Finite-State Programming

Some Examples

Finite-state "programming"

Function	Function on (set of) strings
Source code	Regular expression
Object code	Finite state machine
Compiler	Regexp compiler
Optimization of object code	Determinization, minimization, pruning

Finite-state “programming”

Function composition	(Weighted) composition
Higher-order function	Operator
Function inversion (available in Prolog)	Function inversion
Structured programming	Ops + small regexps

Finite-state "programming"

Parallelism	Apply to set of strings
Nondeterminism	Nondeterminism
Stochasticity	Prob.-weighted arcs

Some Xerox Extensions

`$`	containment
`=>`	restriction
`-> @->`	replacement

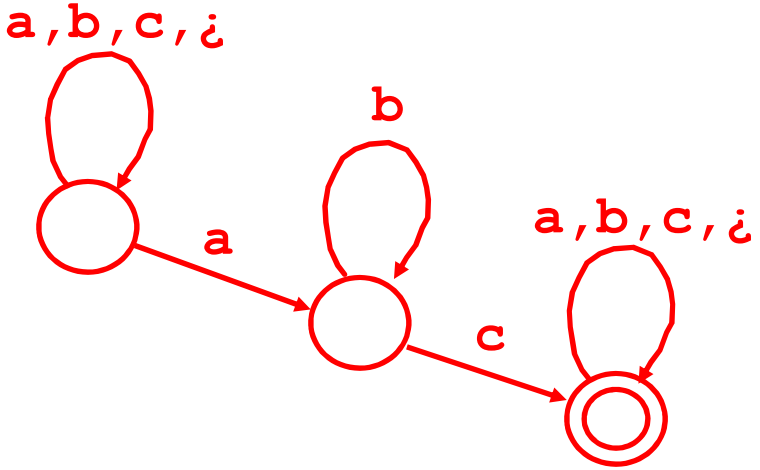
Make it easier to describe complex languages and relations without extending the formal power of finite-state systems.

Containment

`$[ab*c]`

"Must contain a substring that matches `ab*c`."

Accepts **xxxacyy**
Rejects **bcba**



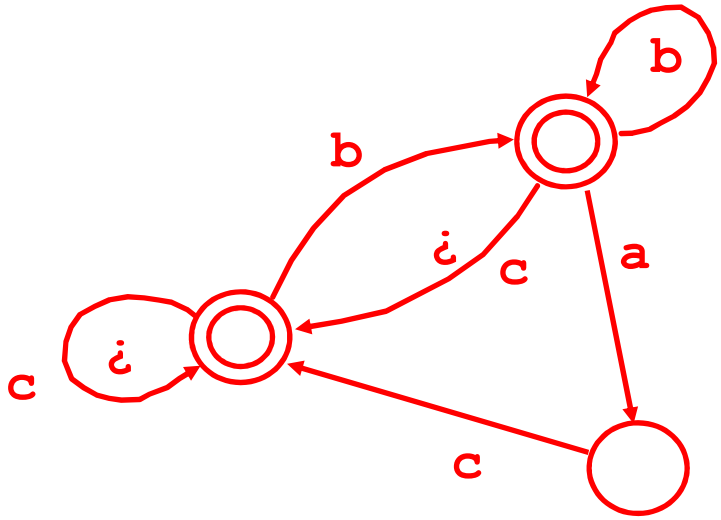
Warning: `?` in regexps means "any character at all."
But `¿` in machines means "any character not explicitly mentioned anywhere in the machine."

`?* [ab*c] ?*`

Equivalent expression

Restriction

```
a => b _ c
```



"Any **a** must be preceded by **b** and followed by **c**."

Accepts **bacbbacde**

Rejects **baca**

contains **a** not preceded by **b**

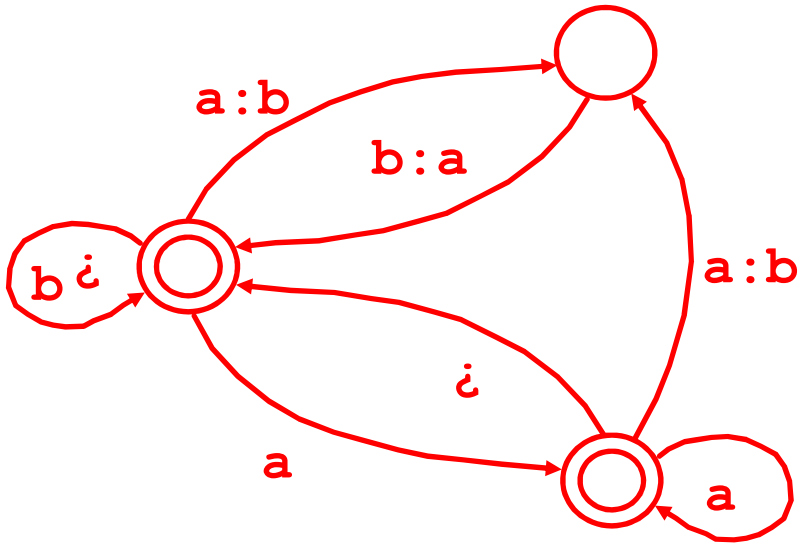
contains **a** not followed by **c**

```
~[~[?* b] a ?*] & ~[?* a ~[c ?*]]
```

Equivalent expression

Replacement

```
a b -> b a
```



"Replace 'ab' by 'ba'."

Transduces **abcdbaba**
to **bacdbbaa**

```
[~$[a b] [[a b] .x. [b a]]]* ~$[a b]
```

Equivalent expression

Replacement is Nondeterministic

```
a b -> b a | x
```

"Replace 'ab' by 'ba' or 'x', nondeterministically."

Transduces **abcdbaba**
to {**bacdbbaa**, **bacdbxa**, **xcdbbaa**, **xcdbxa**}

Replacement is Nondeterministic

```
[ a b -> b a | x ] .o. [ x => _ c ]
```

"Replace 'ab' by 'ba' or 'x', nondeterministically."

Transduces **abcdbaba**
to {**bacdbbaa**, ~~**bacdbxa**~~, **xcdbbaa**, ~~**xcdbxa**~~}

Replacement is Nondeterministic

a b | b | b a | a b a -> x

applied to “aba”

Four overlapping substrings match; we haven’t told it which one to replace so it chooses nondeterministically

a **b** a	a **b a**	**a b** a	**a b a**
a **x** a	a **x**	**x** a	**x**

More Replace Operators

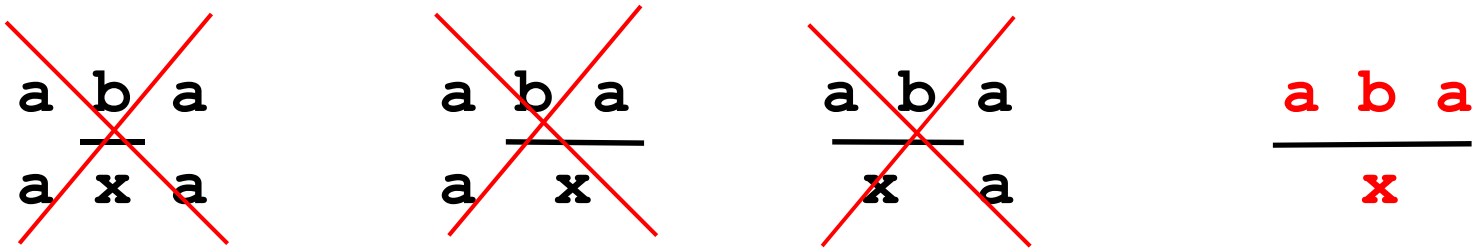
- Optional replacement: `a b (->) b a`

- Directed replacement
  - guarantees a unique result by constraining the factorization of the input string by
    - Direction of the match (rightward or leftward)
    - Length (longest or shortest)

@-> Left-to-right, Longest-match Replacement

```
a b | b | b a | a b a @-> x
```

applied to “aba”

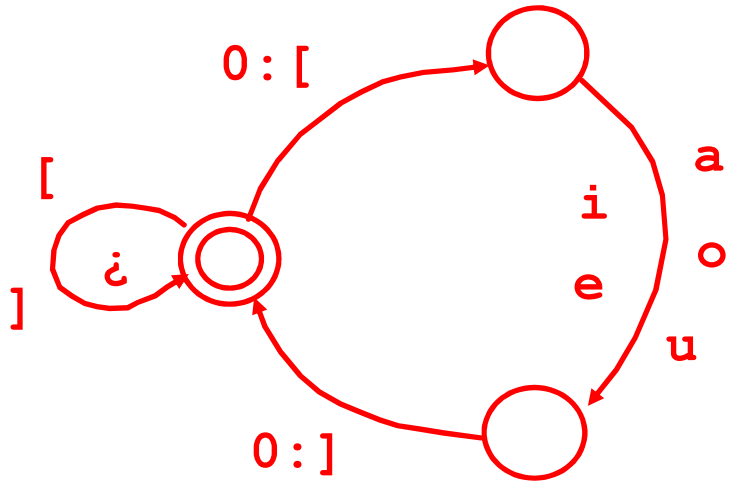


@-> left-to-right, longest match
@> left-to-right, shortest match
->@ right-to-left, longest match
>@ right-to-left, shortest match

Using "..." for marking

```
a|e|i|o|u -> [ ... ]
```

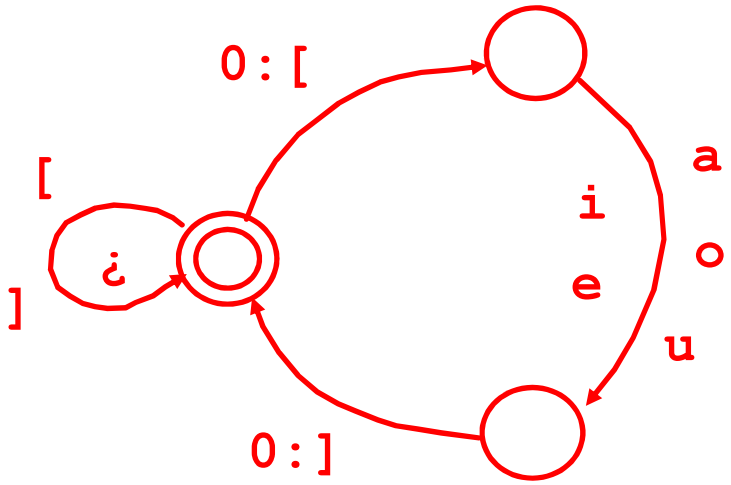
```
p o t a t o
p[o]t[a]t[o]
```



Note: actually have to write as -> %[... %]
or -> "[" ... "]"
since [] are parens in the regexp language

Using "..." for marking

```
a|e|i|o|u -> [ ... ]
```




```
p o t a t o
p[o]t[a]t[o]
```

Which way does the FST transduce potatoe?

```
p o t a t o  e
p[o]t[a]t[o][e]
```

vs.

```
p o t a t o e
p[o]t[a]t[o e]
```

How would you change it to get the other answer?

Example: Finnish Syllabification

```
define C [ b | c | d | f ...
define V [ a | e | i | o | u ];
```

```
[C* V+ C*] @-> ... "-" || _ [C V]
```

"Insert a hyphen after the longest instance of the **C* V+ C*** pattern in front of a **C V** pattern." why?

```
s t r u k   t u   r a   l i s   m i
s t r u k - t u - r a - l i s - m i
```

slide courtesy of L. Karttunen

Conditional Replacement

`A -> B` `L _ R`

Replacement Context

The relation that replaces **`A`** by **`B`** between **`L`** and **`R`** leaving everything else unchanged.

Sources of complexity:

- Replacements and contexts may overlap
- Alternative ways of interpreting "between left and right."

Hand-Coded Example: Parsing Dates

slide courtesy of L. Karttunen

Today is [Tuesday, July 25, 2000]. Best result

Today is Tuesday, [July 25, 2000].
Today is [Tuesday, July 25], 2000.
Today is Tuesday, [July 25], 2000.
Today is [Tuesday], July 25, 2000. Bad results

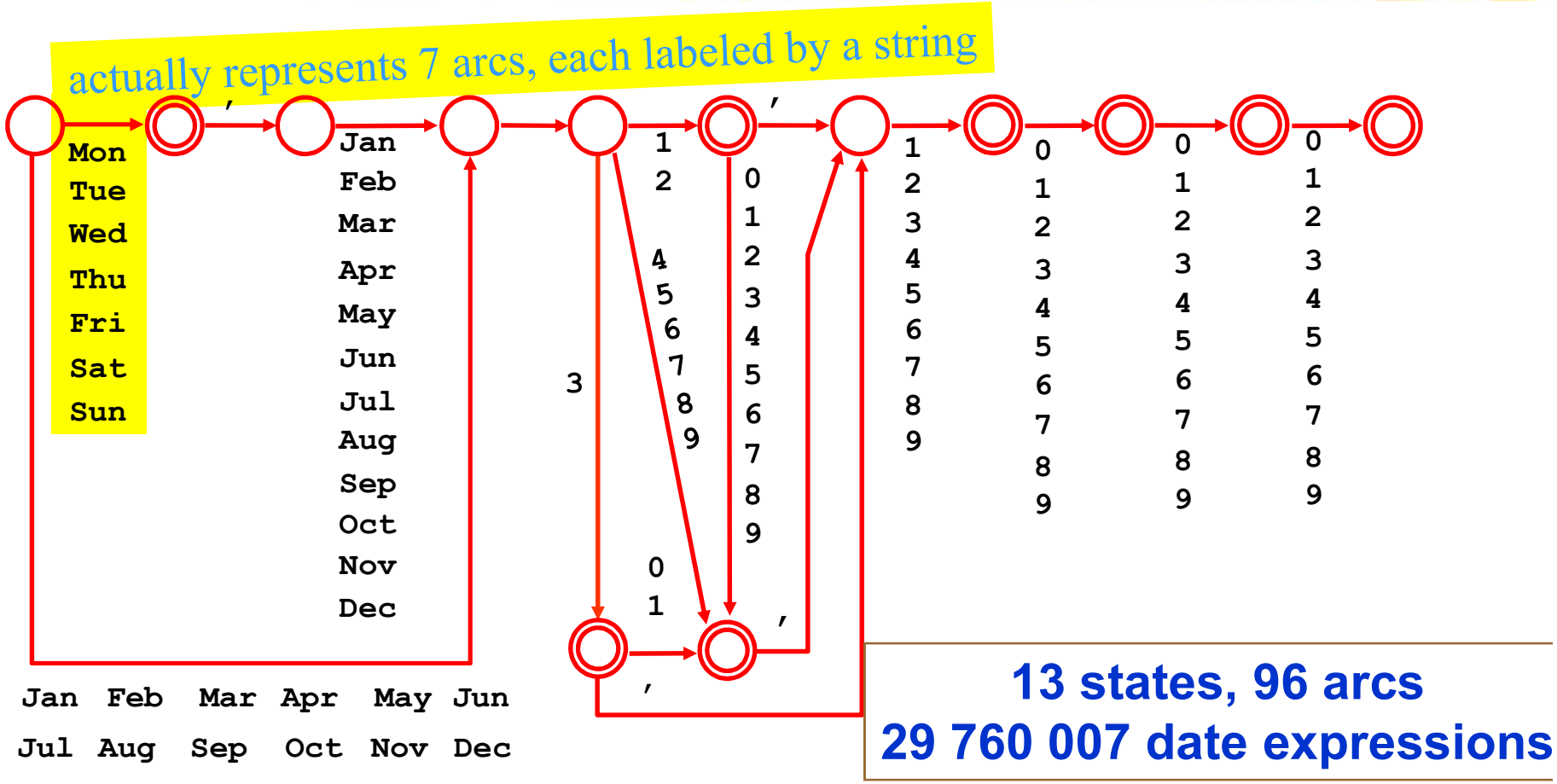
Need left-to-right, longest-match constraints.

Source code: Language of Dates

```
Day = Monday | Tuesday |  ... | Sunday
Month = January | February | ... | December
Date = 1 | 2 | 3 | ... | 3 1
Year = %0To9 (%0To9 (%0To9 (%0To9))) - %0?*
                                  from 1 to 9999
AllDates =  Day | (Day ", ") Month " " Date
  (", " Year))
```

Object code: All Dates from 1/1/1 to 12/31/9999

slide courtesy of L. Karttunen



Parser for Dates

```
AllDates @-> "[DT " ... "]"
```

Compiles into an unambiguous transducer (23 states, 332 arcs).

Xerox left-to-right replacement operator

Today is [DT Tuesday, July 25, 2000] because yesterday was [DT Monday] and it was [DT July 24] so tomorrow must be [DT Wednesday, July 26] and not [DT July 27] as it says on the program.

slide courtesy of L. Karttunen

Problem of Reference

Valid dates

Tuesday, July 25, 2000

Tuesday, February 29, 2000

Monday, September 16, 1996

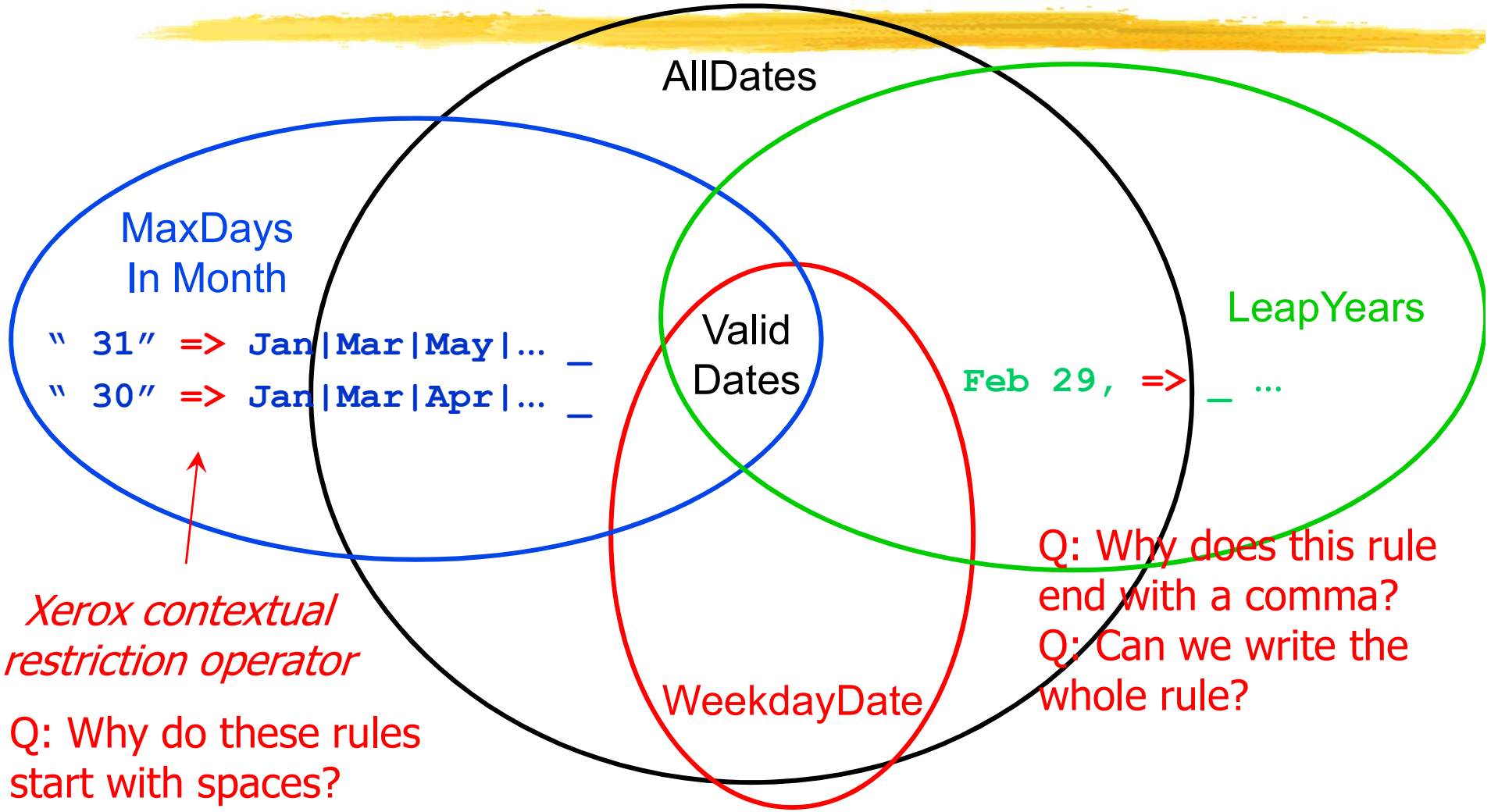
Invalid dates

Wednesday, April 31, 1996

Thursday, February 29, 1900

Tuesday, July 26, 2000

Refinement by Intersection



Xerox contextual restriction operator

Q: Why do these rules start with spaces? (And is it enough?)

Q: Why does this rule end with a comma?

Q: Can we write the whole rule?

Q: LeapYears made use of a "divisible by 4" FSA; can we build a "divisible by 7" FSA (base-ten input)?

Defining Valid Dates

```
AllDates
&
MaxDaysInMonth
&
LeapYears
&
WeekdayDates
```

AllDates: 13 states, 96 arcs
29 760 007 date expressions

= ValidDates

ValidDates: 805 states, 6472 arcs
7 307 053 date expressions

Parser for Valid and Invalid Dates

```
[AllDates - ValidDates] @-> "[ID " ... "]"
                          ,
        ValidDates @-> "[VD " ... "]"
```

2688 states,
20439 arcs

Comma creates a single FST that does left-to-right longest match against either pattern

Today is [VD Tuesday, July 25, 2000],
not [ID Tuesday, July 26, 2000].

valid date

invalid date

More Engineering Applications

- Markup
  - Dates, names, places, noun phrases; spelling/grammar errors?
  - Hyphenation
  - Informative templates for information extraction (FASTUS)
  - Word segmentation (use probabilities!)
  - Part-of-speech tagging (use probabilities – maybe!)
- Translation
  - Spelling correction / edit distance
  - Phonology, morphology: series of little fixups? constraints?
  - Speech
  - Transliteration / back-transliteration
  - Machine translation?
- Learning ...

FASTUS – Information Extraction

Appelt et al, 1992-?

Input: Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan. The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with …

Output:

Relationship:	TIE-UP
Entities:	"Bridgestone Sports Co."
	"A local concern"
	"A Japanese trading house"
Joint Venture Company:	"Bridgestone Sports Taiwan Co."
Amount:	NT$20000000

FASTUS: Successive Markups

(details on subsequent slides)

Tokenization

.o.

Multiwords

.o.

Basic phrases (noun groups, verb groups …)

.o.

Complex phrases

.o.

Semantic Patterns

.o.

Merging different references

FASTUS: Tokenization

- Spaces, hyphens, etc.
- wouldn't → would not
- their → them 's
- company. → company .
  but
  Co. → Co.

FASTUS: Multiwords

- "set up"
- "joint venture"
- "San Francisco Symphony Orchestra," "Canadian Opera Company"

- … use a specialized regexp to match musical groups.
- … what kind of regexp would match company names?

FASTUS : Basic phrases

Output looks like this (no nested brackets!):

… [NG it] [VG had set_up] [NG a joint_venture] [Prep in] …

Company Name:	Bridgestone Sports Co.
Verb Group:	said
Noun Group:	Friday
Noun Group:	it
Verb Group:	had set up
Noun Group:	a joint venture
Preposition:	in
Location:	Taiwan
Preposition:	with
Noun Group:	a local concern

FASTUS: Noun Groups

Build FSA to recognize phrases like

- approximately 5 kg
- more than 30 people
- the newly elected president
- the largest leftist political force
- a government and commercial project

Use the FSA for left-to-right longest-match markup

What does FSA look like? See next slide ...

FASTUS: Noun Groups

Described with a kind of *non-recursive* CFG ...
(a regexp can include names that stand for other regexps)

NG → Pronoun | Time-NP | Date-NP
NG → (Det) (Adjs) HeadNouns
...
Adjs → sequence of adjectives maybe with commas, conjunctions, adverbs
...
Det → DetNP | DetNonNP
DetNP → detailed expression to match "the only five, another three, this, many, hers, all, the most ..."
...

FASTUS: Semantic patterns

BusinessRelationship =
  NounGroup(Company/ies) VerbGroup(Set-up)
  NounGroup(JointVenture) with
  NounGroup(Company/ies) | …

ProductionActivity =
  VerbGroup(Produce) NounGroup(Product)

NounGroup(Company/ies) → NounGroup & …
  is made easy by the processing done at a previous level

Use this for spotting references to put in the database.