Математические модели в морфологии

Введение. Теория формальных языков.

Алексей Андреевич Сорокин

спецкурс, ОТИПЛ МГУ,
осенний семестр 2017–2018 учебного года
19 сентября 2017 г.

Детерминированные конечные автоматы

Детерминированные конечные автоматы

Определение

Автомат с однобуквенными переходами — детерминированный, если ни из какого состояния не выходит двух рёбер, помеченных одинаковыми буквами.

Теорема

Каждый автоматный язык распознаётся детерминированным автоматом.

Детерминированные конечные автоматы

Определение

Автомат с однобуквенными переходами — детерминированный, если ни из какого состояния не выходит двух рёбер, помеченных одинаковыми буквами.

Теорема

Каждый автоматный язык распознаётся детерминированным автоматом.

Схема доказательства

- Новые состояния — множества старых состояний.

Детерминированные конечные автоматы

Определение

Автомат с однобуквенными переходами — детерминированный, если ни из какого состояния не выходит двух рёбер, помеченных одинаковыми буквами.

Теорема

Каждый автоматный язык распознаётся детерминированным автоматом.

Схема доказательства

- Новые состояния — множества старых состояний.
- Ребро, помеченное a, ведёт из Q_1 в Q_2, если Q_2 содержит в точности состояния,достижимые из Q_1 по a.

Детерминированные конечные автоматы

Определение

Автомат с однобуквенными переходами — детерминированный, если ни из какого состояния не выходит двух рёбер, помеченных одинаковыми буквами.

Теорема

Каждый автоматный язык распознаётся детерминированным автоматом.

Схема доказательства

- Новые состояния — множества старых состояний.
- Ребро, помеченное a, ведёт из Q_1 в Q_2, если Q_2 содержит в точности состояния,достижимые из Q_1 по a.
- Стартовое множество состояний $Q_0 = \{q_0\}$.

Детерминированные конечные автоматы

Определение

Автомат с однобуквенными переходами — детерминированный, если ни из какого состояния не выходит двух рёбер, помеченных одинаковыми буквами.

Теорема

Каждый автоматный язык распознаётся детерминированным автоматом.

Схема доказательства

- Новые состояния — множества старых состояний.
- Ребро, помеченное a, ведёт из Q_1 в Q_2, если Q_2 содержит в точности состояния,достижимые из Q_1 по a.
- Стартовое множество состояний $Q_0 = \{q_0\}$.
- Завершающие состояния: множества, содержащие хотя бы одно завершающее.

Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Схема доказательства

- Нужно строить по любому конечному автомату эквивалентное регулярное выражение и наоборот: по выражению — автомат.

Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Схема доказательства

- Нужно строить по любому конечному автомату эквивалентное регулярное выражение и наоборот: по выражению — автомат.
- Автомат $\rightarrow$ выражение: сложно.
- Выражение $\rightarrow$ автомат: индукция по построению:

Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Схема доказательства

- Нужно строить по любому конечному автомату эквивалентное регулярное выражение и наоборот: по выражению — автомат.
- Автомат $\rightarrow$ выражение: сложно.
- Выражение $\rightarrow$ автомат: индукция по построению:
- Регулярные языки строятся из базовых с помощью операций.

Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Схема доказательства

- Нужно строить по любому конечному автомату эквивалентное регулярное выражение и наоборот: по выражению — автомат.
- Автомат $\to$ выражение: сложно.
- Выражение $\to$ автомат: индукция по построению:
- Регулярные языки строятся из базовых с помощью операций.
- Базовые регулярные языки (буквы и пустое слово) — автоматные.

Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Схема доказательства

- Нужно строить по любому конечному автомату эквивалентное регулярное выражение и наоборот: по выражению — автомат.
- Автомат $\to$ выражение: сложно.
- Выражение $\to$ автомат: индукция по построению:
- Регулярные языки строятся из базовых с помощью операций.
- Базовые регулярные языки (буквы и пустое слово) — автоматные.
- Надо доказать, что операции сохраняют автоматность.

Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Схема доказательства

Конкатенация: $L_1 = L(M_1), L_2 = L(M_2) \to L_1 \cdot L_2 = L(M)$

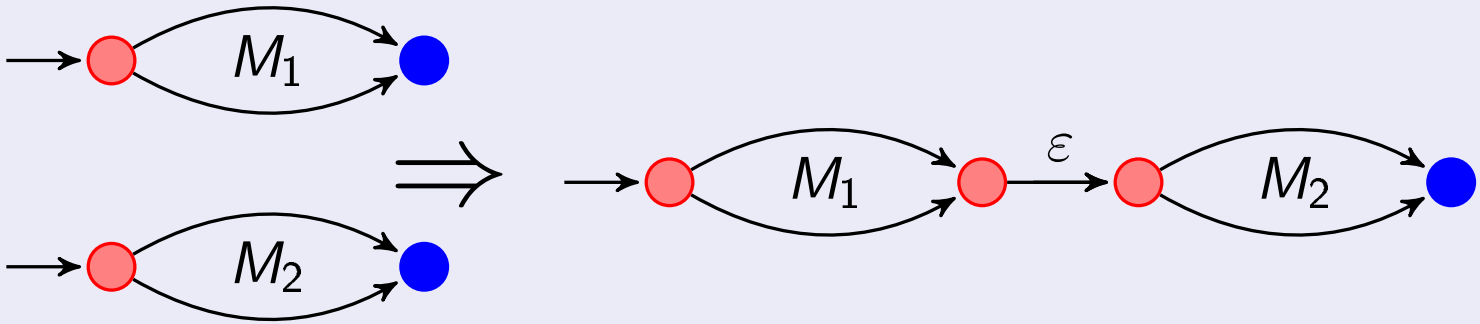
Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Схема доказательства

Конкатенация: $L_1 = L(M_1), L_2 = L(M_2) \to L_1 \cdot L_2 = L(M)$



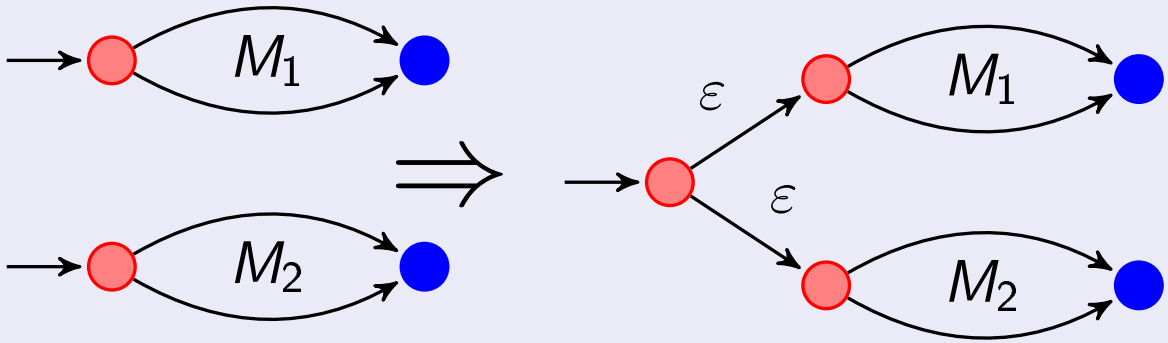
Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Схема доказательства

Объединение: $L_1 = L(M_1), L_2 = L(M_2) \to L_1 \cup L_2 = L(M)$



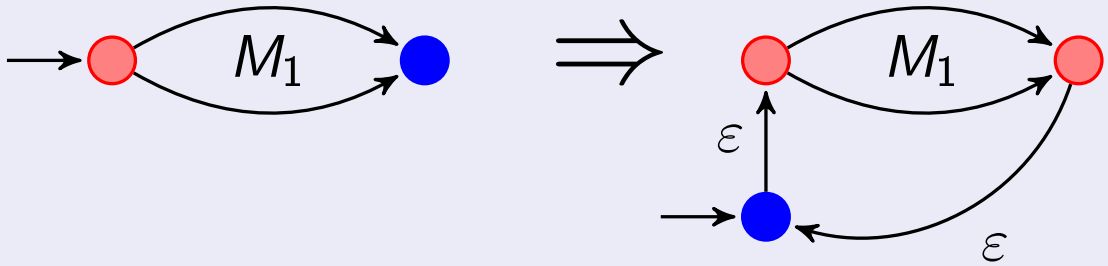
Теорема Клини

Теорема

Классы автоматных и регулярных языков совпадают.

Схема доказательства

Итерация: $L_1 = L(M_1), L_1^* = L(M)$



Дополнение автоматного языка

Теорема

Класс автоматных языков замкнут относительно дополнения.

Идея доказательства

Дополнение автоматного языка

Теорема

Класс автоматных языков замкнут относительно дополнения.

Идея доказательства

- Рассмотрим детерминированный автомат для языка L.

Дополнение автоматного языка

Теорема

Класс автоматных языков замкнут относительно дополнения.

Идея доказательства

- Рассмотрим детерминированный автомат для языка L.
- Пополним его новым “стоковым” состоянием q'.
- Если из состояния q_1 нет ребра по букве a, добавим ребро $\langle q_1, a \rangle \to q'$.

Дополнение автоматного языка

Теорема

Класс автоматных языков замкнут относительно дополнения.

Идея доказательства

- Рассмотрим детерминированный автомат для языка L.
- Пополним его новым “стоковым” состоянием q'.
- Если из состояния q_1 нет ребра по букве a, добавим ребро $\langle q_1, a\rangle \to q'$.
- Добавим рёбра $\langle q', a\rangle \to a$ для всех a.

Дополнение автоматного языка

Теорема

Класс автоматных языков замкнут относительно дополнения.

Идея доказательства

- Рассмотрим детерминированный автомат для языка L.
- Пополним его новым “стоковым” состоянием q'.
- Если из состояния q_1 нет ребра по букве a, добавим ребро $\langle q_1, a\rangle \to q'$.
- Добавим рёбра $\langle q', a\rangle \to a$ для всех a.
- Теперь для всех $q_1 \in Q, a \in \Sigma$ есть ребро вида $\langle q_1, a\rangle \to q_2$.

Дополнение автоматного языка

Теорема

Класс автоматных языков замкнут относительно дополнения.

Идея доказательства

- Рассмотрим детерминированный автомат для языка L.
- Пополним его новым “стоковым” состоянием q'.
- Если из состояния q_1 нет ребра по букве a, добавим ребро $\langle q_1, a\rangle \to q'$.
- Добавим рёбра $\langle q', a\rangle \to a$ для всех a.
- Теперь для всех $q_1 \in Q, a \in \Sigma$ есть ребро вида $\langle q_1, a\rangle \to q_2$.
- То есть каждое слово w приводит из q_0 ровно в 1 состояние.

Дополнение автоматного языка

Теорема

Класс автоматных языков замкнут относительно дополнения.

Идея доказательства

- Рассмотрим детерминированный автомат для языка L.
- Пополним его новым “стоковым” состоянием q'.
- Если из состояния q_1 нет ребра по букве a, добавим ребро $\langle q_1, a\rangle \to q'$.
- Добавим рёбра $\langle q', a\rangle \to a$ для всех a.
- Теперь для всех $q_1 \in Q, a \in \Sigma$ есть ребро вида $\langle q_1, a\rangle \to q_2$.
- То есть каждое слово w приводит из q_0 ровно в 1 состояние.
- Инвертировав завершающие и незавершающие состояния, получим автомат для дополнения.

Пересечение автоматных языков

Теорема

Класс автоматных языков замкнут относительно пересечения.

Идея доказательства

Пересечение автоматных языков

Теорема

Класс автоматных языков замкнут относительно пересечения.

Идея доказательства

- Короткий вариант: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Пересечение автоматных языков

Теорема

Класс автоматных языков замкнут относительно пересечения.

Идея доказательства

- Короткий вариант: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.
- Длинный (но эффективный): рассмотрим полные детерминированные автоматы M_1, M_2 для языков L_1, L_2.
- Пусть Q_1, Q_2 их множества состояний, q_{01}, q_{02} — стартовые, а F_1, F_2 — множества завершающих.

Пересечение автоматных языков

Теорема

Класс автоматных языков замкнут относительно пересечения.

Идея доказательства

- Короткий вариант: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.
- Длинный (но эффективный): рассмотрим полные детерминированные автоматы M_1, M_2 для языков L_1, L_2.
- Пусть Q_1, Q_2 их множества состояний, q_{01}, q_{02} — стартовые, а F_1, F_2 — множества завершающих.
- Состояния нового автомата — пары старых состояний $\langle q_1, q_2 \rangle$, $q_1 \in Q_1, q_2 \in Q_2$.

Пересечение автоматных языков

Теорема

Класс автоматных языков замкнут относительно пересечения.

Идея доказательства

- Короткий вариант: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.
- Длинный (но эффективный): рассмотрим полные детерминированные автоматы M_1, M_2 для языков L_1, L_2.
- Пусть Q_1, Q_2 их множества состояний, q_{01}, q_{02} — стартовые, а F_1, F_2 — множества завершающих.
- Состояния нового автомата — пары старых состояний $\langle q_1, q_2 \rangle$, $q_1 \in Q_1, q_2 \in Q_2$.
- Новое стартовое — пара старых стартовых $\langle q_{01}, q_{02} \rangle$.
- Автомат симулирует M_1 по первой координате и M_2 — по второй.

Пересечение автоматных языков

Теорема

Класс автоматных языков замкнут относительно пересечения.

Идея доказательства

- Короткий вариант: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.
- Длинный (но эффективный): рассмотрим полные детерминированные автоматы M_1, M_2 для языков L_1, L_2.
- Пусть Q_1, Q_2 их множества состояний, q_{01}, q_{02} — стартовые, а F_1, F_2 — множества завершающих.
- Состояния нового автомата — пары старых состояний $\langle q_1, q_2 \rangle$, $q_1 \in Q_1, q_2 \in Q_2$.
- Новое стартовое — пара старых стартовых $\langle q_{01}, q_{02} \rangle$.
- Автомат симулирует M_1 по первой координате и M_2 — по второй.
- Завершающие состояния — пары завершающих (автомат обязан принимать слово по обеим координатам).

Рекурсивное построение автоматов

- Автоматные языки замкнуты относительно множества операций.

Рекурсивное построение автоматов

- Автоматные языки замкнуты относительно множества операций.
- Это замыкание эффективно: соответствующие автоматы строятся алгоритмически.
- Вывод: можно строить регулярные выражения из автоматов (но с большим числом операций).

Рекурсивное построение автоматов

- Автоматные языки замкнуты относительно множества операций.
- Это замыкание эффективно: соответствующие автоматы строятся алгоритмически.
- Вывод: можно строить регулярные выражения из автоматов (но с большим числом операций).
- Множественное число в английском:

$$(L_{sib} \cdot es) \cup (((\overline{L_{sib}} \cap L_C) \cup L_{Vy} \cup L_V) \cdot s),$$

где

- L_{sib} — слова, кончающиеся на шипящий.
- L_C — слова, кончающиеся на согласный.
- L_{Vy} — слова кончающиеся на гласный+y.
- L_V — слова, кончающиеся на гласный (не y).

Рекурсивное построение автоматов

- Автоматные языки замкнуты относительно множества операций.
- Это замыкание эффективно: соответствующие автоматы строятся алгоритмически.
- Вывод: можно строить регулярные выражения из автоматов (но с большим числом операций).
- Множественное число в английском:

$$(L_{sib} \cdot es) \cup (((\overline{L_{sib}} \cap L_C) \cup L_{Vy} \cup L_V) \cdot s),$$

где

  - L_{sib} — слова, кончающиеся на шипящий.
  - L_C — слова, кончающиеся на согласный.
  - L_{Vy} — слова кончающиеся на гласный+y.
  - L_V — слова, кончающиеся на гласный (не y).
- $L_{sib}, L_C, L_{Vy}, L_V$ — автоматные, финальный автомат строится рекурсивно.

Рекурсивное построение автомата

Инфинитив глагола в турецком

Построить конечный автомат для инфинитива в турецком

- Инфинитив имеет форму основа+mEk.
- E равно e, если последний гласный основы — e, i, $ö$, $ü$, и a — если a, $ı$, o, u.

Рекурсивное построение автомата

Инфинитив глагола в турецком

Построить конечный автомат для инфинитива в турецком

- Инфинитив имеет форму основа+mEk.
- E равно e, если последний гласный основы — e, i, $ö$, $ü$, и a — если a, $ı$, o, u.

- M_1 — автомат для выражения C*V(C|V)*m(a|e)k.

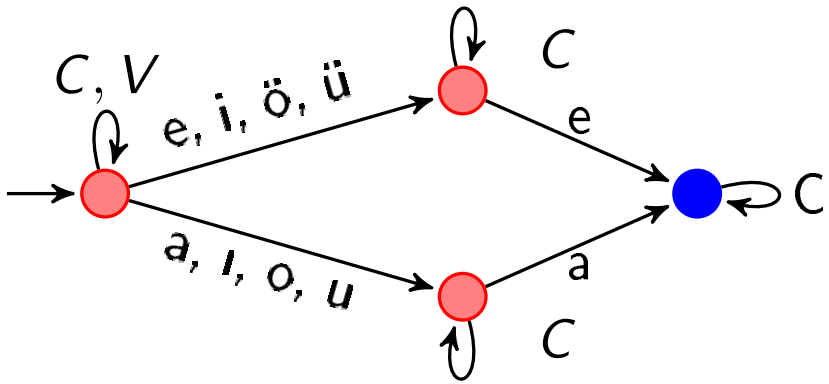
Рекурсивное построение автомата

Инфинитив глагола в турецком

Построить конечный автомат для инфинитива в турецком

- Инфинитив имеет форму основа+mEk.
- E равно e, если последний гласный основы — e, i, $ö$, $ü$, и a — если a, $ı$, o, u.

- M_1 — автомат для выражения C*V(C|V)*m(a|e)k.
- M_2 проверяет гармонию гласных:



- $M_1 \cap M_2$ — требуемый автомат.

Рекурсивное построение автомата

Инфинитив пассива в турецком

Построить конечный автомат для формы инфинитива пассивного залога в турецком языке

- Инфинитив имеет форму основа+X+*mEk*.
- *E* равно *e*, если последний гласный основы — *e*, *i*, *ö*, *ü*, и *a* — если *a*, *ı*, *o*, *u*.
- Суффикс *X* равен *-n*, если основа кончается на гласный, *-In* — если на *l*, и *-Il* — иначе.
- Гласный *I* равен *ı* после *a*, *ı*; *u* после *u*, *o*; *i* после *e*, *i*; *ü* после *ü*, *ö*.

Инфинитив	Инфинитив пассива
varmak “*прибывать*”	*varılmak*
silmek “*удалять*”	*silinmek*
büyümek “*увеличивать*”	*büyünmek*
durmak “*останавливать*”	*durulmak*
bilmek “*знать*”	*bilinmak*

Регулярные преобразования

Пусть зафиксирован конечный алфавит Σ.

Регулярные преобразования

Пусть зафиксирован конечный алфавит Σ.

- Базовые регулярные преобразования: $a : b$ — заменить a на b, где $a, b \in \Sigma \cup 1$.

Регулярные преобразования

Пусть зафиксирован конечный алфавит Σ.

- Базовые регулярные преобразования: $a : b$ — заменить a на b, где $a, b \in \Sigma \cup 1$.
- Бинарные операции: $|$ (объединение) и $\cdot$ (конкатенация или приписывание).
- Унарные операция $*, +$ (итерация, положительная операция), смысл операций тот же, что и в регулярных выражениях.

Регулярные преобразования

Пусть зафиксирован конечный алфавит Σ.

- Базовые регулярные преобразования: $a : b$ — заменить a на b, где $a, b \in \Sigma \cup 1$.
- Бинарные операции: $|$ (объединение) и $\cdot$ (конкатенация или приписывание).
- Унарные операция $*, +$ (итерация, положительная операция), смысл операций тот же, что и в регулярных выражениях.
- **Пример**: преобразование $(a : a \,|\, b : b \,|\, c : c)^+ (1 : a)^*$ задаёт приписывание произвольного количества a в конце любого непустого слова из a, b, c.

Регулярные преобразования

Пусть зафиксирован конечный алфавит Σ.

- Базовые регулярные преобразования: $a : b$ — заменить a на b, где $a, b \in \Sigma \cup 1$.
- Бинарные операции: $|$ (объединение) и $\cdot$ (конкатенация или приписывание).
- Унарные операция $*, +$ (итерация, положительная операция), смысл операций тот же, что и в регулярных выражениях.
- **Пример**: преобразование $(a : a \,|\, b : b \,|\, c : c)^+ (1 : a)^*$ задаёт приписывание произвольного количества a в конце любого непустого слова из a, b, c.
- В качестве регулярных преобразований можно использовать $\alpha : \beta$, где α, β — произвольные регулярные выражения.

Регулярные преобразования

Пусть зафиксирован конечный алфавит Σ.

- Базовые регулярные преобразования: $a : b$ — заменить a на b, где $a, b \in \Sigma \cup 1$.
- Бинарные операции: $|$ (объединение) и $\cdot$ (конкатенация или приписывание).
- Унарные операция $*, +$ (итерация, положительная операция), смысл операций тот же, что и в регулярных выражениях.
- **Пример**: преобразование $(a : a \mid b : b \mid c : c)^+(1 : a)^*$ задаёт приписывание произвольного количества a в конце любого непустого слова из a, b, c.
- В качестве регулярных преобразований можно использовать $\alpha : \beta$, где α, β — произвольные регулярные выражения.
- Регулярные выражения — тоже регулярные преобразования (α означает тождественное преобразование, ограниченное на α).

Примеры регулярных преобразований

- После каждой буквы a вставить букву b: $(a(1:b) \mid b \mid c)^*$.

Примеры регулярных преобразований

- После каждой буквы a вставить букву b: $(a(1:b) \mid b \mid c)^*$.
- Вставить букву b между буквами a и c:

Примеры регулярных преобразований

- После каждой буквы a вставить букву b: $(a(1:b) \mid b \mid c)^*$.
- Вставить букву b между буквами a и c: $(a^+(b|((1:b)c)) \mid b \mid c)^* a^*$.

Примеры регулярных преобразований

- После каждой буквы a вставить букву b: $(a(1:b) \mid b \mid c)^*$.
- Вставить букву b между буквами a и c: $(a^+(b|((1:b)c)) \mid b \mid c)^* a^*$.
- Заменить все последовательности из a на одну букву a, если они идут после c:

Примеры регулярных преобразований

- После каждой буквы a вставить букву b: $(a(1:b) \mid b \mid c)^*$.
- Вставить букву b между буквами a и c: $(a^+(b|((1:b)c)) \mid b \mid c)^* a^*$.
- Заменить все последовательности из a на одну букву a, если они идут после c: $(c^+(b|(a^+:a)) \mid a \mid b)^* c^*$

Конечные преобразователи

Неформально, конечный преобразователь — это автомат с добавленными на рёбра выходными символами.

Конечные преобразователи

Неформально, конечный преобразователь — это автомат с добавленными на рёбра выходными символами.

Пусть Σ, Γ — конечные алфавиты.

Определение конечного преобразователя

Конечный преобразователь: кортеж $M = \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle$, где

- Q — конечное множество состояний

Конечные преобразователи

Неформально, конечный преобразователь — это автомат с добавленными на рёбра выходными символами.

Пусть Σ, Γ — конечные алфавиты.

Определение конечного преобразователя

Конечный преобразователь: кортеж $M = \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle$, где

- Q — конечное множество состояний
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ — конечное множество переходов

Конечные преобразователи

Неформально, конечный преобразователь — это автомат с добавленными на рёбра выходными символами.

Пусть Σ, Γ — конечные алфавиты.

Определение конечного преобразователя

Конечный преобразователь: кортеж $M = \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle$, где

- Q — конечное множество состояний
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ — конечное множество переходов
- $q_0 \in Q$ — стартовое состояние
- $F \subseteq Q$ — завершающие состояния.

Конечные преобразователи

Неформально, конечный преобразователь — это автомат с добавленными на рёбра выходными символами.

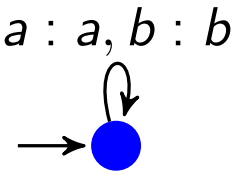
Пусть Σ, Γ — конечные алфавиты.

Определение конечного преобразователя

Конечный преобразователь: кортеж $M = \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle$, где

- Q — конечное множество состояний
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ — конечное множество переходов
- $q_0 \in Q$ — стартовое состояние
- $F \subseteq Q$ — завершающие состояния.

Простейший преобразователь — тождественный (алфавит a, b):



Конечные преобразователи

Неформально, конечный преобразователь — это автомат с добавленными на рёбра выходными символами.

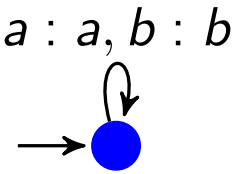
Пусть Σ, Γ — конечные алфавиты.

Определение конечного преобразователя

Конечный преобразователь: кортеж $M = \langle Q, \Sigma, \Gamma, \Delta, q_0, F \rangle$, где

- Q — конечное множество состояний
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ — конечное множество переходов
- $q_0 \in Q$ — стартовое состояние
- $F \subseteq Q$ — завершающие состояния.

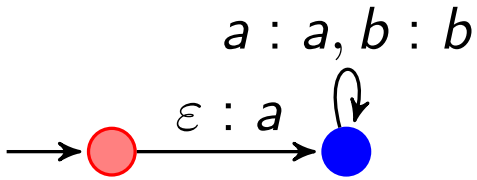
Простейший преобразователь — тождественный (алфавит a, b):



Конечные автоматы можно понимать как преобразователи, возвращающие свой вход (и проверяющие, что вход принадлежит нужному множеству).

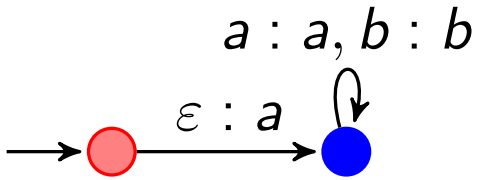
Конечные преобразователи: примеры

- Добавляет a в начале слова:

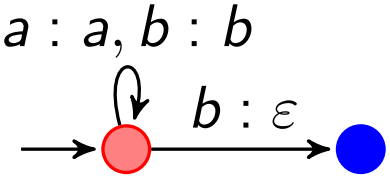


Конечные преобразователи: примеры

- Добавляет a в начале слова:

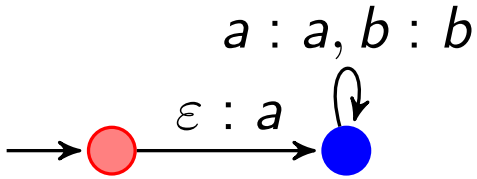


- Удаляет конечную b в тех словах, где она есть, и отвергает остальные:

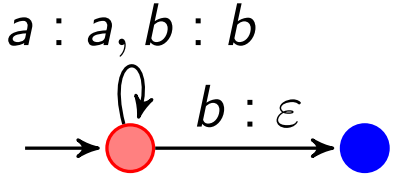


Конечные преобразователи: примеры

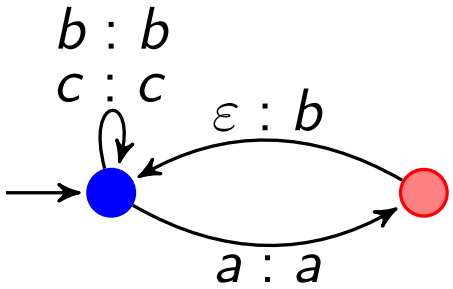
- Добавляет a в начале слова:



- Удаляет конечную b в тех словах, где она есть, и отвергает остальные:

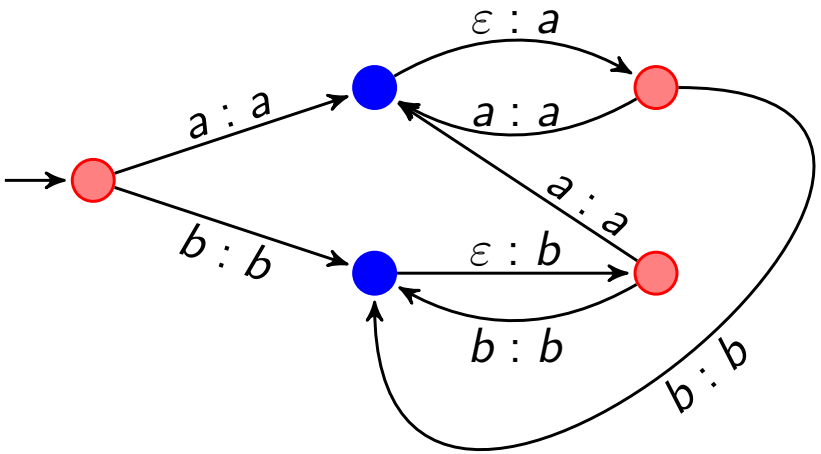


- После каждой a добавляет b:



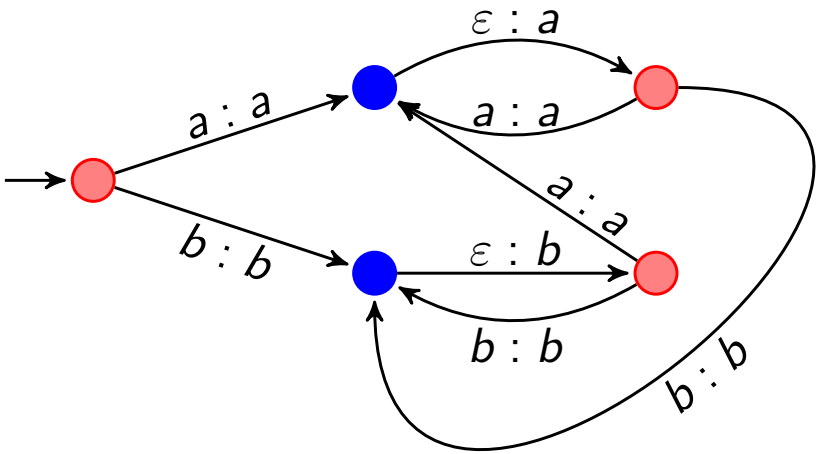
Конечные преобразователи: примеры

- Удваивает все буквы, кроме последней:

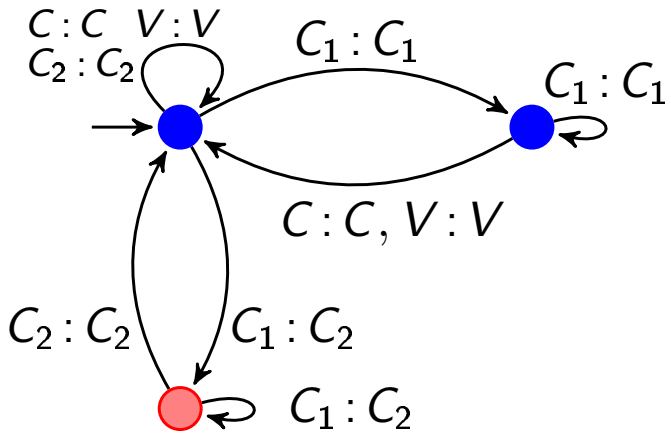


Конечные преобразователи: примеры

- Удваивает все буквы, кроме последней:



- Ретро-ассимилирует C_1 в C_2 (в последовательности C_1, идущей перед C_2, все буквы заменяются на C_2)



Свойства конечных автоматов

- Конечные преобразователи задают в точности регулярные преобразования (аналог теоремы Клини).

Свойства конечных автоматов

- Конечные преобразователи задают в точности регулярные преобразования (аналог теоремы Клини).
- Замкнутость относительно операций:

Операция	Автоматы	Преобразователи
Конкатенация	Да	Да
Итерация	Да	Да
Объединение	Да	Да
Пересечение	Да	Нет
Дополнение	Да	Нет

Свойства конечных преобразователей

- Конечные преобразования также замкнуты относительно
  - Композиции ($\circ$).
  - Приоритетного объединения ($\cup_p$).

Свойства конечных преобразователей

- Конечные преобразования также замкнуты относительно
  - Композиции ($\circ$).
  - Приоритетного объединения ($\cup_p$).

$$(T_1 \cup_p T_2)(u) = \begin{cases} T_1(u), & T_1(u) \neq \varnothing, \\ T_2(u), & T_1(u) = \varnothing. \end{cases}$$

Свойства конечных преобразователей

- Конечные преобразования также замкнуты относительно
  - Композиции ($\circ$).
  - Приоритетного объединения ($\cup_p$).

$$(T_1 \cup_p T_2)(u) = \begin{cases} T_1(u), & T_1(u) \neq \varnothing, \\ T_2(u), & T_1(u) = \varnothing. \end{cases}$$

  - Обращения ($(\cdot)^{-1}$): $\phi^{-1} = \{\langle y, x\rangle \mid \langle x, y\rangle \in \phi\}$.

Свойства конечных преобразователей

- Конечные преобразования также замкнуты относительно
  - Композиции ($\circ$).
  - Приоритетного объединения ($\cup_p$).

$$(T_1 \cup_p T_2)(u) = \begin{cases} T_1(u), & T_1(u) \neq \varnothing, \\ T_2(u), & T_1(u) = \varnothing. \end{cases}$$

  - Обращения ($(\cdot)^{-1}$): $\phi^{-1} = \{\langle y, x\rangle \mid \langle x, y\rangle \in \phi\}$.
- Применения операций:
  - Композиция: последовательное применение преобразований,

Свойства конечных преобразователей

- Конечные преобразования также замкнуты относительно
  - Композиции ($\circ$).
  - Приоритетного объединения ($\cup_p$).

$$(T_1 \cup_p T_2)(u) = \begin{cases} T_1(u), & T_1(u) \neq \varnothing, \\ T_2(u), & T_1(u) = \varnothing. \end{cases}$$

  - Обращения ($(\cdot)^{-1}$): $\phi^{-1} = \{\langle y, x\rangle \mid \langle x, y\rangle \in \phi\}$.
- Применения операций:
  - Композиция: последовательное применение преобразований,
  - Обращение: переход от синтеза к анализу и наоборот,

Свойства конечных преобразователей

- Конечные преобразования также замкнуты относительно
  - Композиции ($\circ$).
  - Приоритетного объединения ($\cup_p$).

$$(T_1 \cup_p T_2)(u) = \begin{cases} T_1(u), & T_1(u) \neq \varnothing, \\ T_2(u), & T_1(u) = \varnothing. \end{cases}$$

  - Обращения ($(\cdot)^{-1}$): $\phi^{-1} = \{\langle y, x\rangle \mid \langle x, y\rangle \in \phi\}$.
- Применения операций:
  - Композиция: последовательное применение преобразований,
  - Обращение: переход от синтеза к анализу и наоборот,
  - Приоритетное объединение: отдельная обработка нерегулярных форм.

Множественное число существительного в английском

Пример.

Опишите преобразователь, преобразующий форму единственного числа существительного в форму множественного для английского языка.

Множественное число существительного в английском

Пример.

Опишите преобразователь, преобразующий форму единственного числа существительного в форму множественного для английского языка.

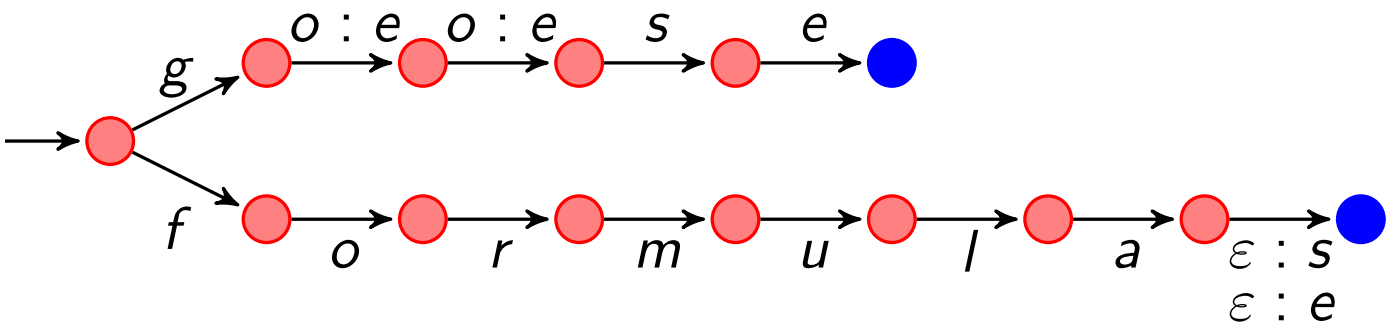
- *torch* $\leftrightarrows$ *torches*
- *monarch* $\leftrightarrows$ *monarchs*
- *ally* $\leftrightarrows$ *allies*
- *play* $\leftrightarrows$ *plays*
- *goose* $\leftrightarrows$ *geese*
- *formula* $\leftrightarrows$ *formulas/formulae*

Пример: множественное число в английском

Пример.

Описать преобразователь, строящий форму множественного числа существительного в английском фзыке.

- Отдельный преобразователь T_{exc} для исключений:

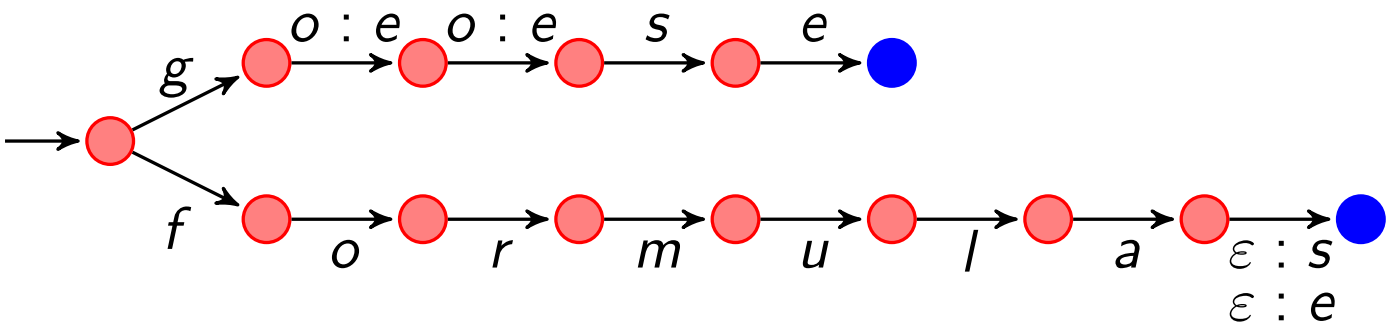


Пример: множественное число в английском

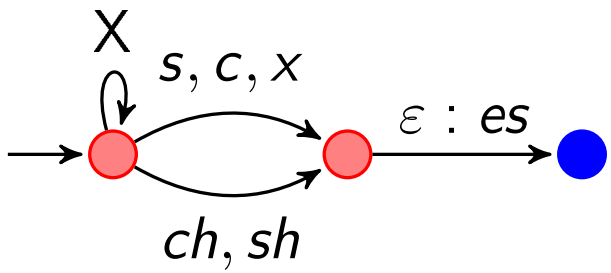
Пример.

Описать преобразователь, строящий форму множественного числа существительного в английском фзыке.

- Отдельный преобразователь T_{exc} для исключений:

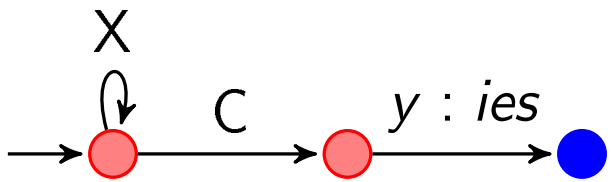


- T_{sib}: добавляет *-es* после финальной шипящей (X — произвольный символ):



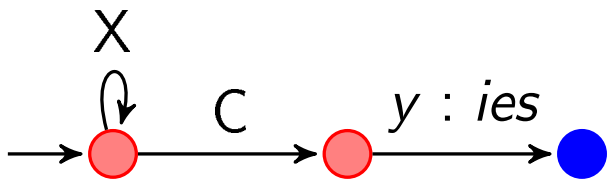
Пример: множественное число в английском

- Преобазователь T_{exc} для исключений.
- Преобразователь T_{sib} для добавления *es*.
- Преобразователь T_{Cy} для замены *y* на *-ies* после согласной.



Пример: множественное число в английском

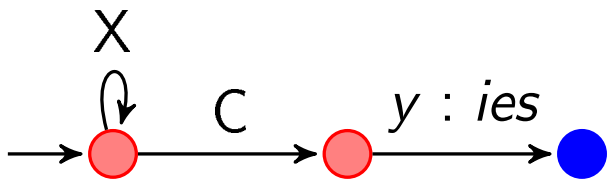
- Преобазователь T_{exc} для исключений.
- Преобразователь T_{sib} для добавления *es*.
- Преобразователь T_{Cy} для замены *y* на *-ies* после согласной.



- T_s — добавляет s в конце.
- $T_{exc,sib}$ — добавляет s к исключениям на *-arch* и отвергает остальные слова (для *monarchs*, *tetrarchs*, . . .).

Пример: множественное число в английском

- Преобазователь T_{exc} для исключений.
- Преобразователь T_{sib} для добавления *es*.
- Преобразователь T_{Cy} для замены *y* на *-ies* после согласной.



- T_s — добавляет s в конце.
- $T_{exc,sib}$ — добавляет s к исключениям на *-arch* и отвергает остальные слова (для *monarchs*, *tetrarchs*, ...).
- Общее решение:

$$T_{exc} \cup_p T_{exc,sib} \cup_p T_{sib} \cup_p T_{Cy} \cup_p T_s$$

Контекстная замена

- Самый частый тип преобразования — контекстная замена:

$$X \rightarrow Y \,||\, U\_V$$

“Заменить X на Y, если слева стоит U, а справа V.”

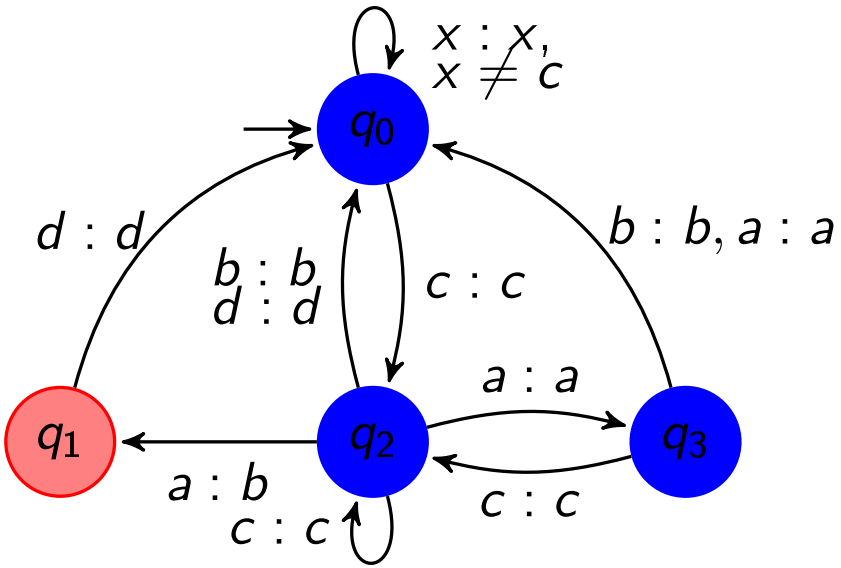
Контекстная замена

- Самый частый тип преобразования — контекстная замена:

$$X \to Y \,||\, U\_V$$

"Заменить X на Y, если слева стоит U, а справа V."

- В простейшем случае X, Y, U, V — буквы.
- Преобразователь для $a \to b||c\_d$:



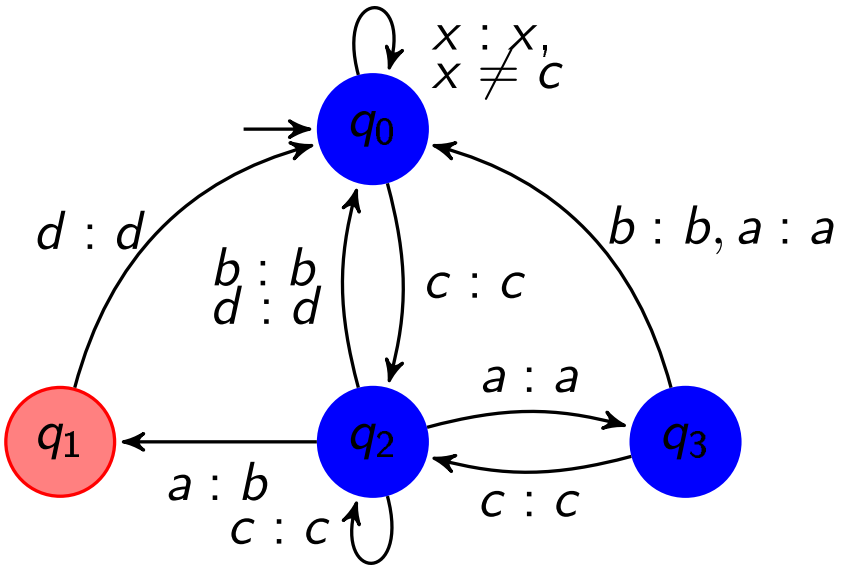
Контекстная замена

- Самый частый тип преобразования — контекстная замена:

$$X \to Y \,||\, U\_V$$

"Заменить X на Y, если слева стоит U, а справа V."

- В простейшем случае X, Y, U, V — буквы.
- Преобразователь для $a \to b||c\_d$:



- В общем случае X, Y, U, V — произвольные регулярные выражения.

Пример: снова множественное число

- Наша модель для английского лингвистически неадекватна.
- Нет отдельных окончаний *-es*, *-ies*, *-s*, они алломорфы окончания *-s*.

Пример: снова множественное число

- Наша модель для английского лингвистически неадекватна.
- Нет отдельных окончаний *-es*, *-ies*, *-s*, они алломорфы окончания *-s*.
- Нужно сначала присоединить прототипическое окончание, потом моделировать контекстные явления.
- Преобразователи для правил:

Пример: снова множественное число

- Наша модель для английского лингвистически неадекватна.
- Нет отдельных окончаний *-es*, *-ies*, *-s*, они алломорфы окончания *-s*.
- Нужно сначала присоединить прототипическое окончание, потом моделировать контекстные явления.
- Преобразователи для правил:
  - T_s: добавить $!s$ в конце слова (! — маркер границы) $\varepsilon \to !s \,||\, \_\$$ ($ — маркер конца слова).

Пример: снова множественное число

- Наша модель для английского лингвистически неадекватна.
- Нет отдельных окончаний *-es*, *-ies*, *-s*, они алломорфы окончания *-s*.
- Нужно сначала присоединить прототипическое окончание, потом моделировать контекстные явления.
- Преобразователи для правил:
  - T_s: добавить $!s$ в конце слова (! — маркер границы) $\varepsilon \to !s \,||\, \_\$$ ($ — маркер конца слова).
  - T_{sib}: добавить e после ! и после шипящей $\varepsilon \to e \,||\, (s|z|x|sh|ch)\_!$.

Пример: снова множественное число

- Наша модель для английского лингвистически неадекватна.
- Нет отдельных окончаний *-es*, *-ies*, *-s*, они алломорфы окончания *-s*.
- Нужно сначала присоединить прототипическое окончание, потом моделировать контекстные явления.
- Преобразователи для правил:
  - T_s: добавить $!s$ в конце слова ($!$ — маркер границы) $\varepsilon \to !s \,||\, \_\$$ ($\$$ — маркер конца слова).
  - T_{sib}: добавить e после $!$ и после шипящей $\varepsilon \to e \,||\, (s|z|x|sh|ch)\_!$.
  - T_y: заменить y на ie перед маркером границы и после согласной $y \to ie \,||\, C\_!$.

Пример: снова множественное число

- Наша модель для английского лингвистически неадекватна.
- Нет отдельных окончаний *-es*, *-ies*, *-s*, они алломорфы окончания *-s*.
- Нужно сначала присоединить прототипическое окончание, потом моделировать контекстные явления.
- Преобразователи для правил:
  - T_s: добавить $!s$ в конце слова ($!$ — маркер границы) $\varepsilon \to !s \,||\, \_\$$ ($\$$ — маркер конца слова).
  - T_{sib}: добавить e после $!$ и после шипящей $\varepsilon \to e \,||\, (s|z|x|sh|ch)\_!$.
  - T_y: заменить y на ie перед маркером границы и после согласной $y \to ie \,||\, C\_!$.
  - $T_{exc,sib}$: ничего не делать со словами с *arch* на конце (возможно, не все такие слова подойдут). $?^* V ?^* arch!s$
  - T_c: удалить маркер морфемной границы $! \to \varepsilon$.

Пример: снова множественное число

- Наша модель для английского лингвистически неадекватна.
- Нет отдельных окончаний *-es*, *-ies*, *-s*, они алломорфы окончания *-s*.
- Нужно сначала присоединить прототипическое окончание, потом моделировать контекстные явления.
- Преобразователи для правил:
  - T_s: добавить $!s$ в конце слова ($!$ — маркер границы) $\varepsilon \rightarrow !s \,||\, \_\$$ ($\$$ — маркер конца слова).
  - T_{sib}: добавить e после $!$ и после шипящей $\varepsilon \rightarrow e \,||\, (s|z|x|sh|ch)\_!$.
  - T_y: заменить y на ie перед маркером границы и после согласной $y \rightarrow ie \,||\, C\_!$.
  - $T_{exc,sib}$: ничего не делать со словами с *arch* на конце (возможно, не все такие слова подойдут). $?^* V ?^* arch!s$
  - T_c: удалить маркер морфемной границы $! \rightarrow \varepsilon$.
- Финальный преобразователь строится с помощью композиции:

$$T_{exc} \cup_p (T_s \circ (T_{exc,sib} \cup_p T_{sib}) \circ T_y \circ T_c)$$

Примеры вычисления

- Исключения: *mouse*$\mapsto$*mice*, *cactus*$\mapsto$*cactuses*, *cacti*
  — обрабатывается в T_{exc} и не идёт в основную ветку.

Примеры вычисления

- Исключения: *mouse*$\mapsto$*mice*, *cactus*$\mapsto$*cactuses*, *cacti*
  — обрабатывается в T_{exc} и не идёт в основную ветку.

Примеры вычисления

- Исключения: *mouse*$\mapsto$*mice*, *cactus*$\mapsto$*cactuses*, *cacti* — обрабатывается в T_{exc} и не идёт в основную ветку.
- Регулярные формы обрабатываются в ветке $T_s \circ (T_{exc,sib} \cup_p T_{sib}) \circ T_y \circ T_c$:

Примеры вычисления

- Исключения: *mouse*$\mapsto$*mice*, *cactus*$\mapsto$*cactuses*, *cacti* — обрабатывается в T_{exc} и не идёт в основную ветку.
- Регулярные формы обрабатываются в ветке $T_s \circ (T_{exc,sib} \cup_p T_{sib}) \circ T_y \circ T_c$:
  - *day*
  - *rally*
  - *witch*

Примеры вычисления

- Исключения: *mouse*$\mapsto$*mice*, *cactus*$\mapsto$*cactuses*, *cacti* — обрабатывается в T_{exc} и не идёт в основную ветку.
- Регулярные формы обрабатываются в ветке $T_s \circ (T_{exc,sib} \cup_p T_{sib}) \circ T_y \circ T_c$:
  - *day* $\rightarrow$ *day*!*s*
  - *rally* $\rightarrow$ *rally*!*s*
  - *witch* $\rightarrow$ *witch*!*s*

Примеры вычисления

- Исключения: *mouse*$\mapsto$*mice*, *cactus*$\mapsto$*cactuses*, *cacti* — обрабатывается в T_{exc} и не идёт в основную ветку.
- Регулярные формы обрабатываются в ветке $T_s \circ (T_{exc,sib} \cup_p T_{sib}) \circ T_y \circ T_c$:
  - *day* $\rightarrow$ *day*!*s* $\rightarrow$ *day*!*s*
  - *rally* $\rightarrow$ *rally*!*s* $\rightarrow$ *rally*!*s*
  - *witch* $\rightarrow$ *witch*!*s* $\rightarrow$ *witche*!*s*

Примеры вычисления

- Исключения: *mouse*$\mapsto$*mice*, *cactus*$\mapsto$*cactuses*, *cacti* — обрабатывается в T_{exc} и не идёт в основную ветку.
- Регулярные формы обрабатываются в ветке $T_s \circ (T_{exc,sib} \cup_p T_{sib}) \circ T_y \circ T_c$:
  - *day* $\rightarrow$ *day*!*s* $\rightarrow$ *day*!*s* $\rightarrow$ *day*!*s*
  - *rally* $\rightarrow$ *rally*!*s* $\rightarrow$ *rally*!*s* $\rightarrow$ *rallie*!*s*
  - *witch* $\rightarrow$ *witch*!*s* $\rightarrow$ *witche*!*s* $\rightarrow$ *witche*!*s*

Примеры вычисления

- Исключения: *mouse*$\mapsto$*mice*, *cactus*$\mapsto$*cactuses*, *cacti* — обрабатывается в T_{exc} и не идёт в основную ветку.
- Регулярные формы обрабатываются в ветке $T_s \circ (T_{exc,sib} \cup_p T_{sib}) \circ T_y \circ T_c$:
  - *day* $\rightarrow$ *day*!*s* $\rightarrow$ *day*!*s* $\rightarrow$ *day*!*s* $\rightarrow$ *days*
  - *rally* $\rightarrow$ *rally*!*s* $\rightarrow$ *rally*!*s* $\rightarrow$ *rallie*!*s* $\rightarrow$ *rallies*
  - *witch* $\rightarrow$ *witch*!*s* $\rightarrow$ *witche*!*s* $\rightarrow$ *witche*!*s* $\rightarrow$ *witches*

Примеры вычисления

- Исключения: *mouse*$\mapsto$*mice*, *cactus*$\mapsto$*cactuses*, *cacti*
  — обрабатывается в T_{exc} и не идёт в основную ветку.
- Регулярные формы обрабатываются в ветке
  $T_s \circ (T_{exc,sib} \cup_p T_{sib}) \circ T_y \circ T_c$:
  - *day* $\to$ *day*!*s* $\to$ *day*!*s* $\to$ *day*!*s* $\to$ *days*
  - *rally* $\to$ *rally*!*s* $\to$ *rally*!*s* $\to$ *rallie*!*s* $\to$ *rallies*
  - *witch* $\to$ *witch*!*s* $\to$ *witche*!*s* $\to$ *witche*!*s* $\to$ *witches*
- Частичное исключение *monarch* обрабатывается веткой
  $T_s \circ T_{exc,sib} \circ T_y \circ T_c$:
  monarch $\to$ *monarch*!*s* $\to$ *monarch*!*s* $\to$ *monarch*!*s* $\to$ *monarchs*.

Инфинитив пассивного залога в турецком

Инфинитив пассивного залога

Построить конечный преобразователь, преобразующий турецкий глагольный инфинитив в инфинитив пассивного залога.

- Пассив образуется вставкой суффикса перед *-mek/-mak*.
- Суффикс пассива: *-n* после гласной, *-In* после *l* и *-Il* иначе.
- I: *ı* после *a*, *ı*; *u* после *u*, *o*; *i* после *e*, *i*; *ü* после *ü*, *ö*.

Инфинитив	Инфинитив пассива
varmak “прибывать”	*varılmak*
silmek “удалять”	*silinmek*
büyümek “увеличивать”	*büyünmek*
durmak “останавливать”	*durulmak*
bilmek “знать”	*bilinmak*

Инфинитив пассивного залога в турецком

Инфинитив пассивного залога

Построить конечный преобразователь, преобразующий турецкий глагольный инфинитив в инфинитив пассивного залога.

- Пассив образуется вставкой суффикса перед *-mek/-mak*.
- Суффикс пассива: *-n* после гласной, *-In* после *l* и *-Il* иначе.
- I: *ı* после *a*, *ı*; *u* после *u*, *o*; *i* после *e*, *i*; *ü* после *ü*, *ö*.

- T_{mark}: вставить ! перед *-mak/-mek*: $\varepsilon \rightarrow ! \,||\, \_m(a|e)k\$$.

Инфинитив пассивного залога в турецком

Инфинитив пассивного залога

Построить конечный преобразователь, преобразующий турецкий глагольный инфинитив в инфинитив пассивного залога.

- Пассив образуется вставкой суффикса перед *-mek/-mak*.
- Суффикс пассива: *-n* после гласной, *-In* после *l* и *-Il* иначе.
- I: *ı* после *a*, *ı*; *u* после *u*, *o*; *i* после *e*, *i*; *ü* после *ü*, *ö*.

- T_{mark}: вставить ! перед *-mak/-mek*: $\varepsilon \rightarrow ! \,||\, \_m(a|e)k\$$.
- Заменить маркер подходящим суффиксом:

Инфинитив пассивного залога в турецком

Инфинитив пассивного залога

Построить конечный преобразователь, преобразующий турецкий глагольный инфинитив в инфинитив пассивного залога.

- Пассив образуется вставкой суффикса перед *-mek/-mak*.
- Суффикс пассива: *-n* после гласной, *-In* после *l* и *-Il* иначе.
- I: *ı* после *a*, *ı*; *u* после *u*, *o*; *i* после *e*, *i*; *ü* после *ü*, *ö*.

- T_{mark}: вставить ! перед *-mak/-mek*: $\varepsilon \rightarrow ! \,||\, \_ m(a|e)k\$$.
- Заменить маркер подходящим суффиксом:
  - *-n* после гласной (T_V): $! \rightarrow n \,||\, V \_ \$$,
  - *-In* после *l* (T_l): $! \rightarrow \mathrm{I}n \,||\, l \_ \$$,
  - *-Il* по умолчанию (T_{def}): $! \rightarrow \mathrm{I}l \,||\, \_$,

Инфинитив пассивного залога в турецком

Инфинитив пассивного залога

Построить конечный преобразователь, преобразующий турецкий глагольный инфинитив в инфинитив пассивного залога.

- Пассив образуется вставкой суффикса перед *-mek/-mak*.
- Суффикс пассива: *-n* после гласной, *-In* после *l* и *-Il* иначе.
- I: *ı* после *a*, *ı*; *u* после *u*, *o*; *i* после *e*, *i*; *ü* после *ü*, *ö*.

- T_{mark}: вставить ! перед *-mak/-mek*: $\varepsilon \rightarrow ! \,||\, \_m(a|e)k\$$.
- Заменить маркер подходящим суффиксом:
  - *-n* после гласной (T_V): $! \rightarrow n \,||\, V \_ \$$,
  - *-In* после *l* (T_l): $! \rightarrow \mathrm{I}n \,||\, l \_ \$$,
  - *-Il* по умолчанию (T_{def}): $! \rightarrow \mathrm{I}l \,||\, \_$,
- Соединить всё вместе $T_{suf} = T_V \circ T_l \circ T_{def}$.

Инфинитив пассивного залога в турецком

Инфинитив пассивного залога

Построить конечный преобразователь, преобразующий турецкий глагольный инфинитив в инфинитив пассивного залога.

- Пассив образуется вставкой суффикса перед *-mek/-mak*.
- Суффикс пассива: *-n* после гласной, *-In* после *l* и *-Il* иначе.
- I: *ı* после *a*, *ı*; *u* после *u*, *o*; *i* после *e*, *i*; *ü* после *ü*, *ö*.

- T_{mark} — вставляет ! перед *-mak/-mek*.
- T_{suf} заменяет маркер на подходящий суффикс.
- T_{fill} заполняет гласную суффикса: $T_{fill} = T_\imath \circ T_u \circ T_i \circ T_U$, where
  - $T_\imath$ проверяет условие для $\imath$: $\mathrm{I} \to \imath \,||\, (a|\imath)C^* \_$.
  - T_u для u: $\mathrm{I} \to u \,||\, (u|o)C^* \_$.
  - T_i для i: $\mathrm{I} \to i \,||\, (e|i)C^* \_$.
  - T_U для $ü$: $\mathrm{I} \to ü \,||\, (ü|ö)C^* \_$.

Инфинитив пассивного залога в турецком

Инфинитив пассивного залога

Построить конечный преобразователь, преобразующий турецкий глагольный инфинитив в инфинитив пассивного залога.

- Пассив образуется вставкой суффикса перед *-mek/-mak*.
- Суффикс пассива: *-n* после гласной, *-In* после *l* и *-Il* иначе.
- I: *ı* после *a*, *ı*; *u* после *u*, *o*; *i* после *e*, *i*; *ü* после *ü*, *ö*.

- T_{mark} — вставляет ! перед *-mak/-mek*.
- T_{suf} заменяет маркер на подходящий суффикс.
- T_{fill} заполняет гласную суффикса: $T_{fill} = T_\imath \circ T_u \circ T_i \circ T_U$, where
  - $T_\imath$ проверяет условие для $\imath$: $\mathrm{I} \to \imath \,||\, (a|\imath)C^*\_$.
  - T_u для u: $\mathrm{I} \to u \,||\, (u|o)C^*\_$.
  - T_i для i: $\mathrm{I} \to i \,||\, (e|i)C^*\_$.
  - T_U для $\ddot{u}$: $\mathrm{I} \to \ddot{u} \,||\, (\ddot{u}|\ddot{o})C^*\_$.
- Финальный ответ:

$$T_{mark} \circ T_{suf} \circ T_{fill}$$

Глагольные формы в языке йоулумни (американдская семья)

основа	герундий	дуратив
caw “кричать”	caw-inay	cawaa-ʔaa-n
cuum “разрушать”	cum-inay	cumuu-ʔaa-n
hoyoo “называть”	hoy-inay	hoyoo-ʔaa-n
diiyl “охранять”	diyl-inay	diyiil-ʔaa-n
ʔilk “петь”	ʔilk-inay	ʔiliik-ʔaa-n
hiwiit “гулять”	hiwt-inay	hiwiit-ʔaa-n

Глагольные формы в языке йоулумни

Глагольные формы в языке йоулумни (американдская семья)

основа	герундий	дуратив
caw “кричать”	caw-inay	cawaa-ʔaa-n
cuum “разрушать”	cum-inay	cumuu-ʔaa-n
hoyoo “называть”	hoy-inay	hoyoo-ʔaa-n
diiyl “охранять”	diyl-inay	diyiil-ʔaa-n
ʔilk “петь”	ʔilk-inay	ʔiliik-ʔaa-n
hiwiit “гулять”	hiwt-inay	hiwiit-ʔaa-n

Глагольные формы в языке йоулумни

Если основа имела вид $\alpha_1 V(V)\alpha_2(V)(V)\alpha_3$, где $\alpha_1, \alpha_2 \in C$, $\alpha_3 \in \{C, \varepsilon\}$, то основа герундия имеет вид $\alpha_1 V \alpha_2 \alpha_3$,

.

Глагольные формы в языке йоулумни (американдская семья)

основа	герундий	дуратив
caw “кричать”	caw-inay	cawaa-ʔaa-n
cuum “разрушать”	cum-inay	cumuu-ʔaa-n
hoyoo “называть”	hoy-inay	hoyoo-ʔaa-n
diiyl “охранять”	diyl-inay	diyiil-ʔaa-n
ʔilk “петь”	ʔilk-inay	ʔiliik-ʔaa-n
hiwiit “гулять”	hiwt-inay	hiwiit-ʔaa-n

Глагольные формы в языке йоулумни

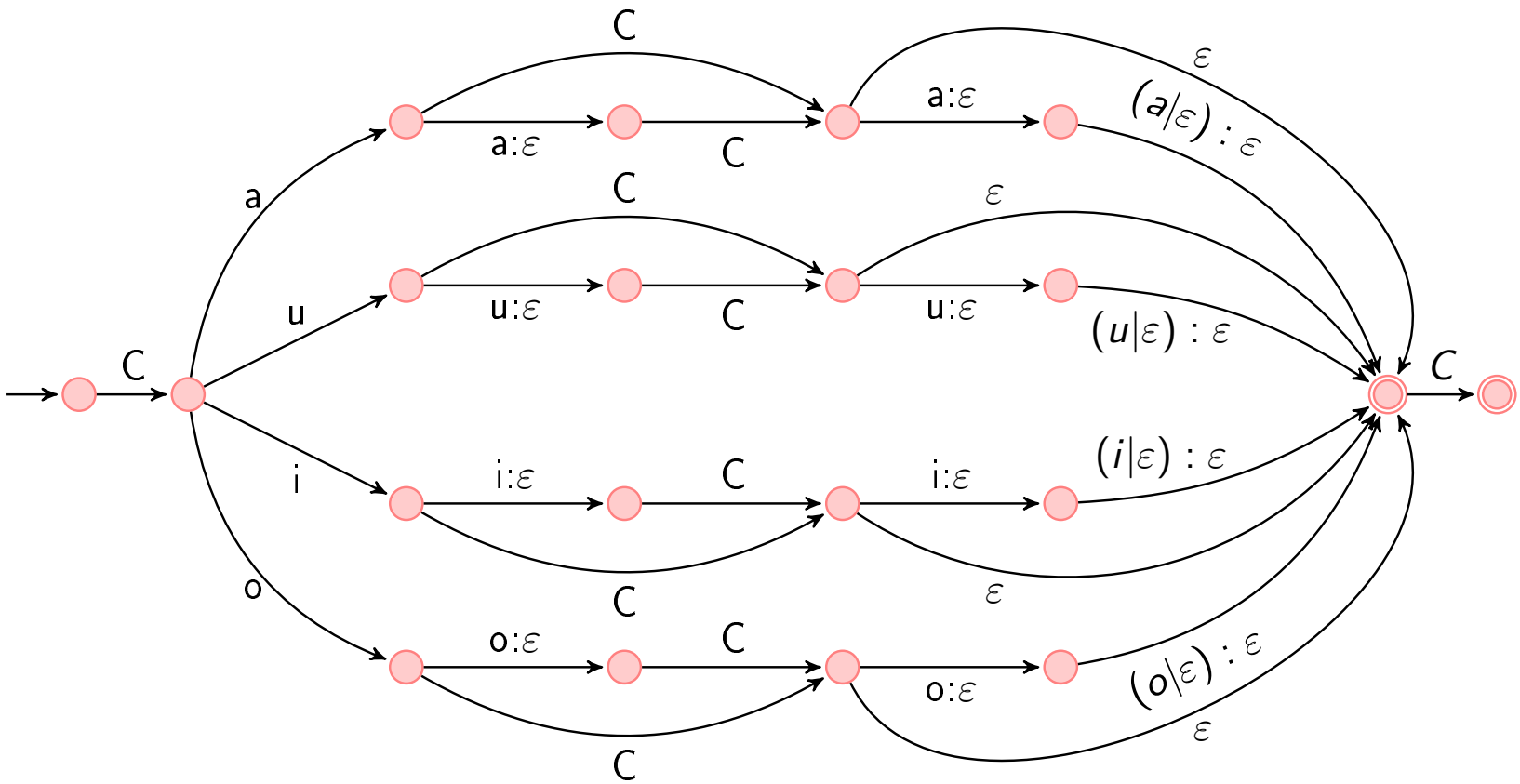
Если основа имела вид $\alpha_1 V(V)\alpha_2(V)(V)\alpha_3$, где $\alpha_1, \alpha_2 \in C$, $\alpha_3 \in \{C, \varepsilon\}$, то основа герундия имеет вид $\alpha_1 V \alpha_2 \alpha_3$, а основа дуратива — $\alpha_1 V \alpha_2 V V \alpha_3$.

Преобразователи для глагольных форм в языке йоулумни

- Основа герундия:

Преобразователи для глагольных форм в языке йоулумни

- Основа герундия:



Преобразователи для глагольных форм в языке йоулумни

- Основа дуратива:

Преобразователи для глагольных форм в языке йоулумни

- Основа дуратива: