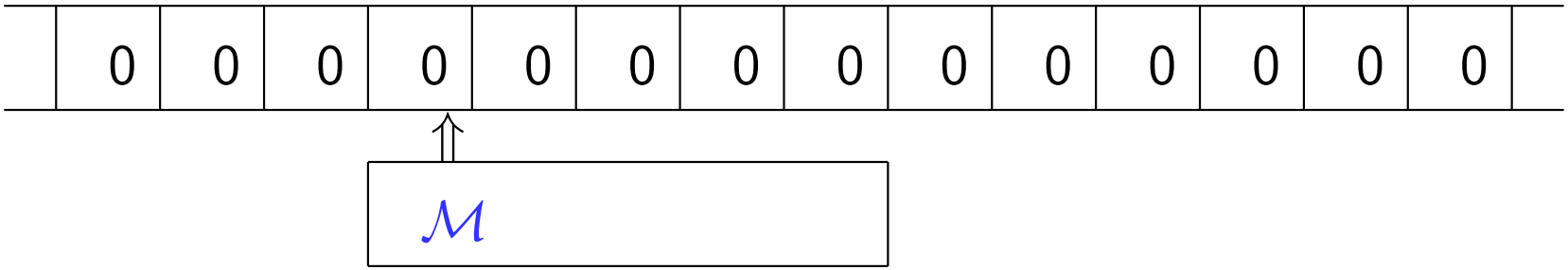
Покажем, как используя МТ $\mathcal{M}_K$, построить тотальную МТ $\mathcal{M}_S$, распознающую язык S, т.е. решающую проблему самоприменимости МТ. Т.к. эта проблема неразрешима, такое построение означает, что МТ $\mathcal{M}_K$, решающую проблему останова построить невозможно.

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

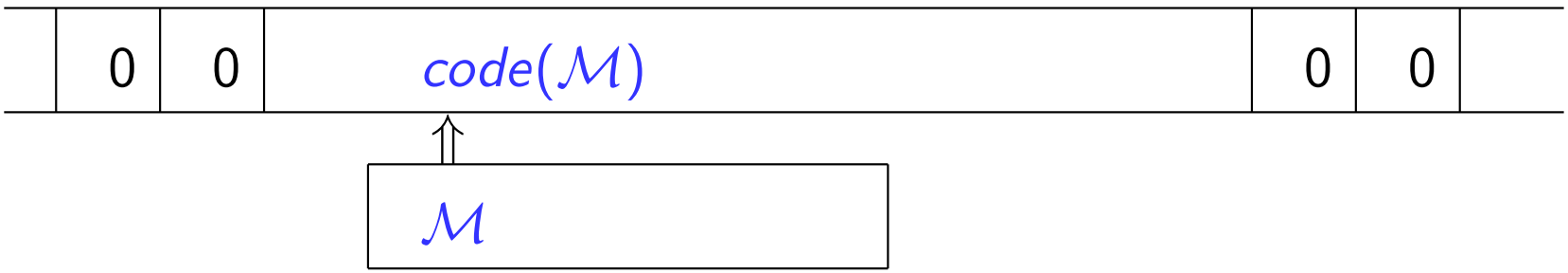


Условие останова МТ: вычисление $\mathcal{M}$ на пустой ленте завершается.

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА



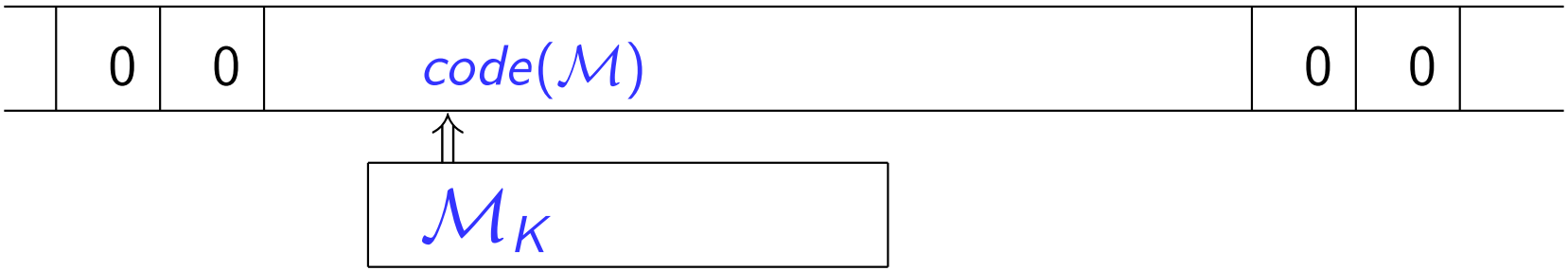
Условие останова МТ: вычисление $\mathcal{M}$ на пустой ленте завершается.



Условие самоприменимости МТ: вычисление $\mathcal{M}$ на входном слове $code(\mathcal{M})$ завершается.

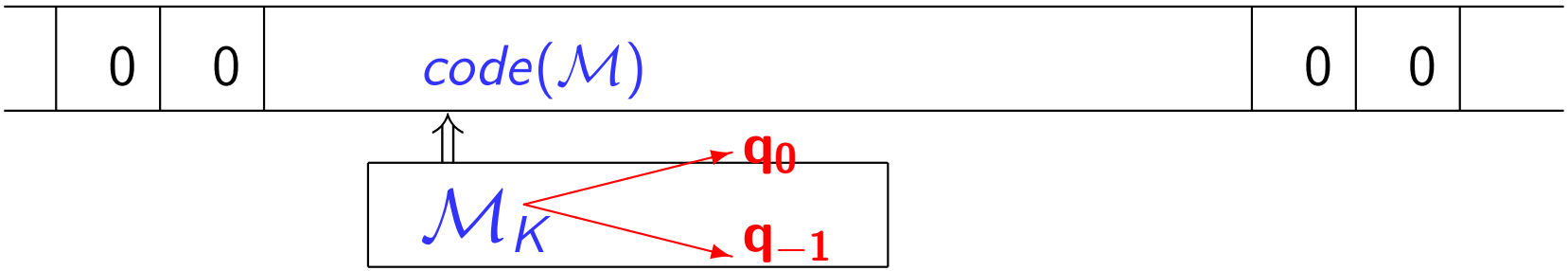
ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Предположим у нас есть МТ $\mathcal{M}_K$, решающая проблему останова на пустой ленте.



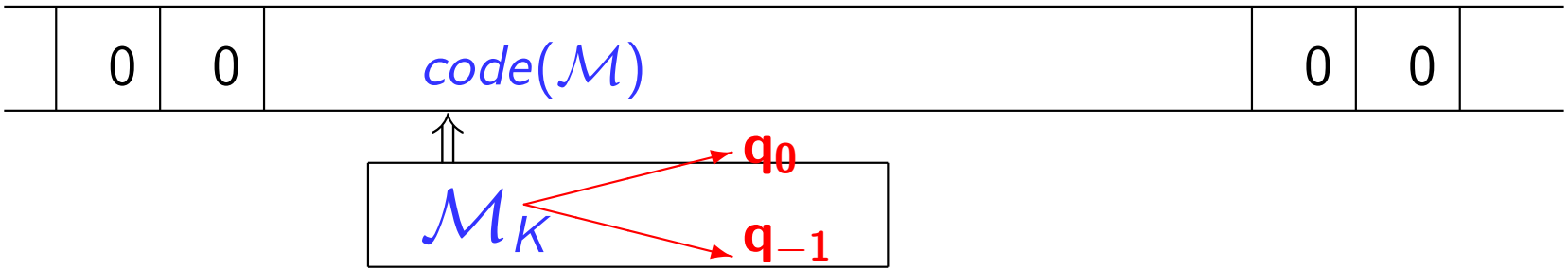
ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Предположим у нас есть МТ $\mathcal{M}_K$, решающая проблему останова на пустой ленте.

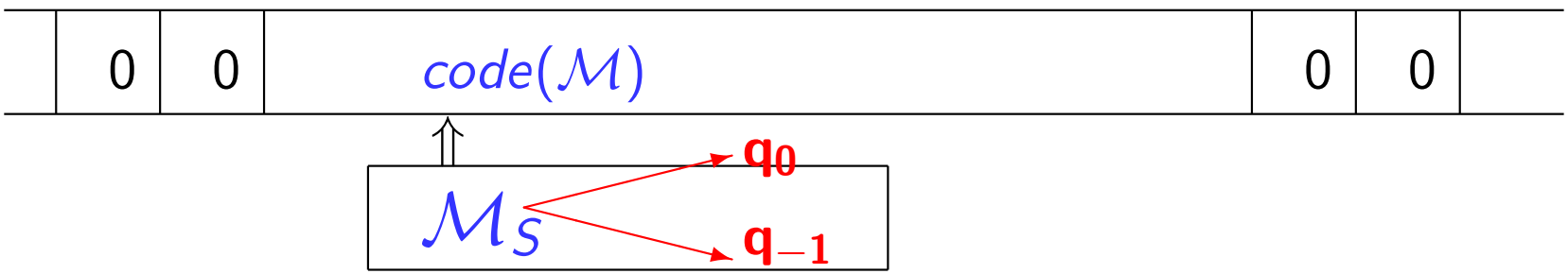


ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

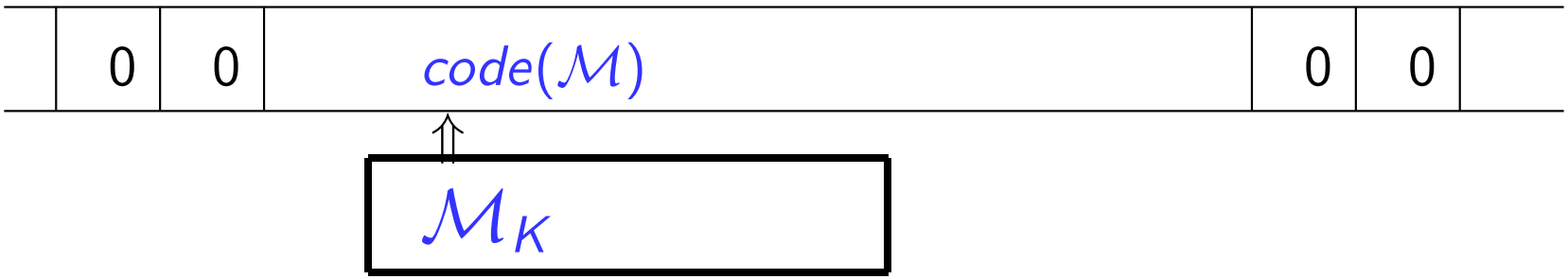
Предположим у нас есть МТ $\mathcal{M}_K$, решающая проблему останова на пустой ленте.



И как нам построить МТ $\mathcal{M}_S$, решающую проблему самоприменимости?



ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА



Заметим, что мы ничего не знаем об устройстве предполагаемой МТ $\mathcal{M}_K$. Мы знаем только ее функциональность: она решает проблему останова на пустой ленте.
Она является для нас «черным ящиком»: мы можем использовать ее функциональность, но не можем вносить изменений в ее устройство.

Как же нам правильно использовать МТ $\mathcal{M}_K$ для проверки самоприменимости МТ?

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Нам надо переформулировать вопрос о самоприменимости заданной МТ так, чтобы на него мог отвечать «черный ящик» $\mathcal{M}_K$, умеющий решать проблему пустоты.

Более конкретно: нужно научиться транслировать программу заданной МТ $\mathcal{M}$ в программу другой МТ $\widehat{\mathcal{M}}$ так, чтобы выполнялось соотношение

$$code(\mathcal{M}) \in S \;\Leftrightarrow\; code(\widehat{\mathcal{M}}) \in K.$$

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Нам надо переформулировать вопрос о самоприменимости заданной МТ так, чтобы на него мог отвечать «черный ящик» $\mathcal{M}_K$, умеющий решать проблему пустоты.

Более конкретно: нужно научиться транслировать программу заданной МТ $\mathcal{M}$ в программу другой МТ $\widehat{\mathcal{M}}$ так, чтобы выполнялось соотношение

$$code(\mathcal{M}) \in S \;\Leftrightarrow\; code(\widehat{\mathcal{M}}) \in K.$$

Если такой транслятор $\mathcal{T}$ удастся построить, то последовательная композиция $\mathcal{T};\mathcal{M}_K$ будет решать проблему самоприменимости МТ!

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Для каждой МТ $\mathcal{M}$ обозначим записью $Write_{\mathcal{M}}$ МТ, которая записывает на пустой ленте код $code(\mathcal{M})$ МТ $\mathcal{M}$.

Покажем, что последовательная композиция МТ $Write_{\mathcal{M}}; \mathcal{M}$ как раз и является желаемым результатом трансляции $\widehat{\mathcal{M}}$.

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Для каждой МТ $\mathcal{M}$ обозначим записью $Write_{\mathcal{M}}$ МТ, которая записывает на пустой ленте код $code(\mathcal{M})$ МТ $\mathcal{M}$.

Покажем, что последовательная композиция МТ $Write_{\mathcal{M}}; \mathcal{M}$ как раз и является желаемым результатом трансляции $\widehat{\mathcal{M}}$.

Действительно, МТ $Write_{\mathcal{M}}; \mathcal{M}$ вначале запишет на пустой входной ленте код $code(\mathcal{M})$, а потом применит МТ $\mathcal{M}$ к этому коду.

Значит, $Write_{\mathcal{M}}; \mathcal{M}$ остановится на пустой входной ленте тогда и только тогда, когда МТ $\mathcal{M}$ самоприменима!

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Осталось построить транслятор $\mathcal{T} : code(\mathcal{M}) \to code(Write_{\mathcal{M}})$. Но это сделать очень просто!

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Осталось построить транслятор $\mathcal{T} : code(\mathcal{M}) \to code(Write_{\mathcal{M}})$. Но это сделать очень просто!

МТ $\mathcal{T}$ прочитывает на входной ленте слово $code(\mathcal{M})$ букву за буквой и для каждой прочитанной буквы записывает на выходной ленте код той команды, которая может записать эту букву:

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Осталось построить транслятор $\mathcal{T} : code(\mathcal{M}) \to code(Write_{\mathcal{M}})$. Но это сделать очень просто!

МТ $\mathcal{T}$ прочитывает на входной ленте слово $code(\mathcal{M})$ букву за буквой и для каждой прочитанной буквы записывает на выходной ленте код той команды, которая может записать эту букву: $\mathcal{T}$:

$$abc$$

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Осталось построить транслятор $\mathcal{T} : code(\mathcal{M}) \to code(Write_{\mathcal{M}})$. Но это сделать очень просто!

МТ $\mathcal{T}$ прочитывает на входной ленте слово $code(\mathcal{M})$ букву за буквой и для каждой прочитанной буквы записывает на выходной ленте код той команды, которая может записать эту букву: $\mathcal{T}$:

$$abc$$

$code(q_1 0 : cq_2 L)$

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Осталось построить транслятор $\mathcal{T} : code(\mathcal{M}) \to code(Write_{\mathcal{M}})$. Но это сделать очень просто!

МТ $\mathcal{T}$ прочитывает на входной ленте слово $code(\mathcal{M})$ букву за буквой и для каждой прочитанной буквы записывает на выходной ленте код той команды, которая может записать эту букву: $\mathcal{T}$:

$$abc$$

$$code(q_1 0 : cq_2 L)code(q_2 0 : bq_3 L)$$

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Осталось построить транслятор $\mathcal{T} : code(\mathcal{M}) \to code(Write_{\mathcal{M}})$. Но это сделать очень просто!

МТ $\mathcal{T}$ прочитывает на входной ленте слово $code(\mathcal{M})$ букву за буквой и для каждой прочитанной буквы записывает на выходной ленте код той команды, которая может записать эту букву: $\mathcal{T}$:

$$abc$$

$$code(q_1 0 : cq_2 L)code(q_2 0 : bq_3 L)code(q_3 0 : bq_4 L)$$

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Осталось построить транслятор $\mathcal{T} : code(\mathcal{M}) \to code(Write_{\mathcal{M}})$. Но это сделать очень просто!

МТ $\mathcal{T}$ прочитывает на входной ленте слово $code(\mathcal{M})$ букву за буквой и для каждой прочитанной буквы записывает на выходной ленте код той команды, которая может записать эту букву: $\mathcal{T}$:

$$abc$$

$$code(q_1 0{:}cq_2 L)code(q_2 0{:}bq_3 L)code(q_3 0{:}bq_4 L)code(q_4 0{:}bq_5 R)$$

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Осталось построить транслятор $\mathcal{T} : code(\mathcal{M}) \to code(Write_{\mathcal{M}})$. Но это сделать очень просто!

МТ $\mathcal{T}$ прочитывает на входной ленте слово $code(\mathcal{M})$ букву за буквой и для каждой прочитанной буквы записывает на выходной ленте код той команды, которая может записать эту букву: $\mathcal{T}$:

$$abc$$

$$code(q_1 0{:}cq_2 L)code(q_2 0{:}bq_3 L)code(q_3 0{:}bq_4 L)code(q_4 0{:}bq_5 R)$$

а потом к этой последовательности кодов команд добавляет код $code(\mathcal{M})$ МТ $\mathcal{M}$.

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Вот и все. Если МТ $\mathcal{T}; \mathcal{M}_K$ применить к входному слову $code(\mathcal{M})$, то транслятор $\mathcal{T}$ запишет на рабочей ленте код МТ $Write_{\mathcal{M}}; \mathcal{M}$, и после этого «черный ящик» $\mathcal{M}_K$ проверит свойство останова для МТ $Write_{\mathcal{M}}; \mathcal{M}$.

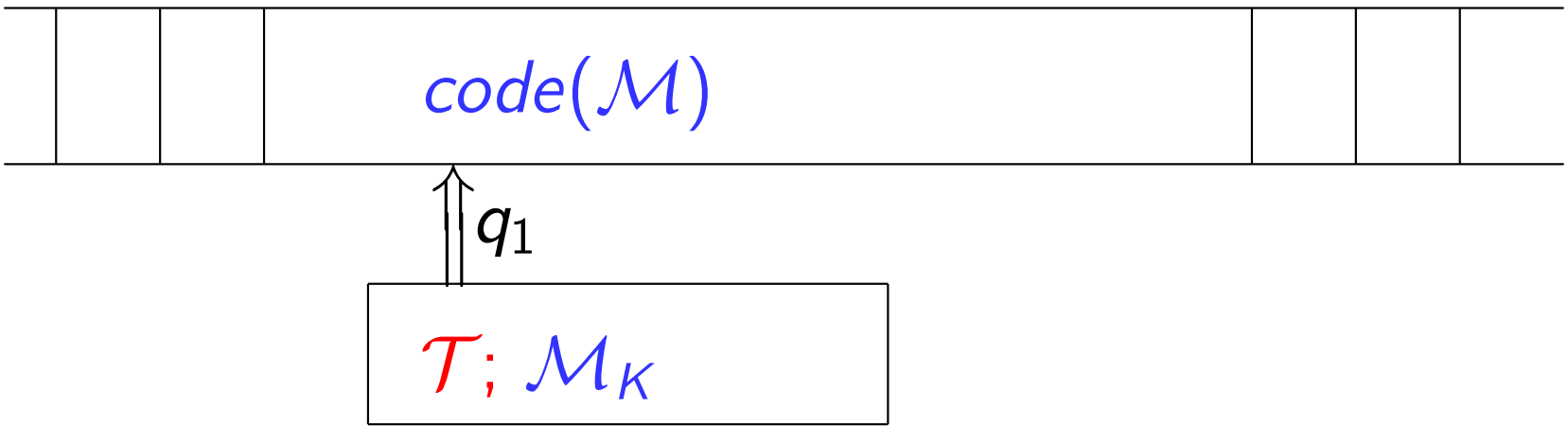
ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА

Вот и все. Если МТ $\mathcal{T}; \mathcal{M}_K$ применить к входному слову $code(\mathcal{M})$, то транслятор $\mathcal{T}$ запишет на рабочей ленте код МТ $Write_{\mathcal{M}}; \mathcal{M}$, и после этого «черный ящик» $\mathcal{M}_K$ проверит свойство останова для МТ $Write_{\mathcal{M}}; \mathcal{M}$.

Поскольку это свойство равносильно свойству самоприменимости МТ $\mathcal{M}$, в результате мы построили тотальную МТ, которая решает проблему самоприменимости, что противоречит Теореме 5.2.

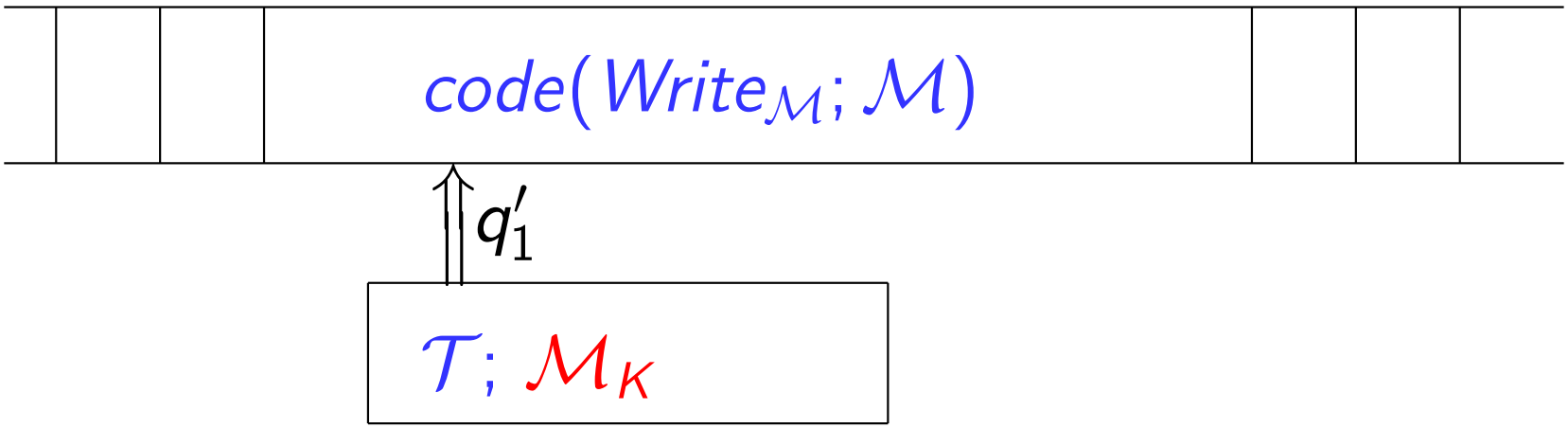
Значит, тотальной МТ, распознающей язык $\mathcal{M}_K$ не существует. QED

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА



Вначале работает МТ $\mathcal{T}$, которая согласно определению записывает на входной ленте слово $code(Write_{\mathcal{M}}; \mathcal{M})$.

ПРОБЛЕМА ОСТАНОВА МАШИН ТЬЮРИНГА



После этого МТ $\mathcal{M}_K$ применяется к входному слову $code(Write_{\mathcal{M}}; \mathcal{M})$, т.е. проверяет, останавливается ли на пустой ленте МТ $Write_{\mathcal{M}}; \mathcal{M}$.

А это равносильно проверке самоприменимости МТ $\mathcal{M}$.

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

В доказательстве Теоремы 5.3 мы построили транслятор, который позволил использовать «черный ящик», решающий одну алгоритмическую проблему, для решения другой проблемы. Этот прием имеет широкое применение.

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

В доказательстве Теоремы 5.3 мы построили транслятор, который позволил использовать «черный ящик», решающий одну алгоритмическую проблему, для решения другой проблемы. Этот прием имеет широкое применение.

Будем говорить, что язык L_1 m-сводится (много-одно сводится) к языку L_2, если существует тотальная T-вычислимая функция $\varphi : \Sigma^* \to \Sigma^*$, удовлетворяющая условию: для любого слова w

$$w \in L_1 \iff \varphi(w) \in L_2.$$

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

МТ, вычисляющая функцию φ, и является тем самым транслятором, который позволяет решать решать одни алгоритмические проблемы при помощи «решателей» других проблем.

Теорема 5.4. Если язык L_2 является рекурсивным (рекурсивно перечислимым), а язык L_1 m-сводится к языку L_2, то язык L_1 также является рекурсивным (рекурсивно перечислимым).

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

МТ, вычисляющая функцию φ, и является тем самым транслятором, который позволяет решать решать одни алгоритмические проблемы при помощи «решателей» других проблем.

Теорема 5.4. Если язык L_2 является рекурсивным (рекурсивно перечислимым), а язык L_1 m-сводится к языку L_2, то язык L_1 также является рекурсивным (рекурсивно перечислимым).

Доказательство. Пусть (тотальная) МТ $\mathcal{M}_2$ распознает язык L_2, а МТ $\mathcal{T}$ вычисляет функцию φ, которая сводит язык L_1 к языку L_2. Тогда (тотальная) МТ $\mathcal{T}; \mathcal{M}_2$ распознает язык L_1. QED

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

Следствие. Если язык L_1 **НЕ** является рекурсивным (рекурсивно перечислимым) и m-сводится к языку L_2, то язык L_2 также **НЕ** является рекурсивным (рекурсивно перечислимым).

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

Следствие. Если язык L_1 **НЕ** является рекурсивным (рекурсивно перечислимым) и m-сводится к языку L_2, то язык L_2 также **НЕ** является рекурсивным (рекурсивно перечислимым).

Так, например, мы воспользовались этим следствием при доказательстве Теоремы 5.3, построив m-сведение языка $S = \{code(\mathcal{M}) : code(\mathcal{M}) \in L(\mathcal{M})\}$ к языку $K = \{code(\mathcal{M}) : \varepsilon \in L(\mathcal{M})\}$.

И точно так же можно показать алгоритмическую неразрешимость многих других ключевых задач программирования.

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

Утверждение 5.5. Язык $Tot = \{code(\mathcal{M}) : L(\mathcal{M}) = \Sigma^*\}$ (проблема тотальности MT) нерекурсивен.

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

Утверждение 5.5. Язык $Tot = \{code(\mathcal{M}) : L(\mathcal{M}) = \Sigma^*\}$ (проблема тотальности МТ) нерекурсивен.

Доказательство. Чтобы проверить самоприменимость МТ $\mathcal{M}$ необходимо и достаточно проверить тотальность МТ

$$Erase; Write_{\mathcal{M}}; \mathcal{M},$$

где $Erase$ — это МТ, которая стирает входное слово. QED

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

Утверждение 5.6. Язык
$EQ = \{code(\mathcal{M}_1)\#code(\mathcal{M}_2) : L(\mathcal{M}_1) = L(\mathcal{M}_2)\}$
(проблема эквивалентности МТ) нерекурсивен.

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

Утверждение 5.6. Язык
$EQ = \{code(\mathcal{M}_1)\#code(\mathcal{M}_2) : L(\mathcal{M}_1) = L(\mathcal{M}_2)\}$
(проблема эквивалентности МТ) нерекурсивен.

Доказательство. Чтобы проверить тотальность МТ $\mathcal{M}$ необходимо и достаточно проверить эквивалентность МТ $\mathcal{M}$ и МТ $\mathcal{M}_0$, программа которой состоит только из команд $(q_1x : xq_0L),\ x \in \Gamma$. QED

СВОДИМОСТЬ АЛГОРИТМИЧЕСКИХ ПРОБЛЕМ

Утверждение 5.6. Язык
$EQ = \{code(\mathcal{M}_1)\#code(\mathcal{M}_2) : L(\mathcal{M}_1) = L(\mathcal{M}_2)\}$
(проблема эквивалентности МТ) нерекурсивен.

Доказательство. Чтобы проверить тотальность МТ $\mathcal{M}$ необходимо и достаточно проверить эквивалентность МТ $\mathcal{M}$ и МТ $\mathcal{M}_0$, программа которой состоит только из команд $(q_1x : xq_0L),\ x \in \Gamma$. QED

А нельзя ли все подобные алгоритмические проблемы, касающиеся свойств вычислений МТ, решить раз и навсегда «в общем виде»?

ТЕОРЕМА РАЙСА

МОЖНО!!!

Рассмотрим множество всех слов, являющихся кодами МТ: $CODE = \{code(\mathcal{M}) : \mathcal{M} - \text{это МТ}\}$.

Будем называть язык $Prop$, $Prop \subseteq CODE$ семантическим свойством МТ (относительно кодирования $code$ МТ), если этот язык удовлетворяет следующему требованию:

$$\forall \mathcal{M}_1, \mathcal{M}_2 (L(\mathcal{M}_1) = L(\mathcal{M}_2) \wedge code(\mathcal{M}_1) \in Prop \Rightarrow code(\mathcal{M}_1) \in Prop)$$

ТЕОРЕМА РАЙСА

МОЖНО!!!

Рассмотрим множество всех слов, являющихся кодами МТ: $CODE = \{code(\mathcal{M}) : \mathcal{M} - \text{это МТ}\}$.

Будем называть язык $Prop$, $Prop \subseteq CODE$ семантическим свойством МТ (относительно кодирования $code$ МТ), если этот язык удовлетворяет следующему требованию:

$$\forall \mathcal{M}_1, \mathcal{M}_2 (L(\mathcal{M}_1) = L(\mathcal{M}_2) \wedge code(\mathcal{M}_1) \in Prop \Rightarrow code(\mathcal{M}_1) \in Prop)$$

Свойство МТ $Prop$, $Prop \subseteq CODE$, называется нетривиальным, если $Prop \neq \varnothing$ и $Prop \neq CODE$.

ТЕОРЕМА РАЙСА

Например,

- $K = \{code(\mathcal{M}) : \varepsilon \in L(\mathcal{M})\}$ (проблема останова на пустой ленте) — нетривиальное семантическое свойство;

ТЕОРЕМА РАЙСА

Например,

- $K = \{code(\mathcal{M}) : \varepsilon \in L(\mathcal{M})\}$ (проблема останова на пустой ленте) — нетривиальное семантическое свойство;
- $K = \{code(\mathcal{M}) : |L(\mathcal{M})| = 99\}$ — нетривиальное семантическое свойство;

ТЕОРЕМА РАЙСА

Например,

- $K = \{code(\mathcal{M}) : \varepsilon \in L(\mathcal{M})\}$ (проблема останова на пустой ленте) — нетривиальное семантическое свойство;
- $K = \{code(\mathcal{M}) : |L(\mathcal{M})| = 99\}$ — нетривиальное семантическое свойство;
- $\varnothing$ — тривиальное семантическое свойство;

ТЕОРЕМА РАЙСА

Например,

- $K = \{code(\mathcal{M}) : \varepsilon \in L(\mathcal{M})\}$ (проблема останова на пустой ленте) — нетривиальное семантическое свойство;
- $K = \{code(\mathcal{M}) : |L(\mathcal{M})| = 99\}$ — нетривиальное семантическое свойство;
- $\varnothing$ — тривиальное семантическое свойство;
- $\{code(\mathcal{M}) : |code(\mathcal{M})| = 99\}$ — нетривиальное несемантическое свойство;

ТЕОРЕМА РАЙСА

Например,

- $K = \{code(\mathcal{M}) : \varepsilon \in L(\mathcal{M})\}$ (проблема останова на пустой ленте) — нетривиальное семантическое свойство;
- $K = \{code(\mathcal{M}) : |L(\mathcal{M})| = 99\}$ — нетривиальное семантическое свойство;
- $\varnothing$ — тривиальное семантическое свойство;
- $\{code(\mathcal{M}) : |code(\mathcal{M})| = 99\}$ — нетривиальное несемантическое свойство;
- $S = \{code(\mathcal{M}) : code(\mathcal{M}) \in L(\mathcal{M})\}$ — нетривиальное несемантическое свойство.

ТЕОРЕМА РАЙСА

Теорема 5.7. [Райса] Любое нетривиальное семантическое свойство МТ нерекурсивно.

ТЕОРЕМА РАЙСА

Теорема 5.7. [Райса] Любое нетривиальное семантическое свойство МТ нерекурсивно.

Доказательство. Пусть $Prop$ — нетривиальное семантическое свойство. Возьмем любую такую МТ $\mathcal{M}_0$, что $L(\mathcal{M}_0) = \varnothing$. Будем полагать (для определенности), что $code(\mathcal{M}_0) \notin Prop$. В случае $code(\mathcal{M}_0) \in Prop$ все последующие рассуждения проводятся аналогично.

ТЕОРЕМА РАЙСА

Теорема 5.7. [Райса] Любое нетривиальное семантическое свойство МТ нерекурсивно.

Доказательство. Пусть $Prop$ — нетривиальное семантическое свойство. Возьмем любую такую МТ $\mathcal{M}_0$, что $L(\mathcal{M}_0) = \varnothing$. Будем полагать (для определенности), что $code(\mathcal{M}_0) \notin Prop$. В случае $code(\mathcal{M}_0) \in Prop$ все последующие рассуждения проводятся аналогично.

Поскольку $Prop$ — нетривиальное свойство МТ, существует и такая МТ $\mathcal{M}_1$, что $code(\mathcal{M}_1) \in Prop$.

ТЕОРЕМА РАЙСА

Покажем, что язык K (проблема останова МТ на пустой ленте) m-сводим к языку $Prop$.

ТЕОРЕМА РАЙСА

Покажем, что язык K (проблема останова МТ на пустой ленте) m-сводим к языку $Prop$.

Чтобы проверить включение $code(\mathcal{M}) \in K$ для произвольной МТ $\mathcal{M}$ транслируем ее в МТ $\widehat{\mathcal{M}}$, которая на всяком входном слове w работает так:

$$\widehat{\mathcal{M}}(q_1 w 0) = \textbf{ if } \varepsilon \in L(\mathcal{M}) \textbf{ then } \mathcal{M}_1(q_1 w 0) \textbf{ else } loop.$$

ТЕОРЕМА РАЙСА

Покажем, что язык K (проблема останова МТ на пустой ленте) m-сводим к языку $Prop$.

Чтобы проверить включение $code(\mathcal{M}) \in K$ для произвольной МТ $\mathcal{M}$ транслируем ее в МТ $\widehat{\mathcal{M}}$, которая на всяком входном слове w работает так:

$$\widehat{\mathcal{M}}(q_1 w 0) = \textbf{ if } \varepsilon \in L(\mathcal{M}) \textbf{ then } \mathcal{M}_1(q_1 w 0) \textbf{ else } loop.$$

Из описания МТ $\widehat{\mathcal{M}}$ видно, что

- если $code(\mathcal{M}) \in K$, то МТ $\widehat{\mathcal{M}}$ на любом входном слове w работает как МТ $\mathcal{M}_1$; в этом случае $L(\widehat{\mathcal{M}}) = L(\mathcal{M}_1)$;

ТЕОРЕМА РАЙСА

Покажем, что язык K (проблема останова МТ на пустой ленте) m-сводим к языку $Prop$.

Чтобы проверить включение $code(\mathcal{M}) \in K$ для произвольной МТ $\mathcal{M}$ транслируем ее в МТ $\widehat{\mathcal{M}}$, которая на всяком входном слове w работает так:

$$\widehat{\mathcal{M}}(q_1 w 0) = \textbf{ if } \varepsilon \in L(\mathcal{M}) \textbf{ then } \mathcal{M}_1(q_1 w 0) \textbf{ else } loop.$$

Из описания МТ $\widehat{\mathcal{M}}$ видно, что

- если $code(\mathcal{M}) \in K$, то МТ $\widehat{\mathcal{M}}$ на любом входном слове w работает как МТ $\mathcal{M}_1$; в этом случае $L(\widehat{\mathcal{M}}) = L(\mathcal{M}_1)$;
- если $code(\mathcal{M}) \notin K$, то на любом входном слове w вычисление МТ $\widehat{\mathcal{M}}$ зацикливается; поэтому $L(\widehat{\mathcal{M}}) = L(\mathcal{M}_0)$.

ТЕОРЕМА РАЙСА

Покажем, что язык K (проблема останова МТ на пустой ленте) m-сводим к языку $Prop$.

Чтобы проверить включение $code(\mathcal{M}) \in K$ для произвольной МТ $\mathcal{M}$ транслируем ее в МТ $\widehat{\mathcal{M}}$, которая на всяком входном слове w работает так:

$$\widehat{\mathcal{M}}(q_1 w 0) = \textbf{ if } \varepsilon \in L(\mathcal{M}) \textbf{ then } \mathcal{M}_1(q_1 w 0) \textbf{ else } loop.$$

Из описания МТ $\widehat{\mathcal{M}}$ видно, что

- если $code(\mathcal{M}) \in K$, то МТ $\widehat{\mathcal{M}}$ на любом входном слове w работает как МТ $\mathcal{M}_1$; в этом случае $L(\widehat{\mathcal{M}}) = L(\mathcal{M}_1)$;
- если $code(\mathcal{M}) \notin K$, то на любом входном слове w вычисление МТ $\widehat{\mathcal{M}}$ зацикливается; поэтому $L(\widehat{\mathcal{M}}) = L(\mathcal{M}_0)$.

Поскольку свойство МТ $Prop$ — семантическое и $code(\mathcal{M}_0) \notin Prop$, $code(\mathcal{M}_1) \in Prop$, приходим к выводу

$$code(\mathcal{M}) \in K \Leftrightarrow code(\widehat{\mathcal{M}}) \in Prop$$

QED.

ТЕОРЕМА РАЙСА

Из теоремы Райса следует, что ВСЕ задачи анализа поведения компьютерных программ, включая

- проверку завершаемости (отсутствия зацикливания);
- проверку корректности результата;
- оптимизацию;
- проверку отстутствия утечки памяти;
- проверку отсутствия недекларируемых возможностей;

и многие-многие-многие другие,

алгоритмически неразрешимы.

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

Есть очень важный аспект, о котором часто забывают при изучении алгоритмически неразрешимых свойств поведения МТ — кодировании МТ.

Рассмотренное нами кодирование $code(\mathcal{M})$ простое и разумное, но не единственно возможное.

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

Есть очень важный аспект, о котором часто забывают при изучении алгоритмически неразрешимых свойств поведения МТ — кодировании МТ.

Рассмотренное нами кодирование $code(\mathcal{M})$ простое и разумное, но не единственно возможное.

Можно взять другое кодирование $code'$, которое описывет каждую МТ $\mathcal{M}$ словом $code(\mathcal{M})\delta$, где $\delta = 1$ если МТ $\mathcal{M}$ останавливается на пустой ленте; в противном случае $\delta = 0$.

Тогда нетривиальное семантическое свойство $K' = \{code'(\mathcal{M}) : \varepsilon \in L(\mathcal{M})\}$, очевидно, рекурсивно! Вопреки теореме Райса!!! Почему?

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

Оказывается, теореме Райса верна не для всякого кодирования МТ.

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

Оказывается, теореме Райса верна не для всякого кодирования МТ.

Кодирование машин Тьюринга enc называется

- вычислимым, если для этого кодирования можно построить универсальную машину Тьюринга U_{enc}:
  $w \in L(\mathcal{M}) \Leftrightarrow w\#enc(\mathcal{M}) \in L(U_{enc})$;

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

Оказывается, теореме Райса верна не для всякого кодирования МТ.

Кодирование машин Тьюринга enc называется

- вычислимым, если для этого кодирования можно построить универсальную машину Тьюринга U_{enc}:
  $w \in L(\mathcal{M}) \Leftrightarrow w\#enc(\mathcal{M}) \in L(U_{enc})$;
- главным, если для любого вычислимого кодирования enc' существует такая тотальная T-вычислимая функция $\psi : \Sigma^* \to \Sigma^*$, что для любой МТ $\mathcal{M}$
  $w \in L(\mathcal{M}) \Leftrightarrow w\#\psi(enc'(\mathcal{M})) \in L(U_{enc})$;

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

То же самое в терминах простого программирования.

- кодирование МТ — это система программирования;

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

То же самое в терминах простого программирования.

- кодирование МТ — это система программирования;
- вычислимое кодирование — это система программирования, для которой можно построить аппаратный интерпретатор (арифметико-логическое устройство, процессор);

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

То же самое в терминах простого программирования.

- кодирование МТ — это система программирования;
- вычислимое кодирование — это система программирования, для которой можно построить аппаратный интерпретатор (арифметико-логическое устройство, процессор);
- главное кодирование — это система программирования, в которую можно транслировать любую другую систему программирования, имеющую интерпретатор.

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

Оказывается, теореме Райса верна только для **главных вычислимых** кодирований МТ.

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

Оказывается, теореме Райса верна только для **главных вычислимых** кодирований МТ.

Например, кодирование $code(\mathcal{M})$ является вычислимым и главным.

А кодирование $code'(\mathcal{M}) = code(\mathcal{M})\delta$ является вычислимым, но неглавным.

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

Оказывается, теореме Райса верна только для **главных вычислимых** кодирований МТ.

Например, кодирование $code(\mathcal{M})$ является вычислимым и главным.

А кодирование $code'(\mathcal{M}) = code(\mathcal{M})\delta$ является вычислимым, но неглавным.

Все «естественные системы» программирования соответствуют главным вычислимым кодированиям.

КОДИРОВАНИЯ МАШИН ТЬЮРИНГА

Задача 5.1. [Трудная] Где в доказательстве теоремы Райса неявно используется то обстоятельство, что $code(\mathcal{M})$ является главным вычислимым кодирование МТ?

Задача 5.2. Докажите, что кодирование $code'(\mathcal{M}) = code(\mathcal{M})\delta$ является вычислимым, но неглавным?

КОНЕЦ ЛЕКЦИИ 5