**Теория автоматов и формальных языков**

**1. Формальные языки и грамматики. Классификация по Хомскому.**

**Формальный язык** – множество строк (цепочек) конечной длины, составленных из символов некоторого алфавита.

**Формальная грамматика** G = (V, Σ, P, S), где: - V – множество нетерминальных символов - Σ – алфавит терминальных символов (V ∩ Σ = ∅) - P – конечное множество правил вывода вида α → β - S – начальный символ (S ∈ V)

**Классификация грамматик по Хомскому**:

**Тип 0** (неограниченные грамматики):

Правила вида α → β, где α содержит хотя бы один нетерминал

Распознаются машинами Тьюринга

Порождают рекурсивно перечислимые языки

**Тип 1** (контекстно-зависимые грамматики):

Правила вида αAβ → αγβ, где A – нетерминал, γ ≠ ε

Распознаются линейно-ограниченными автоматами

Порождают контекстно-зависимые языки

**Тип 2** (контекстно-свободные грамматики):

Правила вида A → γ, где A – нетерминал

Распознаются автоматами с магазинной памятью

Порождают контекстно-свободные языки

**Тип 3** (регулярные грамматики):

Правила вида A → aB или A → a, где A, B – нетерминалы, a – терминал

Распознаются конечными автоматами

Порождают регулярные языки

Иерархия Хомского: Тип 3 ⊂ Тип 2 ⊂ Тип 1 ⊂ Тип 0

**2. Понятие конечного автомата. Существование детерминированного КА, эквивалентного недетерминированному.**

**Конечный автомат (КА)** – абстрактная модель устройства с конечным числом состояний.

**Детерминированный конечный автомат (ДКА)** M = (Q, Σ, δ, q₀, F), где: - Q – конечное множество состояний - Σ – конечный алфавит символов - δ: Q × Σ → Q – функция переходов - q₀ ∈ Q – начальное состояние - F ⊆ Q – множество заключительных (принимающих) состояний

**Недетерминированный конечный автомат (НКА)** M = (Q, Σ, δ, q₀, F), где: - δ: Q × Σ → 2^Q – функция переходов, возвращающая множество возможных состояний - Остальные компоненты как у ДКА

**НКА с ε-переходами** допускает переходы без чтения символов входной строки.

**Теорема о существовании эквивалентного ДКА**: Для любого НКА существует эквивалентный ему ДКА.

**Алгоритм построения ДКА по НКА** (детерминизация): 1. Состояния ДКА – подмножества состояний НКА 2. Начальное состояние ДКА – ε-замыкание начального состояния НКА 3. Для каждого состояния ДКА и каждого символа алфавита определить новое состояние как объединение переходов из всех состояний соответствующего подмножества 4. Заключительные состояния ДКА – подмножества, содержащие хотя бы одно заключительное состояние НКА

**3. Понятие конечно-автоматного языка. Замкнутость относительно операций над языками.**

**Конечно-автоматный язык** (регулярный язык) – язык, распознаваемый конечным автоматом.

Язык L считается распознаваемым конечным автоматом M, если L = L(M) = {w ∈ Σ\* | δ*(q₀, w) ∈ F}, где δ* – расширенная функция переходов.

**Замкнутость регулярных языков относительно операций**:

**Объединение**: Если L₁ и L₂ – регулярные языки, то L₁ ∪ L₂ – регулярный язык.

Построение: создать новое начальное состояние с ε-переходами в начальные состояния автоматов для L₁ и L₂.

**Конкатенация**: Если L₁ и L₂ – регулярные языки, то L₁L₂ – регулярный язык.

Построение: добавить ε-переходы из заключительных состояний автомата для L₁ в начальное состояние автомата для L₂.

**Итерация (звезда Клини)**: Если L – регулярный язык, то L\* – регулярный язык.

Построение: добавить новое начальное/заключительное состояние с ε-переходами в старое начальное состояние и из заключительных состояний обратно в начальное.

**Пересечение**: Если L₁ и L₂ – регулярные языки, то L₁ ∩ L₂ – регулярный язык.

Построение: создать автомат с состояниями-парами (q₁, q₂), где q₁ из автомата для L₁, q₂ из автомата для L₂.

**Дополнение**: Если L – регулярный язык, то Σ\*  L – регулярный язык.

Построение: детерминизировать автомат, сделать его полным и поменять местами заключительные и незаключительные состояния.

**4. Контекстно-свободные грамматики и языки. Преобразования КС-грамматик.**

**Контекстно-свободная грамматика (КСГ)** содержит правила вида A → α, где A – нетерминал, α – произвольная строка терминалов и нетерминалов.

**Контекстно-свободный язык (КСЯ)** – язык, порождаемый контекстно-свободной грамматикой.

**Преобразования КСГ**:

**Устранение бесполезных символов**:

Удаление непродуктивных нетерминалов (не порождающих терминальные строки)

Удаление недостижимых символов (не встречающихся в выводах из начального символа)

**Устранение ε-правил** (A → ε):

Найти множество E нетерминалов, порождающих ε

Для каждого правила вида B → αAβ, где A ∈ E, добавить правило B → αβ

Удалить правила A → ε, если A ≠ S (начальный символ)

**Устранение цепных правил** (A → B, где B – нетерминал):

Для каждого цепного правила A → B и всех правил B → γ добавить правило A → γ

Удалить цепные правила

**Нормальная форма Хомского** (НФХ):

Каждое правило имеет вид A → BC или A → a, где A, B, C – нетерминалы, a – терминал

Алгоритм преобразования: заменить терминалы в правой части на новые нетерминалы, затем длинные правые части разбить на пары нетерминалов

**Нормальная форма Грейбах**:

Каждое правило имеет вид A → aα, где a – терминал, α – строка нетерминалов

Алгоритм преобразования: сначала привести к НФХ, затем устранить левую рекурсию

**5. Автомат с магазинной памятью и его инструкции. Связь с КС-грамматиками.**

**Автомат с магазинной памятью (АМП)** M = (Q, Σ, Γ, δ, q₀, Z₀, F), где: - Q – конечное множество состояний - Σ – входной алфавит - Γ – алфавит магазина - δ: Q × (Σ ∪ {ε}) × Γ → 2^(Q × Γ\*) – функция переходов - q₀ ∈ Q – начальное состояние - Z₀ ∈ Γ – начальный символ магазина - F ⊆ Q – множество заключительных состояний

**Инструкции АМП**: - δ(q, a, Z) = {(p, γ)} означает: “В состоянии q, читая символ a, с верхним символом магазина Z, перейти в состояние p, заменив Z на строку γ” - Если γ = ε, то происходит удаление верхнего символа (pop) - Если |γ| > 1, то происходит замена верхнего символа на несколько символов (push) - Если a = ε, то происходит ε-переход без чтения входного символа

**Принятие входной строки**: - По завершению чтения входной строки автомат находится в заключительном состоянии (принятие по заключительному состоянию) - По завершению чтения входной строки магазин пуст (принятие по пустому магазину)

**Связь с КС-грамматиками**: - **Теорема**: Язык принимается автоматом с магазинной памятью тогда и только тогда, когда он является контекстно-свободным. - **Построение АМП по КСГ**: 1. Привести грамматику к нормальной форме Грейбах 2. Создать АМП, моделирующий левосторонний вывод в грамматике 3. Магазин используется для хранения еще не обработанных нетерминалов - **Построение КСГ по АМП**: 1. Для каждой тройки состояний p, q, r создать нетерминал A[p,q,r] 2. A[p,q,r] порождает все строки, которые автомат может прочитать, переходя из состояния p в состояние q при условии, что символ магазина, находившийся под верхним символом в начале работы, оказывается верхним символом в конце

**6. Машина Тьюринга. Формат команд и программа машины Тьюринга. Рекурсивные и рекурсивно перечислимые языки.**

**Машина Тьюринга (МТ)** – абстрактная вычислительная модель, представляющая собой автомат с конечным числом состояний, снабженный бесконечной лентой, разделенной на ячейки.

Формально, МТ M = (Q, Σ, Γ, δ, q₀, B, F), где: - Q – конечное множество состояний - Σ – входной алфавит - Γ – ленточный алфавит (Σ ⊂ Γ) - δ: Q × Γ → Q × Γ × {L, R, S} – функция переходов - q₀ ∈ Q – начальное состояние - B ∈ Γ– пустой символ - F ⊆ Q – множество заключительных состояний

**Формат команд (инструкций) МТ**: - δ(q, a) = (p, b, D) означает: “В состоянии q, читая символ a, заменить a на b, перейти в состояние p и сдвинуть головку в направлении D” - D = L (влево), R (вправо) или S (остаться на месте)

**Программа МТ** – набор команд, охватывающих все возможные пары (состояние, символ).

**Конфигурация МТ** – тройка (q, w₁, w₂), где q – текущее состояние, w₁ – часть ленты слева от головки, w₂ – символ под головкой и часть ленты справа.

**Рекурсивные языки**: - Язык L называется рекурсивным, если существует МТ, которая всегда останавливается и принимает все строки из L и только их. - Соответствуют разрешимым проблемам – для них существует алгоритм, который за конечное время дает ответ “да” или “нет”.

**Рекурсивно перечислимые языки**: - Язык L называется рекурсивно перечислимым, если существует МТ, которая принимает все строки из L и только их (но может не останавливаться на строках не из L). - Соответствуют перечислимым проблемам – для них существует алгоритм, который за конечное время дает ответ “да” для положительных случаев, но может не останавливаться для отрицательных.

**Соотношение классов языков**: - Все рекурсивные языки являются рекурсивно перечислимыми - Существуют рекурсивно перечислимые языки, не являющиеся рекурсивными - Дополнение рекурсивного языка является рекурсивным - Дополнение рекурсивно перечислимого языка не обязательно рекурсивно перечислимо

**7. Недетерминированные машины Тьюринга. Тезис Чёрча-Тьюринга.**

**Недетерминированная машина Тьюринга (НМТ)** – расширение концепции МТ, где на каждом шаге возможно несколько вариантов перехода.

Формально, НМТ M = (Q, Σ, Γ, δ, q₀, B, F), где: - δ: Q × Γ → 2^(Q × Γ × {L, R, S}) – функция переходов, возвращающая множество возможных действий

**Принятие входной строки**: - НМТ принимает строку, если существует хотя бы одна последовательность выборов, приводящая к принятию - НМТ отвергает строку, если все возможные последовательности выборов приводят к отвержению

**Теорема**: Для любой НМТ существует эквивалентная ей детерминированная МТ. - Детерминированная МТ может моделировать работу НМТ, используя поиск в ширину или глубину по всем возможным путям вычисления

**Тезис Чёрча-Тьюринга** (не доказуемый математически): - Любой алгоритм может быть реализован на машине Тьюринга - Любой “естественный” способ формализации понятия алгоритма приводит к тому же классу вычислимых функций, что и машины Тьюринга

**Эквивалентные модели вычислений**: - Машины Тьюринга (различные варианты) - Рекурсивные функции (по Чёрчу) - Лямбда-исчисление - Нормальные алгоритмы Маркова - Регистровые машины - Современные языки программирования (с неограниченной памятью)

**8. Проблема остановки и другие алгоритмически неразрешимые проблемы.**

**Проблема остановки**: Дана программа (машина Тьюринга) P и входные данные x. Остановится ли P на входе x?

**Теорема**: Проблема остановки алгоритмически неразрешима. - Доказательство: Если бы существовал алгоритм H, решающий проблему остановки, можно было бы построить программу D, которая при получении своего описания в качестве входа зацикливается, если H предсказывает остановку, и останавливается, если H предсказывает зацикливание. Применение D к собственному описанию приводит к противоречию.

**Другие неразрешимые проблемы**:

**Проблема пустоты для машин Тьюринга**: Дана МТ M. Является ли язык L(M) пустым?

**Проблема эквивалентности для машин Тьюринга**: Даны две МТ M₁ и M₂. Верно ли, что L(M₁) = L(M₂)?

**Проблема принадлежности для КС-грамматик**: Дана КС-грамматика G и строка w. Порождает ли G строку w?

**Проблема пост-соответствия**: Даны две последовательности строк u₁, u₂, …, uₙ и v₁, v₂, …, vₙ. Существует ли последовательность индексов i₁, i₂, …, iₖ такая, что u\_{i₁}u\_{i₂}…u\_{i₃} = v\_{i₁}v\_{i₂}…v\_{i₃}?

**Проблема самоприменимости**: Дана МТ M. Принимает ли M свое собственное описание?

**Десятая проблема Гильберта**: Существует ли алгоритм, определяющий, имеет ли произвольное диофантово уравнение целочисленное решение?

**Сводимость по Тьюрингу**: Проблема A сводится к проблеме B, если существует алгоритм решения A с использованием алгоритма решения B как подпрограммы.

**Теорема Райса**: Любое нетривиальное свойство языков, распознаваемых МТ (рекурсивно перечислимых языков), алгоритмически неразрешимо.

**9. Основы теории сложности. Классы P и NP.**

**Теория сложности** изучает ресурсы (время, память), необходимые для решения задач алгоритмическими методами.

**Временная сложность алгоритма** – функция T(n), указывающая максимальное количество элементарных операций, необходимых для обработки входа размера n.

**Пространственная сложность алгоритма** – функция S(n), указывающая максимальный объем памяти, необходимой для обработки входа размера n.

**Асимптотическая нотация**: - O(f(n)): верхняя граница роста функции - Ω(f(n)): нижняя граница роста функции - Θ(f(n)): точная граница роста функции

**Класс P** (Polynomial) – множество задач, для которых существует алгоритм решения с полиномиальной временной сложностью O(n^k).

**Класс NP** (Nondeterministic Polynomial) – множество задач, для которых существует алгоритм недетерминированной машины Тьюринга с полиномиальной временной сложностью. - Эквивалентно: множество задач, для которых существует алгоритм проверки предполагаемого решения с полиномиальной временной сложностью.

**Соотношение классов P и NP**: - P ⊆ NP - Открытая проблема: P = NP?

**NP-полные задачи** – подмножество задач из NP, к которым сводится любая задача из NP за полиномиальное время. - Если хотя бы одна NP-полная задача имеет полиномиальное решение, то P = NP.

**Примеры NP-полных задач**: 1. Задача выполнимости булевых формул (SAT) 2. Задача о клике в графе 3. Задача о вершинном покрытии 4. Задача о рюкзаке 5. Задача о коммивояжере (TSP) 6. Задача о раскраске графа

**Сводимость по Карпу**: Задача A сводится к задаче B, если существует полиномиальный алгоритм, преобразующий вход задачи A во вход задачи B так, что ответы на обе задачи совпадают.

**Теорема Кука-Левина**: Задача выполнимости булевых формул (SAT) является NP-полной.