Математические модели в морфологии

FOMA: компилятор конечных преобразователей.

Алексей Андреевич Сорокин

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

FOMA: компилятор конечных преобразователей.

- FOMA — компилятор конечных преобразователей.
- Разработана Мансом Халденом в 2009–2015гг., последняя официальная версия 0.9.18 — июнь 2015.

FOMA: компилятор конечных преобразователей.

- FOMA — компилятор конечных преобразователей.
- Разработана Мансом Халденом в 2009–2015гг., последняя официальная версия 0.9.18 — июнь 2015.
- Стабильная версия: `https://code.google.com/archive/p/foma/`,
- Текущая версия `https://github.com/mhulden/foma/`.
- Программа с открытым кодом, написана C++, для отдельных операций есть интерфейс в Python.

FOMA: компилятор конечных преобразователей.

- FOMA — компилятор конечных преобразователей.
- Разработана Мансом Халденом в 2009–2015гг., последняя официальная версия 0.9.18 — июнь 2015.
- Стабильная версия: `https://code.google.com/archive/p/foma/`,
- Текущая версия `https://github.com/mhulden/foma/`.
- Программа с открытым кодом, написана C++, для отдельных операций есть интерфейс в Python.
- Основное применения: компиляция контекстных правил в конечные преобразователи.

FOMA: компилятор конечных преобразователей.

- FOMA — компилятор конечных преобразователей.
- Разработана Мансом Халденом в 2009–2015гг., последняя официальная версия 0.9.18 — июнь 2015.
- Стабильная версия: `https://code.google.com/archive/p/foma/`,
- Текущая версия `https://github.com/mhulden/foma/`.
- Программа с открытым кодом, написана C++, для отдельных операций есть интерфейс в Python.
- Основное применения: компиляция контекстных правил в конечные преобразователи.
- Можно использовать для операций с конечными автоматами.

FOMA: компилятор конечных преобразователей.

- FOMA — компилятор конечных преобразователей.
- Разработана Мансом Халденом в 2009–2015гг., последняя официальная версия 0.9.18 — июнь 2015.
- Стабильная версия: `https://code.google.com/archive/p/foma/`,
- Текущая версия `https://github.com/mhulden/foma/`.
- Программа с открытым кодом, написана C++, для отдельных операций есть интерфейс в Python.
- Основное применения: компиляция контекстных правил в конечные преобразователи.
- Можно использовать для операций с конечными автоматами.
- утилита flookup позволяет сохранять конечные преобразователи в двоичном формате.

FOMA: basic usage

Базовые операции: компиляция контекстных правил.

```
foma[0]: ##replace all a by b
foma[0]: regex a -> b || _ ;
374 bytes. 1 state, 3 arcs, Cyclic.
foma[1]: net
Sigma: ? @ a b
Size: 2.
Net: E20E6CF
Flags: deterministic pruned minimized epsilon_free
Arity: 2
Sfs0:   <a:b> -> fs0, b -> fs0, @ -> fs0.
foma[1]:
```

FOMA: basic usage

Базовые операции: задание преобразователей и их применение к строкам.

```
foma[0]: ##replace all a by b
foma[0]: regex a -> b || _ ;
374 bytes. 1 state, 3 arcs, Cyclic.
foma[1]: down
apply down> bcaba
bcbbb
apply down> bbb
bbb
apply down>
foma[1]: up
apply up> aba
???
apply up> cbdb
cada
cadb
cbda
cbdb
apply up> cdc
```

FOMA: basic usage

Множественное число для заканчивающихся на *y* существительных:

```
foma[0]: ## filter y-ending words
foma[0]: define yFinal ?* y ;
redefined yFinal: 321 bytes. 2 states, 4 arcs, Cyclic.
foma[0]: ## Vowel+y
foma[0]: define Vowel [ a | e | i | o | u ];
redefined Vowel: 413 bytes. 2 states, 5 arcs, 5 paths.
foma[0]: define yVowel [..] -> s || [ .#. | Vowel ] y _ .#. ; ## simply append s after Vowel+y
redefined yVowel: 872 bytes. 4 states, 25 arcs, Cyclic.
foma[0]: define yCons y -> i e s || \Vowel _ .#. ;
redefined yCons: 920 bytes. 6 states, 28 arcs, Cyclic.
foma[0]: ## combine the variants for vowels and consonants
foma[0]: define yChange yFinal .o. yVowel .o. yCons ;
redefined yChange: 936 bytes. 6 states, 29 arcs, Cyclic.
foma[0]: push yChange
936 bytes. 6 states, 29 arcs, Cyclic.
foma[1]: down
apply down> valley
valleys
apply down> ally
allies
apply down> y
ys
apply down> tray
trays
apply down> granny
grannies
```

FOMA: операции с автоматами

Операция	Обозначение
Конкатенация X, Y	XY
Пересечение X, Y	$X\&Y$
Объединение X, Y	$X\|Y$
Разность X, Y	$X - Y$
Итерация X	X^*
Положительная итерация X	X^+
Отрицание X	$\backslash X$
Контекстное ограничение (X возможен только между Y и Z)	$X \rightarrow Y \_ Z$

Операции с автоматами в FOMA

FOMA: операции с автоматами

Операция	Обозначение
Контекстная замена (Замена X на Y в контексте U_V)	$X \rightarrow Y \| \| U\_V$
Композиция X, Y	$X.\mathrm{o}.Y$
Приоритетное объединение X, Y	$X.\mathrm{P}.Y$
Декартово объединение X, Y	$X : Y$
Область определения X	$X.u$
Область значений X	$X.l$
Обратное преобразование к X	$X.i$
Множественные контексты	$X \rightarrow Y \| \| U1\_V1, U2\_V2$
Множественная замена	$X1 \rightarrow Y1, X2 \rightarrow Y2 \| \| U\_V$

Операции с преобразователями в FOMA

FOMA: применение преобразователей

Операция	Обозначение
Сохранить пр-ль в переменную	**define** ⟨var_name⟩ ⟨expression⟩
Поместить пр-ль в стек	**push** ⟨var_name⟩
Поместить выражение в стек	**regex** ⟨expression⟩
Применить верхний пр-ль в стеке в “прямом” направлении	**down** (apply down)
Применить верхний пр-ль в стеке в обратном направлении	**up** (apply up)
Очистить стек	**clear**
Прочитать файл с лексиконом и сохранить пр-ль в переменную	**read** ⟨filename⟩ **define** ⟨var_name⟩
сохранить пр-ль в двоичный файл	**save stack** ⟨filename⟩

Применение преобразователей в FOMA

FOMA: внешнее использование и документация

- Документация: `https://code.google.com/archive/p/foma/wikis`.
- Описание операций: `https://code.google.com/archive/p/foma/wikis/RegularExpressionReference.wiki`.

FOMA: внешнее использование и документация

- Документация: `https://code.google.com/archive/p/foma/wikis`.
- Описание операций: `https://code.google.com/archive/p/foma/wikis/RegularExpressionReference.wiki`.
- Преобразователи можно сохранять в двоичном формате командой **save stack**. Потом их можно запускать из командной строки с помощью утилиты **flookup**.

FOMA: внешнее использование и документация

- Документация: `https://code.google.com/archive/p/foma/wikis`.
- Описание операций: `https://code.google.com/archive/p/foma/wikis/RegularExpressionReference.wiki`.
- Преобразователи можно сохранять в двоичном формате командой **save stack**. Потом их можно запускать из командной строки с помощью утилиты **flookup**.
- Основной способ использования:

  flookup -i -x -w ⟨binary_file⟩ < ⟨input_file⟩ (> ⟨output_file⟩)

- Применяет преобразователь из ⟨binary_file⟩ ко всем строкам в ⟨input_file⟩ и печатает результат (в выходной поток или ⟨output_file⟩).

FOMA: внешнее использование и документация

- Документация: `https://code.google.com/archive/p/foma/wikis`.
- Описание операций: `https://code.google.com/archive/p/foma/wikis/RegularExpressionReference.wiki`.
- Преобразователи можно сохранять в двоичном формате командой **save stack**. Потом их можно запускать из командной строки с помощью утилиты **flookup**.
- Основной способ использования:

  flookup -i -x -w ⟨binary_file⟩ < ⟨input_file⟩ (> ⟨output_file⟩)

- Применяет преобразователь из ⟨binary_file⟩ ко всем строкам в ⟨input_file⟩ и печатает результат (в выходной поток или ⟨output_file⟩).
- Без ключа -x key также печатается входное слово.

FOMA: внешнее использование и документация

- Документация: `https://code.google.com/archive/p/foma/wikis`.
- Описание операций: `https://code.google.com/archive/p/foma/wikis/RegularExpressionReference.wiki`.
- Преобразователи можно сохранять в двоичном формате командой **save stack**. Потом их можно запускать из командной строки с помощью утилиты **flookup**.
- Основной способ использования:

  flookup -i -x -w ⟨binary_file⟩ < ⟨input_file⟩ (> ⟨output_file⟩)
- Применяет преобразователь из ⟨binary_file⟩ ко всем строкам в ⟨input_file⟩ и печатает результат (в выходной поток или ⟨output_file⟩).
- Без ключа -x key также печатается входное слово.
- Документация: `https://code.google.com/archive/p/foma/wikis/FlookupDocumentation.wiki`.

Множественное число существительного в английском

```
### english.foma ###
read lexc irregular.lexc
define IrregularNounPlural;

define Vowel [ a | i | e | o | u | y ];
define Consonant [ b | c | d | f | g | h | j | k | l | m | n | p | q | r | s | t | v | w | x | z ];
define Letter [Vowel | Consonant];
define Word [ Letter ]+;
define NounMark "+N";
define NounNumber "+Sg" | "+Pl";
define Noun Word NounMark NounNumber;

define NounAffixation "+N" "+Sg" -> "" || _ .#., "+N" "+Pl" -> "!" s || _ .#.;
define Sibilant [ x | s | z | c h | s h ];
define sibException ?* Vowel ?* a r c h "!" s ;
define eInsertion [..] -> e || Sibilant _ "!" s .#.;
define checkSibilant [ sibException .P. eInsertion ];
define yReplacement y -> i e || Consonant _ "!" s .#.;
define Cleanup "!" -> "" || _ ;
define RegularNoun [ NounAffixation .o. yReplacement .o. checkSibilant .o. Cleanup ] ;
define Grammar Noun .o. [ IrregularNounPlural .P. RegularNoun ];
push Grammar
```

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

```
# symbol classes
define HardStraightVowel a | I ;
define HardRoundVowel o | u ;
define SoftStraightVowel e | i ;
define SoftRoundVowel O | U ;
define HardVowel HardStraightVowel | HardRoundVowel ;
define SoftVowel SoftStraightVowel | SoftRoundVowel ;
define Vowel HardVowel | SoftVowel ;
define Consonant b | c | C | d | f | g | G | h | j | k | l | m | n | p | r | s | S | t | v | y | z ;
define Letter Consonant | Vowel ;

# contexts for stem
define LastVowelHard HardVowel Consonant* ;
define LastVowelSoft SoftVowel Consonant* ;
define LastVowelHardRound HardRoundVowel Consonant* ;
define LastVowelHardStraight HardStraightVowel Consonant* ;
define LastVowelSoftRound SoftRoundVowel Consonant* ;
define LastVowelSoftStraight SoftStraightVowel Consonant* ;
```

Инфинитив пассивного залога в турецком

```
# infinitive vowel check
define Stem Letter* Vowel Letter* ;
define InfinitiveSuffixInsertion [..] -> E || _ .#. ;
define InfinitiveSuffix [ E -> m a k || HardVowel Consonant* _ .#. ] .o. [ E -> m e k ||
       SoftVowel Consonant* _ .#. ] ;
define SuffixTransform Stem .o. InfinitiveSuffixInsertion .o. InfinitiveSuffix ;
define Infinitive SuffixTransform.l ;
define Input Infinitive "+Pass";

# suffix insertion
define MarkerInsertion [..] -> "!" || _ m [ a | e ] k "+Pass" .#. ;
define MarkerAfterVowel "!" -> I || Vowel _ ;
define MarkerAfterL "!" -> A n || l _ ;
define MarkerAfterAll "!" -> A l || _ ;
define MarkerReplacement MarkerAfterVowel .o. MarkerAfterL .o. MarkerAfterAll ;

# combining all
define VowelFill [ A -> I || LastVowelHardStraight _ ] .o. [ A -> i ||
      LastVowelSoftStraight _ ] .o. [ A -> u || LastVowelHardRound _ ].o. [ A -> U ||
      LastVowelSoftRound _ ] ;
define Cleanup "+Pass" -> "" ;
define Grammar Input .o. MarkerInsertion .o. MarkerReplacement .o. VowelFill ;

push Grammar
```

Глагольные формы в йоулумне

stem	gerund	durative
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 йоулумне (американская семья)

Для основы $\alpha_1 V(V)\alpha_2(V)(V)\alpha_3$, где $\alpha_1, \alpha_2 \in C$, $\alpha_3 \in \{C, \varepsilon\}$:

- Основа герундия равна $\alpha_1 V \alpha_2 \alpha_3$,
- а основа дуратива $\alpha_1 V \alpha_2 VV \alpha_3$.

Глагольные формы в йоулумне

- Преобразователь для йоулумне легко строится вручную.
- Можно ли это сделать в FOMA?
- Первый шаг: задать изменения контекстными правилами.

Глагольные формы в йоулумне

- Преобразователь для йоулумне легко строится вручную.
- Можно ли это сделать в FOMA?
- Первый шаг: задать изменения контекстными правилами.
- Герундий: удалить все гласные, кроме первой слева.
- Левый контекст: $C^*V(C|V)^*$.

Глагольные формы в йоулумне

- Преобразователь для йоулумне легко строится вручную.
- Можно ли это сделать в FOMA?
- Первый шаг: задать изменения контекстными правилами.
- Герундий: удалить все гласные, кроме первой слева.
- Левый контекст: $C^*V(C|V)^*$.
- Дуратив:
  - Удалить вторую гласную в первом слоге (левый контекст $\hat{}C^+V$).

Глагольные формы в йоулумне

- Преобразователь для йоулумне легко строится вручную.
- Можно ли это сделать в FOMA?
- Первый шаг: задать изменения контекстными правилами.
- Герундий: удалить все гласные, кроме первой слева.
- Левый контекст: $C^*V(C|V)^*$.
- Дуратив:
  - Удалить вторую гласную в первом слоге (левый контекст $\hat{}C^+V$).
  - Вставить во второй слог удвоенную гласную из первого.

Глагольные формы в йоулумне

- Преобразователь для йоулумне легко строится вручную.
- Можно ли это сделать в FOMA?
- Первый шаг: задать изменения контекстными правилами.
- Герундий: удалить все гласные, кроме первой слева.
- Левый контекст: $C^*V(C|V)^*$.
- Дуратив:
  - Удалить вторую гласную в первом слоге (левый контекст $\hat{}C^+V$).
  - Вставить во второй слог удвоенную гласную из первого.

Глагольные формы в йоулумне

- Преобразователь для йоулумне легко строится вручную.
- Можно ли это сделать в FOMA?
- Первый шаг: задать изменения контекстными правилами.
- Герундий: удалить все гласные, кроме первой слева.
- Левый контекст: $C^*V(C|V)^*$.
- Дуратив:
  - Удалить вторую гласную в первом слоге (левый контекст $\hat{}C^+V$).
  - Вставить во второй слог удвоенную гласную из первого.
- Проверка равенства гласных для дуратива:
  - Вставить все возможные пары гласных (*aa*, *ii*, *oo*, *uu*).

Глагольные формы в йоулумне

- Преобразователь для йоулумне легко строится вручную.
- Можно ли это сделать в FOMA?
- Первый шаг: задать изменения контекстными правилами.
- Герундий: удалить все гласные, кроме первой слева.
- Левый контекст: $C^*V(C|V)^*$.
- Дуратив:
  - Удалить вторую гласную в первом слоге (левый контекст $\hat{}C^+V$).
  - Вставить во второй слог удвоенную гласную из первого.
- Проверка равенства гласных для дуратива:
  - Вставить все возможные пары гласных (*aa*, *ii*, *oo*, *uu*).
  - Проверить гармонию гласных, перечислив все варианты вида $C^+xC^+x^+C^*$ где x — произвольный гласный.

Двухуровневая морфология

- Морфология с конечным числом состояний основана на конкатенации.
- Идеал: агглютинативные языки (турецкий, финский, etc.).

Двухуровневая морфология

- Морфология с конечным числом состояний основана на конкатенации.
- Идеал: агглютинативные языки (турецкий, финский, etc.).
- Общая схема:
  - Создать слоты для прототипических морфем.

Двухуровневая морфология

- Морфология с конечным числом состояний основана на конкатенации.
- Идеал: агглютинативные языки (турецкий, финский, etc.).
- Общая схема:
  - Создать слоты для прототипических морфем.
  - Заполнить слоты с учётом морфотактики и фонологии.

Двухуровневая морфология

- Морфология с конечным числом состояний основана на конкатенации.
- Идеал: агглютинативные языки (турецкий, финский, etc.).
- Общая схема:
  - Создать слоты для прототипических морфем.
  - Заполнить слоты с учётом морфотактики и фонологии.
- Пример (ниже): турецкое глагольное словоизменение.

Двухуровневая морфология

- Морфология с конечным числом состояний основана на конкатенации.
- Идеал: агглютинативные языки (турецкий, финский, etc.).
- Общая схема:
  - Создать слоты для прототипических морфем.
  - Заполнить слоты с учётом морфотактики и фонологии.
- Пример (ниже): турецкое глагольное словоизменение.
- Моделируемые категории:
  - Залог: пассивный, активный.
  - Время: аорист, прогрессив.

Двухуровневая морфология

- Морфология с конечным числом состояний основана на конкатенации.
- Идеал: агглютинативные языки (турецкий, финский, etc.).
- Общая схема:
  - Создать слоты для прототипических морфем.
  - Заполнить слоты с учётом морфотактики и фонологии.
- Пример (ниже): турецкое глагольное словоизменение.
- Моделируемые категории:
  - Залог: пассивный, активный.
  - Время: аорист, прогрессив.
  - Число: единственное, множественное.

Двухуровневая морфология

- Морфология с конечным числом состояний основана на конкатенации.
- Идеал: агглютинативные языки (турецкий, финский, etc.).
- Общая схема:
  - Создать слоты для прототипических морфем.
  - Заполнить слоты с учётом морфотактики и фонологии.
- Пример (ниже): турецкое глагольное словоизменение.
- Моделируемые категории:
  - Залог: пассивный, активный.
  - Время: аорист, прогрессив.
  - Число: единственное, множественное.
  - Число: $1, 2, 3$.

Турецкое глагольное спряжение

- Формат входа: инфинитив+Voice+Tense+Person+Number.

Турецкое глагольное спряжение

- Формат входа: инфинитив+Voice+Tense+Person+Number.
- +Voice: +Pass/+Act.
- +Tense: +Aor/+Cont.
- +Person: +1/+2/+3.
- +Number: +Sg/+Pl.

Турецкое глагольное спряжение

- Формат входа: инфинитив+Voice+Tense+Person+Number.
- +Voice: +Pass/+Act.
- +Tense: +Aor/+Cont.
- +Person: +1/+2/+3.
- +Number: +Sg/+Pl.
- Структура глагольной словоформы:

$$\langle \text{stem} \rangle \langle \text{VoiceSuf} \rangle \langle \text{Tense} \rangle \langle \text{PersNumSuf} \rangle$$

Турецкое глагольное спряжение

- Глагольная словоформа: $\langle \text{stem} \rangle \langle \text{VoiceSuf} \rangle \langle \text{Tense} \rangle \langle \text{PersNumSuf} \rangle$.

Турецкое глагольное спряжение

- Глагольная словоформа: $\langle \mathrm{stem} \rangle \langle \mathrm{VoiceSuf} \rangle \langle \mathrm{Tense} \rangle \langle \mathrm{PersNumSuf} \rangle$.
- Суффикс пассива: *-n* after vowels, *-In* after *l*, *-Il* otherwise.

Турецкое глагольное спряжение

- Глагольная словоформа: $\langle \mathrm{stem} \rangle \langle \mathrm{VoiceSuf} \rangle \langle \mathrm{Tense} \rangle \langle \mathrm{PersNumSuf} \rangle$.
- Суффикс пассива: *-n* after vowels, *-In* after *l*, *-Il* otherwise.
- Суффикс аориста:
  - *-r* после гласных.
  - *-Ir* после согласных в многосложных основах.
  - *-Ar* после согласных в односложных основах.
  - *-Ir* после 13 односложных исключений.

Турецкое глагольное спряжение

- Глагольная словоформа: $\langle \text{stem} \rangle \langle \text{VoiceSuf} \rangle \langle \text{Tense} \rangle \langle \text{PersNumSuf} \rangle$.
- Суффикс пассива: *-n* after vowels, *-In* after *l*, *-Il* otherwise.
- Суффикс аориста:
  - *-r* после гласных.
  - *-Ir* после согласных в многосложных основах.
  - *-Ar* после согласных в односложных основах.
  - *-Ir* после 13 односложных исключений.
- Суффикс прогрессива:
  - *-Iyor* после согласных.
  - *-yor* после u, ü, i, ı.
  - *-Iyor* после гласных, гласная удаляется.
  - *-iyor* после корней *de-/ye-*, гласная удаляется.

Турецкое глагольное спряжение

- Глагольная словоформа: $\langle \mathrm{stem} \rangle \langle \mathrm{VoiceSuf} \rangle \langle \mathrm{Tense} \rangle \langle \mathrm{PersNumSuf} \rangle$.
- Суффикс пассива: *-n* after vowels, *-In* after *l*, *-Il* otherwise.
- Суффикс аориста:
  - *-r* после гласных.
  - *-Ir* после согласных в многосложных основах.
  - *-Ar* после согласных в односложных основах.
  - *-Ir* после 13 односложных исключений.
- Суффикс прогрессива:
  - *-Iyor* после согласных.
  - *-yor* после u, ü, i, ı.
  - *-Iyor* после гласных, гласная удаляется.
  - *-iyor* после корней *de-/ye-*, гласная удаляется.
- Окончание глагола ($\langle \mathrm{PersNumSuf} \rangle$):

Число \ Лицо	1	2	3
Единств.	-Im	-sIn	-∅
Множ.	-Iz	-sInIz	-lAr

Турецкое глагольное спряжение

Глагольная словоформа:
$\langle \text{stem} \rangle \langle \text{VoiceSuf} \rangle \langle \text{Tense} \rangle \langle \text{PersNumSuf} \rangle$.

okumak+Pass+Prog+1+Pl	okunuyoruz
gelmek+Pass+Aor+2+Sg	gelinirsin
uyumak+Act+Prog+3+Pl	uyuyorlar
izlemek+Act+Prog+3+Pl	izliyorlar
bilmek+Act+Aor+2+Pl	bilirsiniz
görmek+Act+Aor+2+Pl	görersiniz

Турецкое глагольное спряжение

1 шаг: Формируем слоты

```
define Voice "+Act" | "+Pass" ;
define Tense "+Aor" | "+Prog";
define Number "+Sg" | "+Pl" ;
define Person "+1" | "+2" | "+3" ;
define Input Infinitive Voice Tense Person Number;
# deleting -mAk and defining slots
define MarkerInsertion [..] -> "!" || _ m [a | e] k Voice ;
define InfinitiveDeletion m [ a | e ] k -> "" || "!" _ ;
define TensePattern [ [..] -> "!AorSuffix!" || "!" _ ?+ "+Aor" ] .o. [ [..] -> "!ProgSuffix!"
      || "!" _ ?+ "+Prog" ];
define PassivePattern [..] -> "!PassSuffix!" || "!" _ ?+ "+Pass" ;
define Cleanup [ Voice | Tense | "!" ] -> "" ;
```

```
$ flookup -i -w "" turkish_diacr.bin < test.in
okumak+Pass+Prog+1+Pl    oku!PassSuffix!!ProgSuffix!+1+Pl
gelmek+Pass+Aor+2+Sg     gel!PassSuffix!!AorSuffix!+2+Sg
uyumak+Act+Prog+3+Pl     uyu!ProgSuffix!+3+Pl
izlemek+Act+Prog+3+Pl    izle!ProgSuffix!+3+Pl
bilmek+Act+Aor+2+Pl      bil!AorSuffix!+2+Pl
görmek+Act+Aor+2+Pl      gör!AorSuffix!+2+Pl
```

Турецкое глагольное спряжение

2 шаг: показатель залога

```
## passive suffix filling
define Passive1 "!PassSuffix!" -> ı ı || [ Consonant - ı ] _ ;
define Passive2 "!PassSuffix!" -> ı n || ı _ ;
define Passive3 "!PassSuffix!" -> n || Vowel _ ;
define PassiveSuffix Passive1 .o. Passive2 .o. Passive3 ;
```

```
$ flookup -i -w "" turkish_diacr.bin < test.in
okumak+Pass+Prog+1+Pl    okun!ProgSuffix!+1+Pl
gelmek+Pass+Aor+2+Sg     gelIn!AorSuffix!+2+Sg
uyumak+Act+Prog+3+Pl     uyu!ProgSuffix!+3+Pl
izlemek+Act+Prog+3+Pl    izle!ProgSuffix!+3+Pl
bilmek+Act+Aor+2+Pl      bil!AorSuffix!+2+Pl
görmek+Act+Aor+2+Pl      gör!AorSuffix!+2+Pl
```

Турецкое глагольное спряжение

3 шаг: показатель аориста

```
## aorist suffix filling
define PseudoVowel Vowel | I | A ;
read lexc aor_exception.lexc
define AorException;
define Monosyllable Consonant* Vowel Consonant* ;
define AorSuffix0 "!AorSuffix!" -> I r || .#. AorException _ ;
define AorSuffix1 "!AorSuffix!" -> r || PseudoVowel _ ;
define AorSuffix2 "!AorSuffix!" -> A r || .#. Monosyllable _ ;
define AorSuffix3 "!AorSuffix!" -> I r || _ ;
define AorSuffix AorSuffix0 .o. AorSuffix1 .o. AorSuffix2 .o. AorSuffix3 ;
```

```
$ flookup -i -w "" turkish_diacr.bin < test.in
okumak+Pass+Prog+1+Pl    okun!ProgSuffix!+1+Pl
gelmek+Pass+Aor+2+Sg     gelInIr+2+Sg
uyumak+Act+Prog+3+Pl     uyu!ProgSuffix!+3+Pl
izlemek+Act+Prog+3+Pl    izle!ProgSuffix!+3+Pl
bilmek+Act+Aor+2+Pl      bilIr+2+Pl
görmek+Act+Aor+2+Pl      görAr+2+Pl
```

Турецкое глагольное спряжение

4 шаг: показатель прогрессива

```
## progressive suffix filling
define ProgSuffix0 "!ProgSuffix!" -> "!" I y o r || _ ;
## after i, ı, u, ü
define ProgSuffixVowel0 "!" I -> "" || [ u | ü| i | ı] _ ;
## other vowels
define ProgSuffixVowel1 [ a | o | e | ö] "!" -> "" || _ ;
## demek, yemek
define ProgDemek e "!" I -> i || .#. [ d | y ] _ ;
## sonorization
define ProgSonor t -> d || .#. [ g i | e | t a ] _ "!" ;
define ProgCleanup "!" -> "" || _ ;
define ProgSuffix ProgSuffix0 .o. ProgSuffixVowel0 .o. ProgSuffixVowel1 .o. ProgDemek .
    o. ProgSonor .o. ProgCleanup ;
```

```
$ flookup -i -w "" turkish_diacr.bin < test.in
okumak+Pass+Prog+1+Pl    okunIyor+1+Pl
gelmek+Pass+Aor+2+Sg     gelInIr+2+Sg
uyumak+Act+Prog+3+Pl     uyuyor+3+Pl
izlemek+Act+Prog+3+Pl    izlIyor+3+Pl
bilmek+Act+Aor+2+Pl      bilIr+2+Pl
görmek+Act+Aor+2+Pl      görAr+2+Pl
```

Турецкое глагольное спряжение

5 шаг: окончания глагола

```
## ending filling
define Ending1s "+1" "+Sg" -> I m || _ ;
define Ending2s "+2" "+Sg" -> s I n || _ ;
define Ending3s "+3" "+Sg" -> "" || _ ;
define Ending1p "+1" "+Pl" -> I z || _ ;
define Ending2p "+2" "+Pl" -> s I n I z || _ ;
define Ending3p "+3" "+Pl" -> I A r || _ ;
define Ending Ending1s .o. Ending2s .o. Ending3s .o. Ending1p .o. Ending2p .o.
     Ending3p ;
```

```
$ flookup -i -w "" turkish_diacr.bin < test.in
okumak+Pass+Prog+1+Pl    okunIyorIz
gelmek+Pass+Aor+2+Sg     gelInIrsIn
uyumak+Act+Prog+3+Pl     uyuyorlAr
izlemek+Act+Prog+3+Pl    izlIyorlAr
bilmek+Act+Aor+2+Pl      bilIrsInIz
görmek+Act+Aor+2+Pl      görArsInIz
```

Турецкое глагольное спряжение

6 шаг: сингармонизм

```
## Vowel Harmony (left context on output size)
define VowelHarmony [A -> a // LastVowelHard _ ,, A -> e // LastVowelSoft _ ,,
                     I -> ı// LastVowelHardStraight _ ,, I -> i //
                          LastVowelSoftStraight _ ,,
                     I -> u // LastVowelHardRound _ ,, I -> ü//
                          LastVowelSoftRound _] ;

define Fill PassiveSuffix .o. AorSuffix .o. ProgSuffix .o. Ending .o. VowelHarmony ;
define Grammar Input .o. Pattern .o. Fill ;
```

```
$ flookup -i -w "" turkish_diacr.bin < test.in
okumak+Pass+Prog+1+Pl    okunuyoruz
gelmek+Pass+Aor+2+Sg     gelinirsin
uyumak+Act+Prog+3+Pl     uyuyorlar
izlemek+Act+Prog+3+Pl    izliyorlar
bilmek+Act+Aor+2+Pl      bilirsiniz
görmek+Act+Aor+2+Pl      görersiniz
```

Сложности для морфологии с конечным числом состояний

- Морфология с конечным числом состояний годится для агглютинативных языков.
- Явления, вызывающие сложности:
  - Регулярные исключения.

Сложности для морфологии с конечным числом состояний

- Морфология с конечным числом состояний годится для агглютинативных языков.
- Явления, вызывающие сложности:
  - Регулярные исключения.
  - Нерегулярные исключения.

Сложности для морфологии с конечным числом состояний

- Морфология с конечным числом состояний годится для агглютинативных языков.
- Явления, вызывающие сложности:
  - Регулярные исключения.
  - Нерегулярные исключения.
  - Фузия.

Сложности для морфологии с конечным числом состояний

- Морфология с конечным числом состояний годится для агглютинативных языков.
- Явления, вызывающие сложности:
  - Регулярные исключения.
  - Нерегулярные исключения.
  - Фузия.
  - Неконкатенативная морфология (арабский, другие семитские языки).

Сложности для морфологии с конечным числом состояний

- Морфология с конечным числом состояний годится для агглютинативных языков.
- Явления, вызывающие сложности:
  - Регулярные исключения.
  - Нерегулярные исключения.
  - Фузия.
  - Неконкатенативная морфология (арабский, другие семитские языки).
- Не описывается моделью: неограниченная редупликация.