# or even...


clear evidence that MT is used in real life.

# Context in Translation

xiaoxin gou
小心 狗 <=> be aware of dog

小心 VP <=> be careful not to VP

小心 NP <=> be careful of NP

xiaoxin
小心 X <=> be careful not to X

fluency problem
(*n*-gram)

(SCFG)



syntax should help...

# How do people translate?

1. understand the source language sentence
2. generate the target language translation

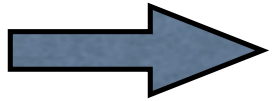
| 布什 | 与 | 沙龙 | 举行 | 了 | 会谈 |
|---|---|---|---|---|---|
| *Bùshí* | *yu* | *Shalóng* | *juxíng* | *le* | *huìtán* |
| Bush | and/ with | Sharon | hold | [*past.*] | meeting |

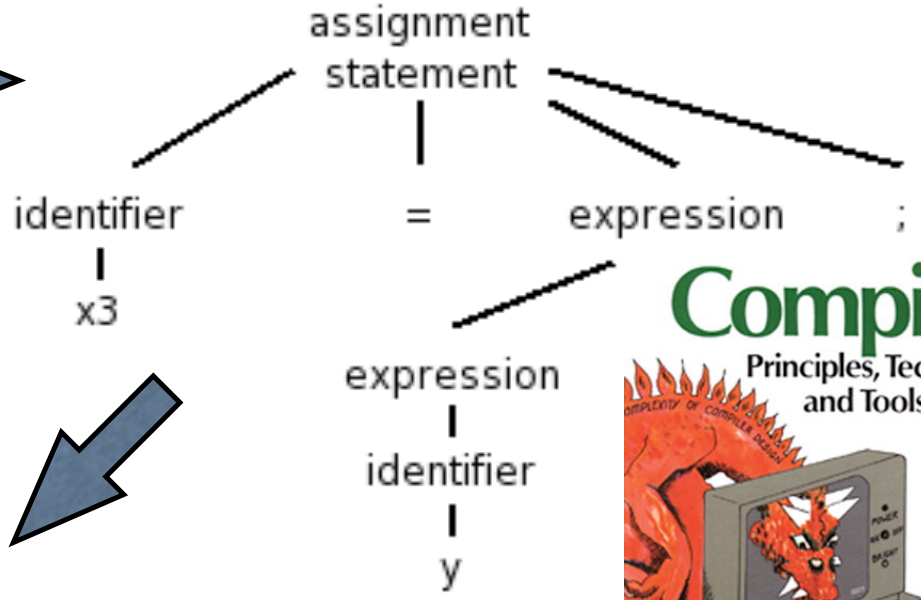
"Bush held a meeting with Sharon"

# How do compilers translate?

1. parse high-level language program into a syntax tree
2. generate intermediate or machine code accordingly

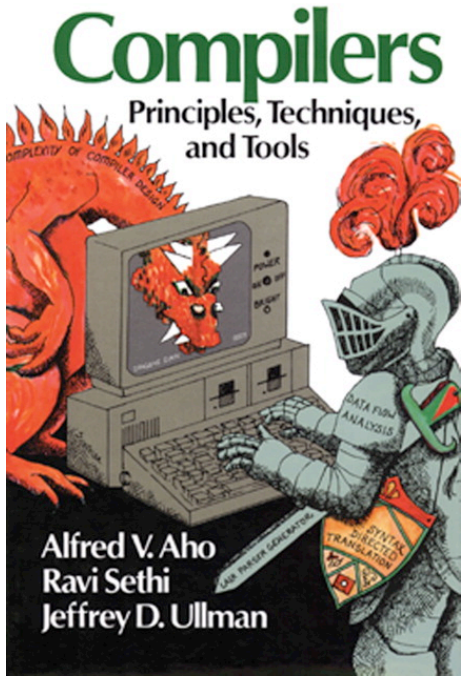
```
x3 = y + 3;
```








```
LD      R1,  id2
ADDF    R1,  R1, #3.0   // add float
RTOI    R2,  R1         // real to int
ST      id1, R2
```




syntax-directed translation (~1960)

Penn
University of Pennsylvania

# Syntax-Directed Machine Translation

1. parse the source-language sentence into a tree
2. recursively convert it into a target-language sentence

*Bùshí yǔ Shālóng jǔxíng le huìtán*

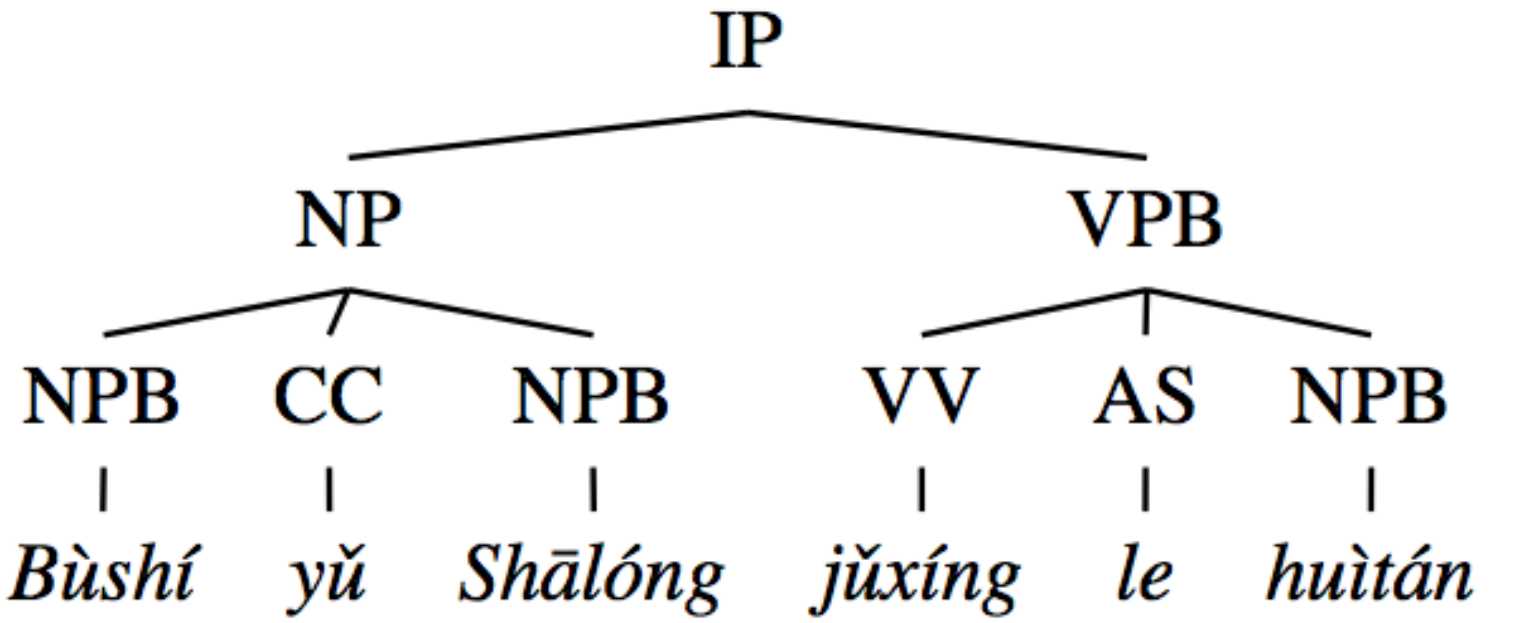
Bush and/with Sharon hold [*past.*] meeting



(Irons 1961; Lewis, Stearns 1968; Aho, Ullman 1972) ==> (Huang, Knight, Joshi 2006)

# Tree-based Translation

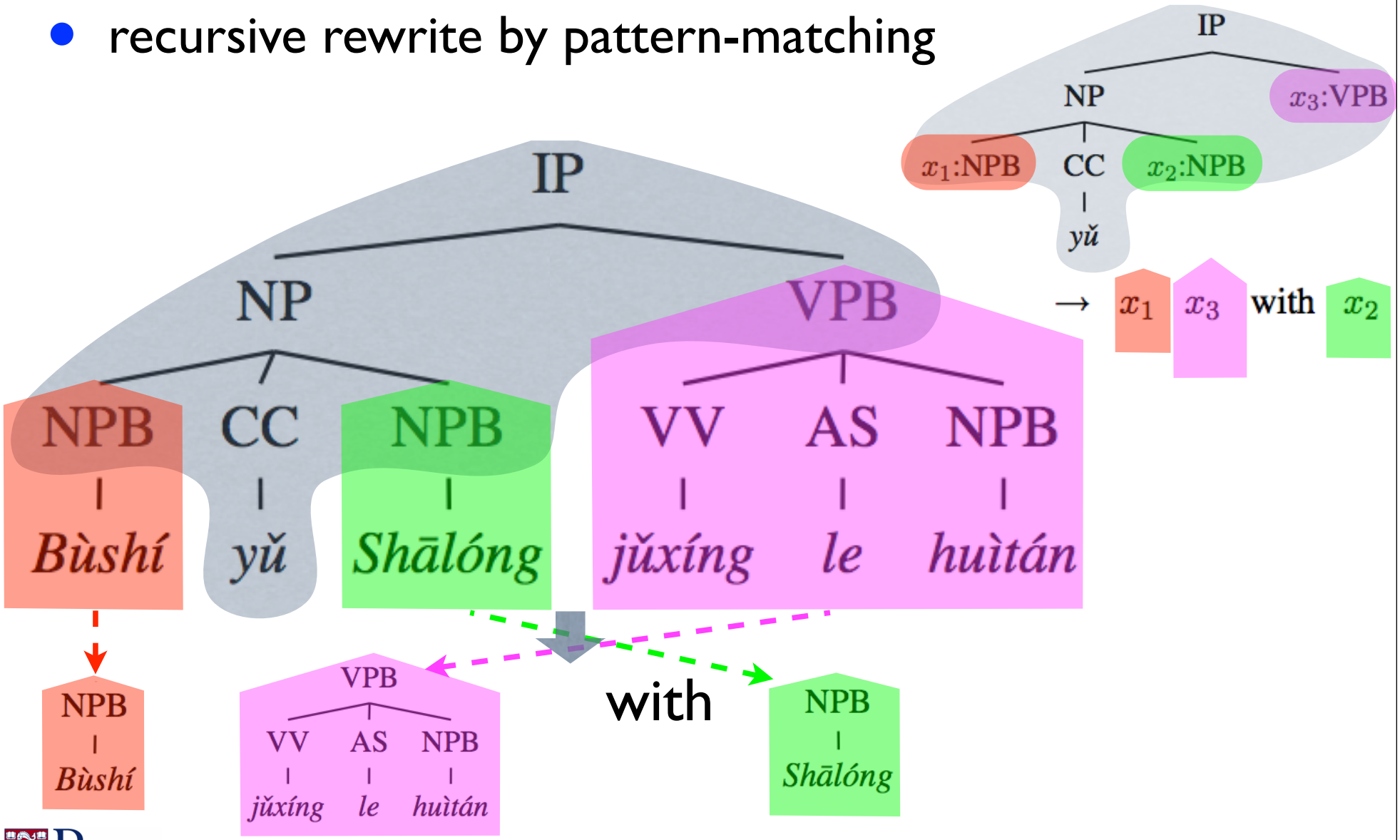
- get 1-best parse tree; then convert to English




"Bush held a meeting with Sharon"

# Tree-based Translation

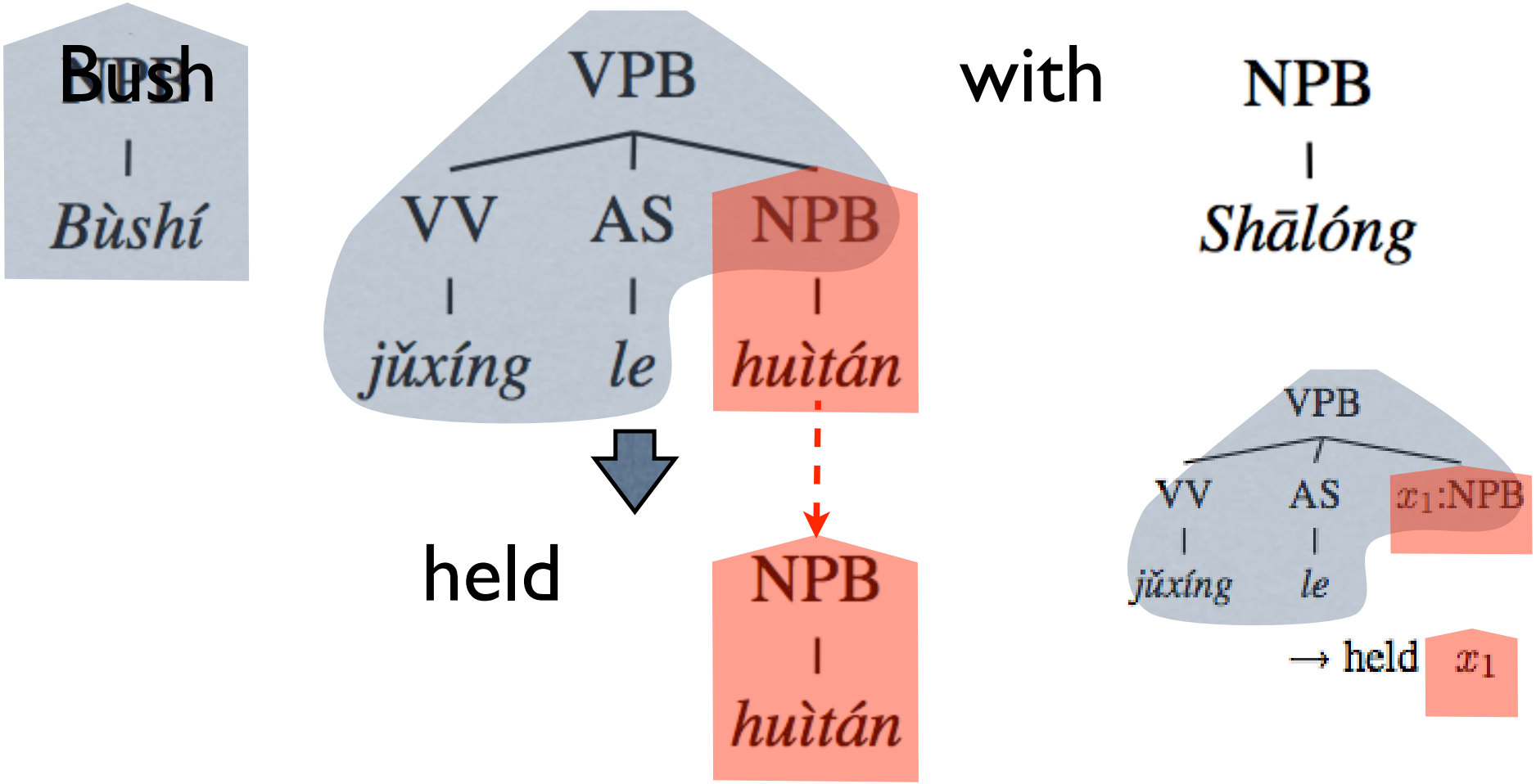
- recursive rewrite by pattern-matching




(Huang, Knight, Joshi 2006)

# Tree-based Translation

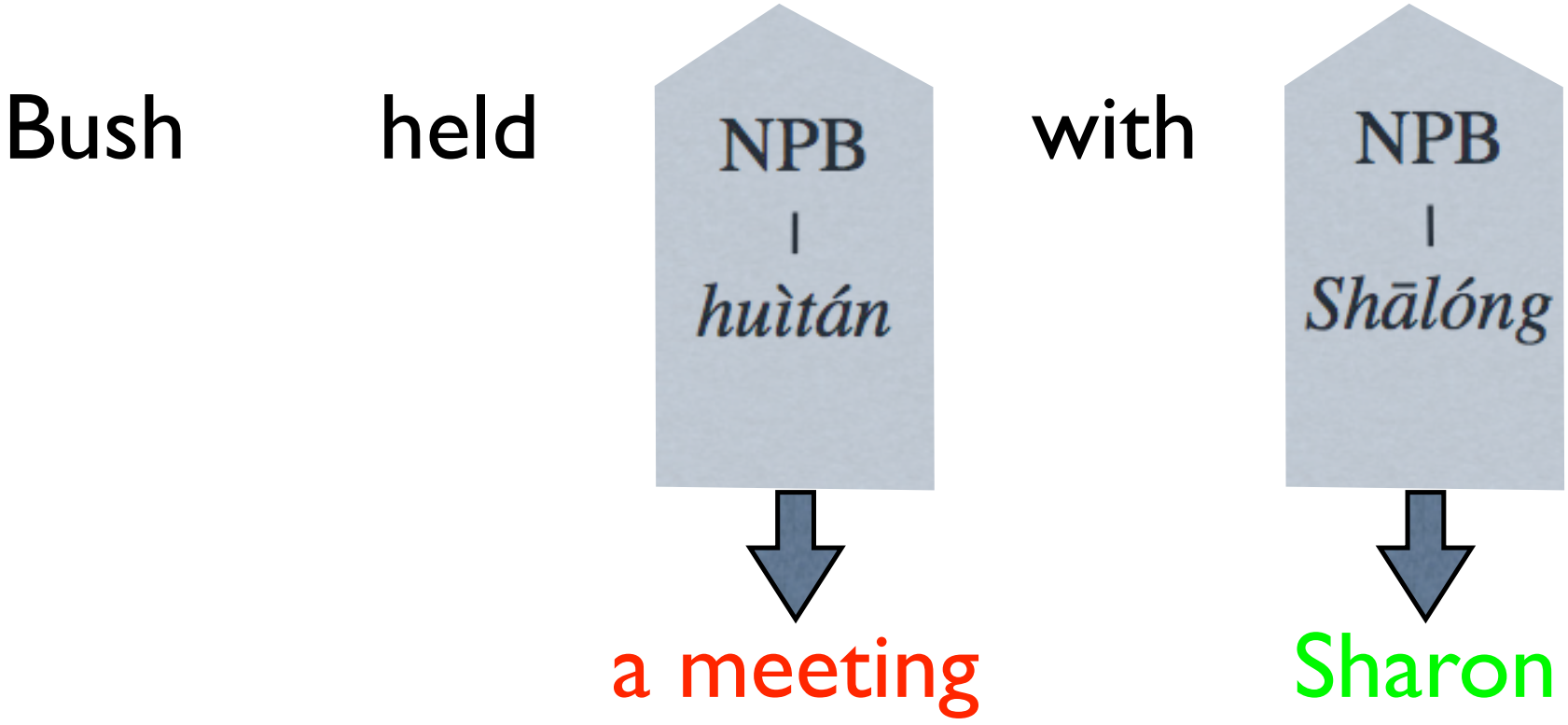
- recursively solve unfinished subproblems




(Huang, Knight, Joshi 2006)

# Tree-based Translation

- continue pattern-matching






(Huang, Knight, Joshi 2006)

# Tree-based Translation

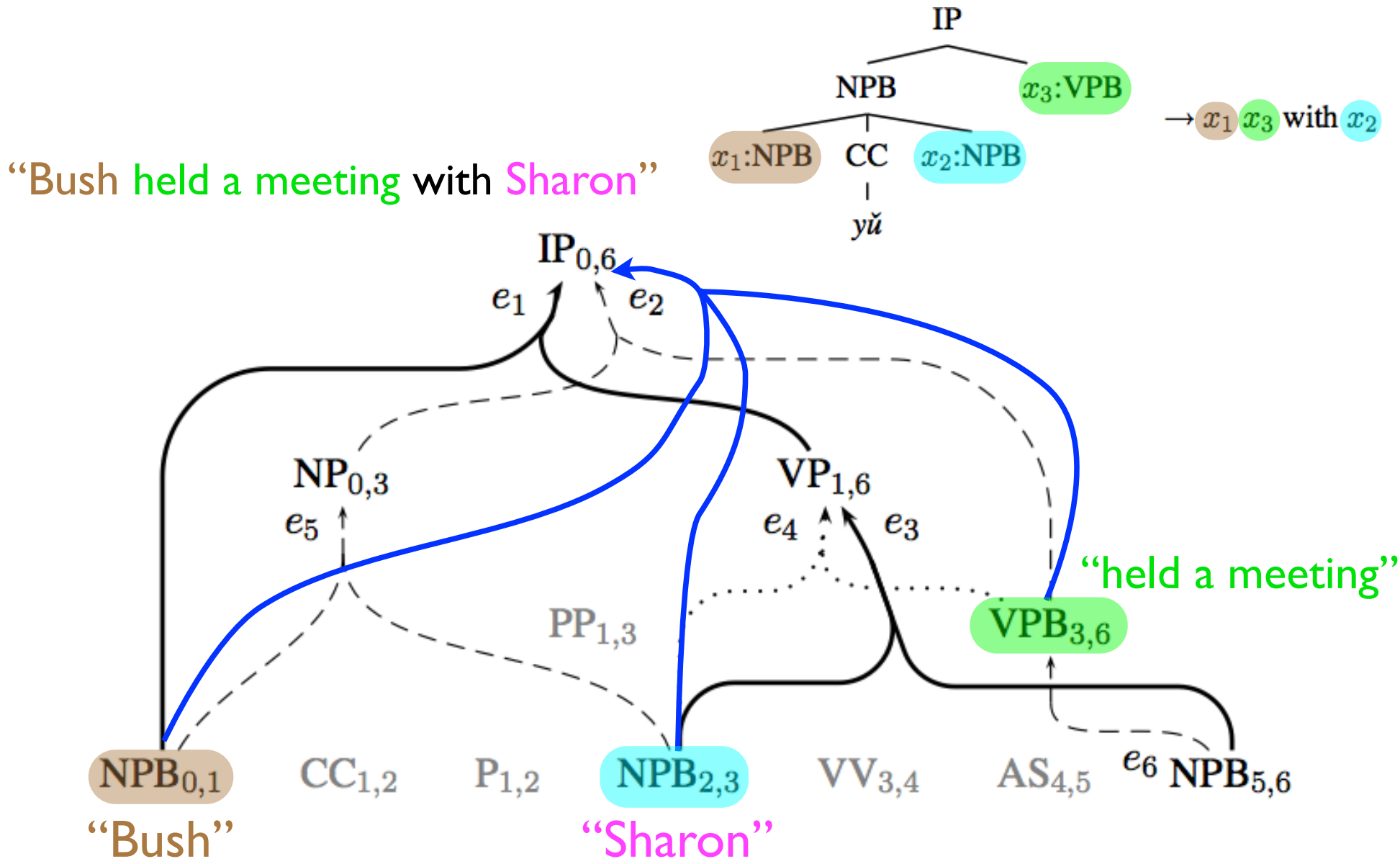
- continue pattern-matching

Bush held a meeting with Sharon

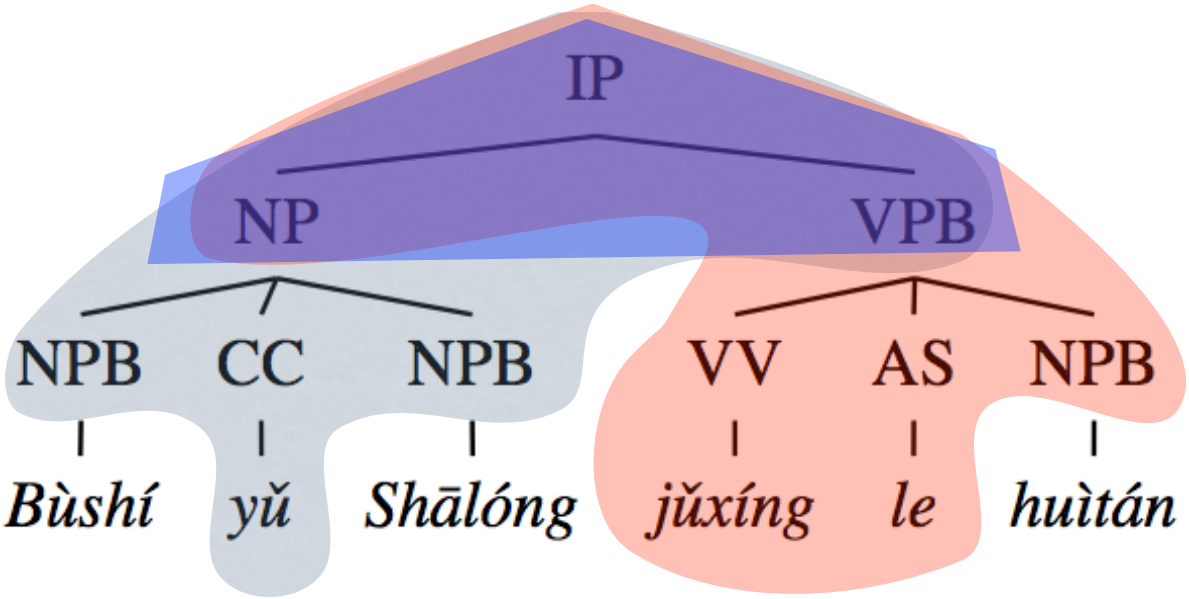


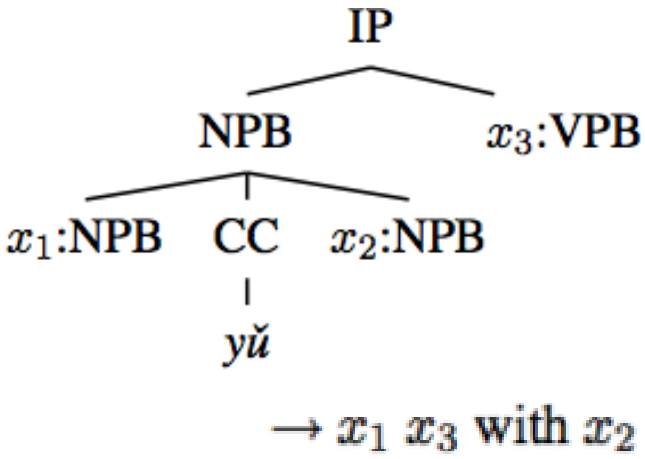
(Galley et al. 2004; Huang, Knight, Joshi 2006)

# Recursion


IP
NPB
$x_3$:VPB
$x_1$:NPB
CC
$x_2$:NPB
yǔ
→ $x_1$ $x_3$ with $x_2$
"Bush held a meeting with Sharon"
$IP_{0,6}$
$e_1$
$e_2$
$NP_{0,3}$
$e_5$
$VP_{1,6}$
$e_4$
$e_3$
"held a meeting"
$PP_{1,3}$
$VPB_{3,6}$
$NPB_{0,1}$
$CC_{1,2}$
$P_{1,2}$
$NPB_{2,3}$
$VV_{3,4}$
$AS_{4,5}$
$e_6$ $NPB_{5,6}$
"Bush"
"Sharon"

# Memoization (dynamic programming)




cache the best translation (and its probability)
at each node (i.e. subtree)

# Pseudocode

**Algorithm 1** Top-down Memoized Recursion

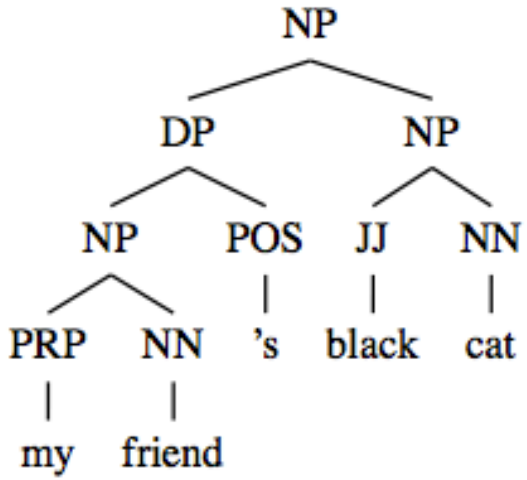
```
1:  function TRANSLATE(η)
2:      if cache[η] defined then                          ▷ this sub-tree visited before?
3:          return cache[η]
4:      best ← 0
5:      for r ∈ R do                                      ▷ try each rule r
6:          matched, sublist ← PATTERNMATCH(t(r), η)      ▷ tree pattern matching
7:          if matched then   ▷ if matched, sublist contains a list of matched subtrees
8:              prob ← Pr(r)                              ▷ the probability of rule r
9:              for η_i ∈ sublist do
10:                 p_i, s_i ← TRANSLATE(η_i)             ▷ recursively solve each sub-problem
11:                 prob ← prob · p_i
12:             if prob > best then
13:                 best ← prob
14:                 str ← [x_i ↦ s_i]s(r)                 ▷ plug in the results
15:     cache[η] ← best, str                              ▷ caching the best solution for future use
16:     return cache[η]                                   ▷ returns the best string with its prob.
```

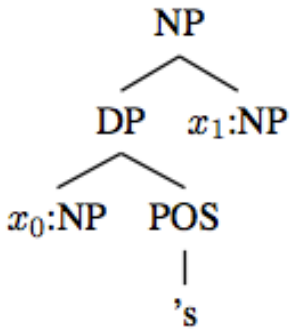
# English-to-Spanish/French Example

(2) my friend's black cat

| | | | | | |
|---|---|---|---|---|---|
| le | chat | noir | de | mon | ami |
| el | gato | negro | de | mi | amigo |
| [the] | [cat] | [black] | [of] | [my] | [friend] |

Again, assume the parse tree for the English phrase is:




$(r'_1)$  $\rightarrow$ $x_1$ de $x_0$



Then we use a rule to translate the English phrase "my friend":

$(r'_2)$ NP( PRP (my) NN (friend) ) $\rightarrow$ mon ami / mi amigo

and a rule for reordering the adjective (black) with the noun (ca

$(r'_3)$ NP( $x_0$:JJ $x_1$:NN ) $\rightarrow$ $x_1$ $x_0$

finally, we finish the translation with the following two lexical r

$(r'_4)$ black $\rightarrow$ noir / negro

$(r'_5)$ cat $\rightarrow$ le chat / el gato

# Sample Input/Ouptut (online)

input1.txt:

```
NP(DP(NP(PRP("my") NN("friend")) POS("'s")) NP(JJ("black") NN("cat")))
NP(DP(NP(PRP("my") NN("friend")) POS("'s")) NP(JJ("white") NN("cat")))
```

rules1.txt:

```
NP(DP(x0:NP POS("'s")) x1:NP) -> x1 "de" x0 ### prob=1.0
NP(PRP("my") NN("friend")) -> "mon" "ami" ### prob=0.51
NP(PRP("my") NN("friend")) -> "mon" "amie" ### prob=0.49
NP(x0:JJ x1:NN) -> x1 x0 ### prob=0.7
NP(x0:JJ x1:NN) -> x0 x1 ### prob=0.3
JJ("black") -> "noir" ### prob=0.6
JJ("black") -> "noire" ### prob=0.4
NN("cat") -> "le" "chat" ### prob=1.0
NP(x0:JJ NN("cat")) -> "le" "chat" x0 ### prob=1
NP(JJ("black") x0:NN) -> x0 "noir" ### prob=0.55
NP(JJ("black") x0:NN) -> x0 "noire" ### prob=0.45
```

To run the program, type in the terminal

```
cat input1.txt | ./translate.py rules1.txt
```

and you will get the output

```
my friend 's black cat -> le chat noir de mon ami ### prob=0.306
my friend 's white cat -> *** failed ***
```

# Derivation and k-best translation

```
cat input1.txt | ./translate.py rules1.txt -d
```

should output the following:

```
my friend 's black cat -> le chat noir de mon ami ### prob=0.306
 NP (DP (x0:NP POS ('s)) x1:NP) -> x1 de x0 ### prob=1.000
  | x0: NP (PRP (my) NN (friend)) -> mon ami ### prob=0.510
  | x1: NP (x0:JJ NN (cat)) -> le chat x0 ### prob=1.000
  | | x0: JJ (black) -> noir ### prob=0.600
  | black cat -> le chat noir ### prob=0.600
 my friend 's black cat -> le chat noir de mon ami ### prob=0.306
my friend 's white cat -> *** failed ***
```

```
cat input1.txt | ./translate.py rules1.txt -k 3
```

will output

```
my friend 's black cat -> le chat noir de mon ami ### prob=0.306
my friend 's black cat -> le chat noir de mon amie ### prob=0.294
my friend 's black cat -> le chat noire de mon ami ### prob=0.204
my friend 's white cat -> *** failed ***
```

Note: do not print duplicate translations (say, there might be another way of deriving the best translate