Математические модели в морфологии

FOMA: компилятор конечных преобразователей.

Алексей Андреевич Сорокин

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

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

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
```

Сложности для морфологии с конечным числом состояний

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

Сложности для морфологии с конечным числом состояний

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

Сложности для морфологии с конечным числом состояний

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

Сложности для морфологии с конечным числом состояний

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

Регулярная модель

- Глагольные окончания:

Число	Лицо	*-ar* (tomar)	*-er* (comer)	*-ir* (escribir)
Единств.	1	tomo	como	escribo
	2	tomas	comes	escribes
	3	toma	come	escribe
Множ.	1	tomamos	comemos	escribimos
	2	tomáis	coméis	escribís
	3	toman	comen	escriben

Регулярная модель

- Глагольные окончания:

Число	Лицо	*-ar* (tomar)	*-er* (comer)	*-ir* (escribir)
Единств.	1	tomo	como	escribo
	2	tomas	comes	escribes
	3	toma	come	escribe
Множ.	1	tomamos	comemos	escribimos
	2	tomáis	coméis	escribís
	3	toman	comen	escriben

- Фонетические чередования:
  - В +1+Sg *g* переходит в *j* перед *-er*: emerger → emerjo.

Регулярная модель

- Глагольные окончания:

Число	Лицо	*-ar* (tomar)	*-er* (comer)	*-ir* (escribir)
Единств.	1	tomo	como	escribo
	2	tomas	comes	escribes
	3	toma	come	escribe
Множ.	1	tomamos	comemos	escribimos
	2	tomáis	coméis	escribís
	3	toman	comen	escriben

- Фонетические чередования:
  - В +1+Sg *g* переходит в *j* перед *-er*: emerger → emerjo.
  - В +1+Sg *c* переходит в *zc* перед *-er*/*-ir* и после гласных: *conducir* → *conduzco*, *agradecer* → *agradezco* (хотя *mecer* → *mezo*).

Регулярные чередования

- Также присутствуют регулярные отклонения:

Число	Person	*-o-/-ue-* *contar*	*-e-/-ie-* *sentir*	*-e-/-i-* *servir*
Единств.	1	cuento	siento	sirvo
	2	cuentas	sientes	sirves
	3	cuenta	siente	sirve
Множ.	1	contamos	sentimos	servimos
	2	contáis	sentís	servís
	3	cuentan	sienten	sirven

Регулярные чередования

- Также присутствуют регулярные отклонения:

Число	Person	*-o-/-ue-* *contar*	*-e-/-ie-* *sentir*	*-e-/-i-* *servir*
Единств.	1	cuento	siento	sirvo
	2	cuentas	sientes	sirves
	3	cuenta	siente	sirve
Множ.	1	contamos	sentimos	servimos
	2	contáis	sentís	servís
	3	cuentan	sienten	sirven

- В этих классах куда больше глаголов:
  - *-o-/-ue-*: *morir*, *dormir*, *soler*, *soñar*, ...
  - *-e-/-ie-*: *pensar*, *entender*, *perder*, *preferir*, ...

Регулярные чередования

- Также присутствуют регулярные отклонения:

Число	Person	*-o-/-ue-* *contar*	*-e-/-ie-* *sentir*	*-e-/-i-* *servir*
Единств.	1	cuento	siento	sirvo
	2	cuentas	sientes	sirves
	3	cuenta	siente	sirve
Множ.	1	contamos	sentimos	servimos
	2	contáis	sentís	servís
	3	cuentan	sienten	sirven

- В этих классах куда больше глаголов:
  - *-o-/-ue-*: *morir*, *dormir*, *soler*, *soñar*, ...
  - *-e-/-ie-*: *pensar*, *entender*, *perder*, *preferir*, ...
  - *-e-/-i-*: *pedir*, *vestir*, *elegir*, *expedir*, ...

Нерегулярные чередования

- Также имеются неправильные глаголы:

Число	Лицо	*estar*	*ser*	*haber*
Singular	1	estoy	soy	he
	2	estás	eres	has
	3	está	es	ha
Plural	1	estamos	somos	hemos
	2	estáis	sois	habéis
	3	están	son	han

Нерегулярные чередования

- Также имеются неправильные глаголы:

Число	Лицо	*estar*	*ser*	*haber*
Singular	1	estoy	soy	he
	2	estás	eres	has
	3	está	es	ha
Plural	1	estamos	somos	hemos
	2	estáis	sois	habéis
	3	están	son	han

- Ещё неправильные глаголы: *decir*, *dar*, *ver*, . . .
- У некоторых глаголов неправильная только формы +1+Sg:
  - *traer* $\rightarrow$ *traigo* (также *caer*).
  - *valer* $\rightarrow$ *valgo* (также *salir*, *poner*).
  - *saber* $\rightarrow$ *sé*, *caber* $\rightarrow$ *quepo*.

Нерегулярные чередования

- Также имеются неправильные глаголы:

Число	Лицо	*estar*	*ser*	*haber*
Singular	1	estoy	soy	he
	2	estás	eres	has
	3	está	es	ha
Plural	1	estamos	somos	hemos
	2	estáis	sois	habéis
	3	están	son	han

- Ещё неправильные глаголы: *decir*, *dar*, *ver*, . . .
- У некоторых глаголов неправильная только формы +1+Sg:
  - *traer* → *traigo* (также *caer*).
  - *valer* → *valgo* (также *salir*, *poner*).
  - *saber* → *sé*, *caber* → *quepo*.
- Как это моделировать?

Регулярная модель

Шаг 1: правильные глаголы (+регулярные чередования):

```
define Vowel e | i | é| í| a | u | o | á| ú| ó;
define Cons b | c | d | f | g | h | j | k | l | m | n | ñ| p | q | r | s | t | v | x | y | z ;
define Letter Cons | Vowel ;
define Stem Letter* Vowel Letter* ;
define InfSuffix [ a | i | e ] r ;
define Infinitive Stem InfSuffix ;
define Number "+Sg" | "+Pl" ;
define Person "+1" | "+2" | "+3" ;
define Input Infinitive Number Person;
## phonetic alterations
define ChangeEndCons1 c -> z c || Vowel _ [ e | i ] r "+Sg" "+1";
define ChangeEndCons2 c -> z || [ Cons - z ] _ [ e | i ] r "+Sg" "+1" ;
define ChangeEndCons3 g -> j, g u -> g, q u -> c || _ [ e | i ] r "+Sg" "+1" ;
define UIR [..] -> y || [ Letter - q ] u _ i r [ "+Sg" | "+Pl" "+3" ] ;
define ChangeEnd ChangeEndCons1 .o. ChangeEndCons2 .o. ChangeEndCons3 .o. UIR ;
## endings
define ieInfSuffix [ i | e ] r ;
define PresEnding1s InfSuffix -> o || _ "+Sg" "+1" ;
define PresEnding2s a r -> a s, ieInfSuffix -> e s || _ "+Sg" "+2" ;
define PresEnding3s a r -> a, ieInfSuffix -> e || _ "+Sg" "+3" ;
define PresEnding1p r -> m o s || _ "+Pl" "+1" ;
define PresEnding2p a r -> ái s, e r -> éi s, i r -> ís || _ "+Pl" "+2" ;
define PresEnding3p a r -> a n, ieInfSuffix -> e n || _ "+Pl" "+3" ;
define PresEnding PresEnding1s .o. PresEnding2s .o. PresEnding3s .o. PresEnding1p .o. PresEnding2p .o. PresEnding3p ;
## combining all
define CleanUp [ Person | Number ] -> "" || _ ;
define Regular [ Input .o. ChangeEnd .o. PresEnding ] ;
define Grammar [ IrregularForm .P. Regular ] .o. CleanUp ;
```

Лексиконы для исключений

Исключения задаются в файле с лексиконом:

```
Multichar_Symbols +Sg +Pl +1 +2 +3
LEXICON Root

Verb ; Sg1Verb ;

LEXICON Verb

estar+Sg+1:estoy #;
estar+Sg+2:estás #;
estar+Sg+3:está #;
estar+Pl+3:están #;

ser+Sg+1:soy #;
ser+Sg+2:eres #;
ser+Sg+3:es #;
ser+Pl+1:somos #;
ser+Pl+2:sois #;
ser+Pl+3:son #;
```

Чередование в основе: глагол в испанском

Регулярная модель: применение

```
$ flookup -i -w "" spanish.bin < spanish_test.in
```

caer+Sg+1	caigo	comer+Sg+3	come
ser+Pl+1	somos	correr+Pl+2	corréis
ser+Pl+2	sois	vender+Pl+3	venden
ser+Sg+1	soy	escribir+Sg+2	escribes
estar+Pl+3	están	surgir+Pl+1	surgimos
estar+Sg+2	estás	destruir+Pl+3	destruyen
estar+Sg+3	está	instruir+Sg+2	instruyes
hablar+Sg+1	hablo	cojer+Sg+1	cojo
hablar+Sg+2	hablas	distinguir+Sg+1	distingo
cantar+Pl+1	cantamos	conducir+Sg+1	conduzco

Чередование в основе: глагол в испанском

- Регулярные чередования происходят в нескольких формах (все +Sg и +Pl+3).
- Прописывать все 4 чередования в лексиконе неэкономно.

Чередование в основе: глагол в испанском

- Регулярные чередования происходят в нескольких формах (все +Sg и +Pl+3).
- Прописывать все 4 чередования в лексиконе неэкономно.
- Кроме того, основы после чередования подвержены регулярным фонетическим изменениям:
  - *elegir*+Sg+1 → *elijo* (**eligo*)
  - *seguir*+Sg+1 → *sigo* (**siguo*).

Чередование в основе: глагол в испанском

- Регулярные чередования происходят в нескольких формах (все +Sg и +Pl+3).
- Прописывать все 4 чередования в лексиконе неэкономно.
- Кроме того, основы после чередования подвержены регулярным фонетическим изменениям:
  - *elegir*+Sg+1 → *elijo* (**eligo*)
  - *seguir*+Sg+1 → *sigo* (**siguo*).
- Таким образом, перед добавлением окончаний и применением морфотактических правил нужно преобразовать основы в соответствии с лексиконом.

Чередование в основе: глагол в испанском

- Две ветки для чередований:
  - Первая добавляет *-(i)g-* перед окончанием для нерегулярных форм +Sg+1: (*caer*+Sg+1 $\rightarrow$ *caigo*, *salir*+Sg+1 $\rightarrow$ *salgo*).
  - Вторая обрабатывает чередования гласных (*-o-/-ue-*, *-e-/-ie-*, *-e-/-i-*).
- У первой ветви приоритет выше: (*tener+Sg+1* $\rightarrow$ *tengo*, но *tener+Sg+2* $\rightarrow$ *tienes*, *tener+Sg+3* $\rightarrow$ *tiene*).

Чередование в основе: глагол в испанском

- Две ветки для чередований:
  - Первая добавляет *-(i)g-* перед окончанием для нерегулярных форм +Sg+1: (*caer*+Sg+1 → *caigo*, *salir*+Sg+1 → *salgo*).
  - Вторая обрабатывает чередования гласных (*-o-/-ue-*, *-e-/-ie-*, *-e-/-i-*).
- У первой ветви приоритет выше: (*tener+Sg+1* → *tengo*, но *tener+Sg+2* → *tienes*, *tener+Sg+3* → *tiene*).
- Чтобы избежать псевдоформы (**traiger*) окончание заменяется на специальный символ:

```
!!!first_stem.lexc!!!
LEXICON Root
traer:traiG%!Ending2%! #;
salir:salG%!Ending3%! #;
```

Чередование в основе: глагол в испанском

- Две ветки для чередований:
  - Первая добавляет *-(i)g-* перед окончанием для нерегулярных форм +Sg+1: (*caer*+Sg+1 → *caigo*, *salir*+Sg+1 → *salgo*).
  - Вторая обрабатывает чередования гласных (*-o-/-ue-*, *-e-/-ie-*, *-e-/-i-*).
- У первой ветви приоритет выше: (*tener+Sg+1* → *tengo*, но *tener+Sg+2* → *tienes*, *tener+Sg+3* → *tiene*).
- Чтобы избежать псевдоформы (**traiger*) окончание заменяется на специальный символ:

```
!!!first_stem.lexc!!!
LEXICON Root
traer:traiG%!Ending2%! #;
salir:salG%!Ending3%! #;
```

- Аналогично для второй ветви (*dorm-* → *duerm-*):

```
!!!second_stem.lexc!!!
LEXICON Root
tener:tien%!Ending2%! #;
pedir:pid%!Ending2%! #;
```

Чередование в основе: глагол в испанском

- Глагольные окончания заменяются на маркеры (нужно аналогичным образом изменить правила):

```
define Marker [ a r ] -> "!Ending1!" , [ e r ] -> "!Ending2!" ,
       [ i r ] -> "!Ending3!" || _ Number ;
```

- Чередования заданы в лексиконе:

```
## lexicon for stem changes
read lexc first_stem.lexc
define FirstStem ;
define FirstStemChange FirstStem "+Sg" "+1" ;
read lexc second_stem.lexc
define SecondStem ;
define SecondStemChange SecondStem ["+Sg" ? | "+Pl" "+3" ] ;
define IrregularStemChange FirstStemChange .P. SecondStemChange ;
```

Чередование в основе: глагол в испанском

- Глагольные окончания заменяются на маркеры (нужно аналогичным образом изменить правила):

```
define Marker [ a r ] -> "!Ending1!" , [ e r ] -> "!Ending2!" ,
       [ i r ] -> "!Ending3!" || _ Number ;
```

- Чередования заданы в лексиконе:

```
## lexicon for stem changes
read lexc first_stem.lexc
define FirstStem ;
define FirstStemChange FirstStem "+Sg" "+1" ;
read lexc second_stem.lexc
define SecondStem ;
define SecondStemChange SecondStem ["+Sg" ? | "+Pl" "+3" ] ;
define IrregularStemChange FirstStemChange .P. SecondStemChange ;
```

- Порядок правил управляется приоритетным объединением:

```
define Regular [ Input .o. [IrregularStemChange .P. Marker ] .o. ChangeEnd .o.
      PresEnding ] ;
```

Чередование в основе: глагол в испанском

- Чередования основ:

```
$ flookup -i -w "" spanish_full.bin < spanish_stem.in
```

detraer+Sg+1	detraigo	pensar+Pl+1	pensamos
tener+Pl+1	tenemos	morir+Sg+3	muere
tener+Pl+2	tenéis	morir+Pl+2	morís
tener+Sg+1	tengo	pedir+Pl+3	piden
dormir+Pl+3	duermen	pedir+Sg+2	pides
dormir+Sg+2	duermes	preferir+Pl+1	preferimos
hacer+Sg+1	hago	preferir+Pl+3	prefieren
hacer+Sg+3	hace	preferir+Sg+1	prefiero
pensar+Sg+1	pienso	decir+Sg+3	dice
pensar+Sg+2	piensas	preferir+Sg+1	prefiero

Чередование в основе: глагол в испанском

- Чередования основ:

```
$ flookup -i -w "" spanish_full.bin < spanish_stem.in
```

detraer+Sg+1	detraigo	pensar+Pl+1	pensamos
tener+Pl+1	tenemos	morir+Sg+3	muere
tener+Pl+2	tenéis	morir+Pl+2	morís
tener+Sg+1	tengo	pedir+Pl+3	piden
dormir+Pl+3	duermen	pedir+Sg+2	pides
dormir+Sg+2	duermes	preferir+Pl+1	preferimos
hacer+Sg+1	hago	preferir+Pl+3	prefieren
hacer+Sg+3	hace	preferir+Sg+1	prefiero
pensar+Sg+1	pienso	decir+Sg+3	dice
pensar+Sg+2	piensas	preferir+Sg+1	prefiero

- Необходимо добавить: словообразующие префиксы.
  - *tener* $\rightarrow$ *contener*, *mantener*, *detener*, ...
  - *hacer* $\rightarrow$ *rehacer*, *deshacer*, ...

Фузия: испанский глагол

- чуть больше чередований:

Инфинитив	+1+Sg	герундий
partir	parto	partiendo
imbuir	imbuyo	imbuyendo
destruir	destruyo	destruyendo
delinquir	delinco	delinquiendo
distinguir	distingo	distinguiendo
coger	cojo	cogiendo
agradecer	agradezco	agradeciendo
mecer	mezo	meciendo

Фузия: испанский глагол

- чуть больше чередований:

Инфинитив	+1+Sg	герундий
partir	parto	partiendo
imbuir	imbuyo	imbuyendo
destruir	destruyo	destruyendo
delinquir	delinco	delinquiendo
distinguir	distingo	distinguiendo
coger	cojo	cogiendo
agradecer	agradezco	agradeciendo
mecer	mezo	meciendo

- Происходит фузия между основой и первым гласным окончания.
- Нужно много “фонетических” правил.

Трансфиксация: глагольные формы в арабском

- До этого момента морфемная структура была линейной.

Трансфиксация: глагольные формы в арабском

- До этого момента морфемная структура была линейной.
- Недостаточно для семитских языков (e.g. Arabic):

kataba	“(он) писал+Perf”
kattabat	“(она много) писал+Perf”
yaktubu	“(это+Masc) было написано+Imp”
takattibu	“(это+Fem) было (интенсивно) написано+Imp”

Трансфиксация: глагольные формы в арабском

- До этого момента морфемная структура была линейной.
- Недостаточно для семитских языков (e.g. Arabic):

kataba	“(он) писал+Perf”
kattabat	“(она много) писал+Perf”
yaktubu	“(это+Masc) было написано+Imp”
takattibu	“(это+Fem) было (интенсивно) написано+Imp”

- Корень *k-t-b* состоит из согласных (обычно 3).
- Гласные в основном отражают морфологию.

Трансфиксация: глагольные формы в арабском

- До этого момента морфемная структура была линейной.
- Недостаточно для семитских языков (e.g. Arabic):

kataba	“(он) писал+Perf”
kattabat	“(она много) писал+Perf”
yaktubu	“(это+Masc) было написано+Imp”
takattibu	“(это+Fem) было (интенсивно) написано+Imp”

- Корень *k-t-b* состоит из согласных (обычно 3).
- Гласные в основном отражают морфологию.
- Разным глаголам соответствуют разные трансфиксы:

marida	“(он стал) больным+Perf”
marradat	“(она стала сильно) больной+Perf”
yamradu	“(его) сделали больным+Imp”
tamarridu	“(её) сделали (сильно) больной+Imp”

Арабский: иллюстрационный пример

- Постановка задачи:

$$\langle \text{stem} \rangle \langle \text{Type} \rangle \langle \text{Voice} \rangle \langle \text{Aspect} \rangle \langle \text{Person} \rangle \langle \text{Gender} \rangle \mapsto \langle \text{wordForm} \rangle$$

Арабский: иллюстрационный пример

- Постановка задачи:

$$\langle \text{stem} \rangle \langle \text{Type} \rangle \langle \text{Voice} \rangle \langle \text{Aspect} \rangle \langle \text{Person} \rangle \langle \text{Gender} \rangle \mapsto \langle \text{wordForm} \rangle$$

- Возможные значения:
  - $\langle \text{Type (порода)} \rangle \in \{I, II (\text{интенсив})\}$,
  - $\langle \text{Voice} \rangle \in \{\text{Act}, \text{Pass}\}$,
  - $\langle \text{Aspect} \rangle \in \{\text{Perf}, \text{Imperf}\}$,
  - $\langle \text{Person} \rangle \in \{3\}$,
  - $\langle \text{Gender} \rangle \in \{\text{M}, \text{F}\}$.
- 16 вариантов.

Арабский: иллюстрационный пример

- Постановка задачи:

$$\langle \text{stem} \rangle \langle \text{Type} \rangle \langle \text{Voice} \rangle \langle \text{Aspect} \rangle \langle \text{Person} \rangle \langle \text{Gender} \rangle \mapsto \langle \text{wordForm} \rangle$$

- Возможные значения:
  - $\langle \text{Type (порода)} \rangle \in \{I, II (\text{интенсив})\}$,
  - $\langle \text{Voice} \rangle \in \{\text{Act}, \text{Pass}\}$,
  - $\langle \text{Aspect} \rangle \in \{\text{Perf}, \text{Imperf}\}$,
  - $\langle \text{Person} \rangle \in \{3\}$,
  - $\langle \text{Gender} \rangle \in \{\text{M}, \text{F}\}$.
- 16 вариантов.
- Моделируем лишь один словоизменительный класс (*KTB* “писать”).

Глагольное спряжение в арабском

- Словообразование в арабском (конспект А. А. Зализняка):
- Варианты основы:

Порода	Модель	Пример
I (базовая)	K-T-B	kataba “писать”
II (интенсивная)	K-TT-B	kattaba “много писать”

- Префиксы/суффиксы:

Лицо+Число	перфект (суфф.)	имперфект
+3+Masc	-a	ya- -u
+3+Fem	-at	ta- -u

- Варианты огласовок:

Вид	Залог	Префикс	Огласовка I	Огласовка II
Сов.	Активный		a-a	a-a
Сов.	Пассивный		u-i	u-i
Несов.	Активный	ya-	∅-u	a-i
Несов.	Пассивный	yu-	∅-a	a-a

Арабское спряжение в FOMA: формат входа

- Формат входа:

```
define Vowel [ a | i | u ];
define Consonant [ k | t | b | z | h | r | s | f | m | d | n | y];
define Letter [ Vowel | Consonant ];
define Stem Consonant Consonant Consonant;
define Type [ "+I" | "+II" ];
define Voice ["+Act" | "+Pass"];
define Aspect ["+Perf" | "+Imperf"];
define Person "+3";
define Gender ["+M" | "+F"];
define Input Stem Type Voice Aspect Person Gender;
```

Арабское спряжение в FOMA: формат входа

- Формат входа:

```
define Vowel [ a | i | u ];
define Consonant [ k | t | b | z | h | r | s | f | m | d | n | y];
define Letter [ Vowel | Consonant ];
define Stem Consonant Consonant Consonant;
define Type [ "+I" | "+II" ];
define Voice ["+Act" | "+Pass"];
define Aspect ["+Perf" | "+Imperf"];
define Person "+3";
define Gender ["+M" | "+F"];
define Input Stem Type Voice Aspect Person Gender;
```

- Позиции гласных маркируются цифрами:

```
define 0Insertion [..] -> "0" || .#. _ ;
define 1Insertion [..] -> "1" || "0" Consonant _ ;
define 2Insertion [..] -> "2" || "1" Consonant _ ;
define 3Insertion [..] -> "3" || "2" Consonant _ ;
define PosInsertion 0Insertion .o. 1Insertion .o. 2Insertion .o. 3Insertion;
```

Арабское спряжение в FOMA: огласовки

- Удвоение второй согласной в интенсиве:

```
define CheckTypeI ?* "+I" ?*;
define CheckTypeII ?* "+II" ?*;
define TypeIIDuplication k -> [k k], b -> [b b], t -> [t t], z -> [z z], h
      -> [h h], r -> [r r], s -> [s s], f -> [f f], m -> [m m], d -> [d d], n
       -> [n n] || _ "2";
define StemProcessing [ CheckTypeI ] | [ CheckTypeII .o. TypeIIDuplication ];
```

Арабское спряжение в FOMA: огласовки

- Удвоение второй согласной в интенсиве:

```
define CheckTypeI ?* "+I" ?*;
define CheckTypeII ?* "+II" ?*;
define TypeIIDuplication k -> [k k], b -> [b b], t -> [t t], z -> [z z], h
      -> [h h], r -> [r r], s -> [s s], f -> [f f], m -> [m m], d -> [d d], n
       -> [n n] || _ "2";
define StemProcessing [ CheckTypeI ] | [ CheckTypeII .o. TypeIIDuplication ];
```

- Огласовки:

```
define aaFill "1" -> a, "2" -> a;
define aiFill "1" -> a, "2" -> i;
define uiFill "1" -> u, "2" -> i;
define 0aFill "1" -> [], "2"-> a;
define 0uFill "1" -> [], "2"-> u;
```

Арабское спряжение в FOMA: выбор правила

- Полный перебор в поисках правила:

```
define PerfectActiveFill aaFill;
define ImperfectActiveFill [ CheckTypeI .o. 0uFill ] | [ CheckTypeII .o. aiFill ];
define ActiveFill [CheckPerf .o. PerfectActiveFill] | [CheckImperf .o.
      ImperfectActiveFill];
define PerfectPassiveFill uiFill;
define ImperfectPassiveFill [ CheckTypeI .o. 0aFill ] | [ CheckTypeII .o. aaFill ];
define PassiveFill [CheckPerf .o. PerfectPassiveFill] | [CheckImperf .o.
      ImperfectPassiveFill];
define Fill [CheckPass .o. PassiveFill] | [CheckAct .o. ActiveFill] ;
```

Арабское спряжение в FOMA: выбор правила

- Полный перебор в поисках правила:

```
define PerfectActiveFill aaFill;
define ImperfectActiveFill [ CheckTypeI .o. 0uFill ] | [ CheckTypeII .o. aiFill ];
define ActiveFill [CheckPerf .o. PerfectActiveFill] | [CheckImperf .o.
      ImperfectActiveFill];
define PerfectPassiveFill uiFill;
define ImperfectPassiveFill [ CheckTypeI .o. 0aFill ] | [ CheckTypeII .o. aaFill ];
define PassiveFill [CheckPerf .o. PerfectPassiveFill] | [CheckImperf .o.
      ImperfectPassiveFill];
define Fill [CheckPass .o. PassiveFill] | [CheckAct .o. ActiveFill] ;
```

- То же для префиксов (маркер 0):

```
define 0Prefix "0" -> [];
define yaPrefix "0" -> y a;
define yuPrefix "0" -> y u;
define taPrefix "0" -> t a;
define tuPrefix "0" -> t u;
define PerfectPrefix 0Prefix;
define ImperfectActivePrefix [CheckMasc .o. yaPrefix] | [CheckFem .o. taPrefix] ;
define ImperfectPassivePrefix [CheckMasc .o. yuPrefix] | [CheckFem .o. tuPrefix] ;
define ImperfectPrefix [CheckAct .o. ImperfectActivePrefix] | [CheckPass .o.
      ImperfectPassivePrefix] ;
define Prefix [CheckPerf .o. PerfectPrefix] | [CheckImperf .o. ImperfectPrefix] ;
```

Арабское спряжение в FOMA: выбор правила

- Выбор суффикса (маркер 3):

```
define ImperfectSuffix "3" -> u || _ Type;
define PerfectMascSuffix "3" -> a || _ Type;
define PerfectFemSuffix "3" -> a t || _ Type;
define PerfectSuffix [ CheckMasc .o. PerfectMascSuffix ] | [CheckFem .o.
      PerfectFemSuffix ] ;
define Suffix [ CheckPerf .o. PerfectSuffix ] | [ CheckImperf .o. ImperfectSuffix ];
```

Арабское спряжение в FOMA: выбор правила

- Выбор суффикса (маркер 3):

```
define ImperfectSuffix "3" -> u || _ Type;
define PerfectMascSuffix "3" -> a || _ Type;
define PerfectFemSuffix "3" -> a t || _ Type;
define PerfectSuffix [ CheckMasc .o. PerfectMascSuffix ] | [CheckFem .o.
      PerfectFemSuffix ] ;
define Suffix [ CheckPerf .o. PerfectSuffix ] | [ CheckImperf .o. ImperfectSuffix ];
```

- Объединение этапов:

```
define Cleanup Type | Voice | Aspect | Person | Gender -> [] ;
define Grammar Input .o. PosInsertion .o. StemProcessing .o. Fill .o. Prefix .o.
      Suffix .o. Cleanup;
```

Арабское спряжение в FOMA: выбор правила

- Выбор суффикса (маркер 3):

```
define ImperfectSuffix "3" -> u || _ Type;
define PerfectMascSuffix "3" -> a || _ Type;
define PerfectFemSuffix "3" -> a t || _ Type;
define PerfectSuffix [ CheckMasc .o. PerfectMascSuffix ] | [CheckFem .o.
      PerfectFemSuffix ] ;
define Suffix [ CheckPerf .o. PerfectSuffix ] | [ CheckImperf .o. ImperfectSuffix ];
```

- Объединение этапов:

```
define Cleanup Type | Voice | Aspect | Person | Gender -> [] ;
define Grammar Input .o. PosInsertion .o. StemProcessing .o. Fill .o. Prefix .o.
      Suffix .o. Cleanup;
```

- В реальности всё ещё сложнее.

Арабское спряжение в FOMA: выбор правила

- Выбор суффикса (маркер 3):

```
define ImperfectSuffix "3" -> u || _ Type;
define PerfectMascSuffix "3" -> a || _ Type;
define PerfectFemSuffix "3" -> a t || _ Type;
define PerfectSuffix [ CheckMasc .o. PerfectMascSuffix ] | [CheckFem .o.
      PerfectFemSuffix ] ;
define Suffix [ CheckPerf .o. PerfectSuffix ] | [ CheckImperf .o. ImperfectSuffix ];
```

- Объединение этапов:

```
define Cleanup Type | Voice | Aspect | Person | Gender -> [] ;
define Grammar Input .o. PosInsertion .o. StemProcessing .o. Fill .o. Prefix .o.
      Suffix .o. Cleanup;
```

- В реальности всё ещё сложнее.
- Тем не менее, один из первых языков с автоматной моделью морфологии (Beesley, 1990).

Редупликация: математические свойства

- Область определения конечного преобразования — автоматный язык.
- Область значений конечного преобразования — автоматный язык.

Редупликация: математические свойства

- Область определения конечного преобразования — автоматный язык.
- Область значений конечного преобразования — автоматный язык.

Теорема

Язык квадратов $\{ww|w \in \Sigma^*\}$ неавтоматен при $|\Sigma| \geqslant 2$ (то есть уже над $\{a, b\}$).

Редупликация: математические свойства

- Область определения конечного преобразования — автоматный язык.
- Область значений конечного преобразования — автоматный язык.

Теорема

Язык квадратов $\{ww|w \in \Sigma^*\}$ неавтоматен при $|\Sigma| \geqslant 2$ (то есть уже над $\{a, b\}$).

Доказательство

- От противного: пусть данный язык автоматен, тогда для него существует ДКА с конечным числом состояний.

Редупликация: математические свойства

- Область определения конечного преобразования — автоматный язык.
- Область значений конечного преобразования — автоматный язык.

Теорема

Язык квадратов $\{ww|w \in \Sigma^*\}$ неавтоматен при $|\Sigma| \geqslant 2$ (то есть уже над $\{a, b\}$).

Доказательство

- От противного: пусть данный язык автоматен, тогда для него существует ДКА с конечным числом состояний.
- Слов вида a^k бесконечное число $\Leftrightarrow$ какие-то два из них приводят в одно состояние.
- Пусть это a^r и a^s, $r \neq s$.

Редупликация: математические свойства

- Область определения конечного преобразования — автоматный язык.
- Область значений конечного преобразования — автоматный язык.

Теорема

Язык квадратов $\{ww|w \in \Sigma^*\}$ неавтоматен при $|\Sigma| \geqslant 2$ (то есть уже над $\{a, b\}$).

Доказательство

- От противного: пусть данный язык автоматен, тогда для него существует ДКА с конечным числом состояний.
- Слов вида a^k бесконечное число $\Leftrightarrow$ какие-то два из них приводят в одно состояние.
- Пусть это a^r и a^s, $r \neq s$.
- Продолжим их словом b^r, тогда $a^r b^r$ и $a^s b^r$ тоже приводят в одно состояние.

Редупликация: математические свойства

- Область определения конечного преобразования — автоматный язык.
- Область значений конечного преобразования — автоматный язык.

Теорема

Язык квадратов $\{ww|w \in \Sigma^*\}$ неавтоматен при $|\Sigma| \geqslant 2$ (то есть уже над $\{a, b\}$).

Доказательство

- От противного: пусть данный язык автоматен, тогда для него существует ДКА с конечным числом состояний.
- Слов вида a^k бесконечное число $\Leftrightarrow$ какие-то два из них приводят в одно состояние.
- Пусть это a^r и a^s, $r \neq s$.
- Продолжим их словом b^r, тогда $a^r b^r$ и $a^s b^r$ тоже приводят в одно состояние.
- Но это невозможно: одно из них в языке, другое — нет.

Редупликация: невозможность реализации

- Если б редупликация задавалась конечным преобразованием, то её область значений была бы автоматной.

Редупликация: невозможность реализации

- Если б редупликация задавалась конечным преобразованием, то её область значений была бы автоматной.
- Это не так: шаблон ww не автоматен.

Редупликация: невозможность реализации

- Если б редупликация задавалась конечным преобразованием, то её область значений была бы автоматной.
- Это не так: шаблон *ww* не автоматен.
- Невозможно задать:

Редупликация в языке бамбара (семья манде):

wulu “собака”	*wulu-o-wulu* “любая собака”
wulu-nyinina “видящий собаку”	*wulu-nyinina-o-wulu-nyinina* “кто угодно, видящий собаку”

Редупликация: невозможность реализации

- Если б редупликация задавалась конечным преобразованием, то её область значений была бы автоматной.
- Это не так: шаблон *ww* не автоматен.
- Невозможно задать:

  Редупликация в языке бамбара (семья манде):

wulu “собака”	*wulu-o-wulu* “любая собака”
wulu-nyinina “видящий собаку”	*wulu-nyinina-o-wulu-nyinina* “кто угодно, видящий собаку”

- Ограниченная редупликация (только первый слог) тоже не задаётся.