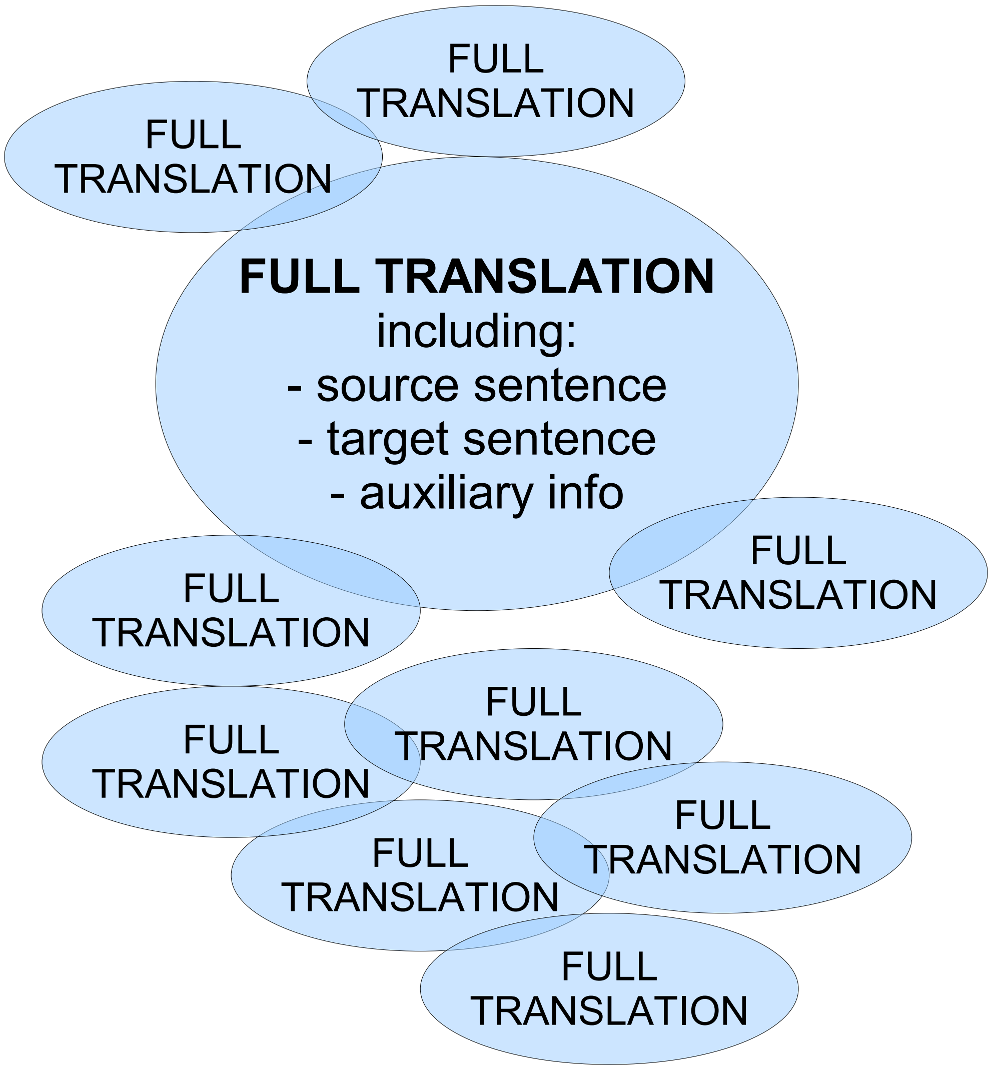
Machine Translation as Tree Labeling

Mark Hopkins and Jonas Kuhn
Department of Linguistics
University of Potsdam
hopkins@ling.uni-potsdam.de
kuhn@ling.uni-potsdam.de

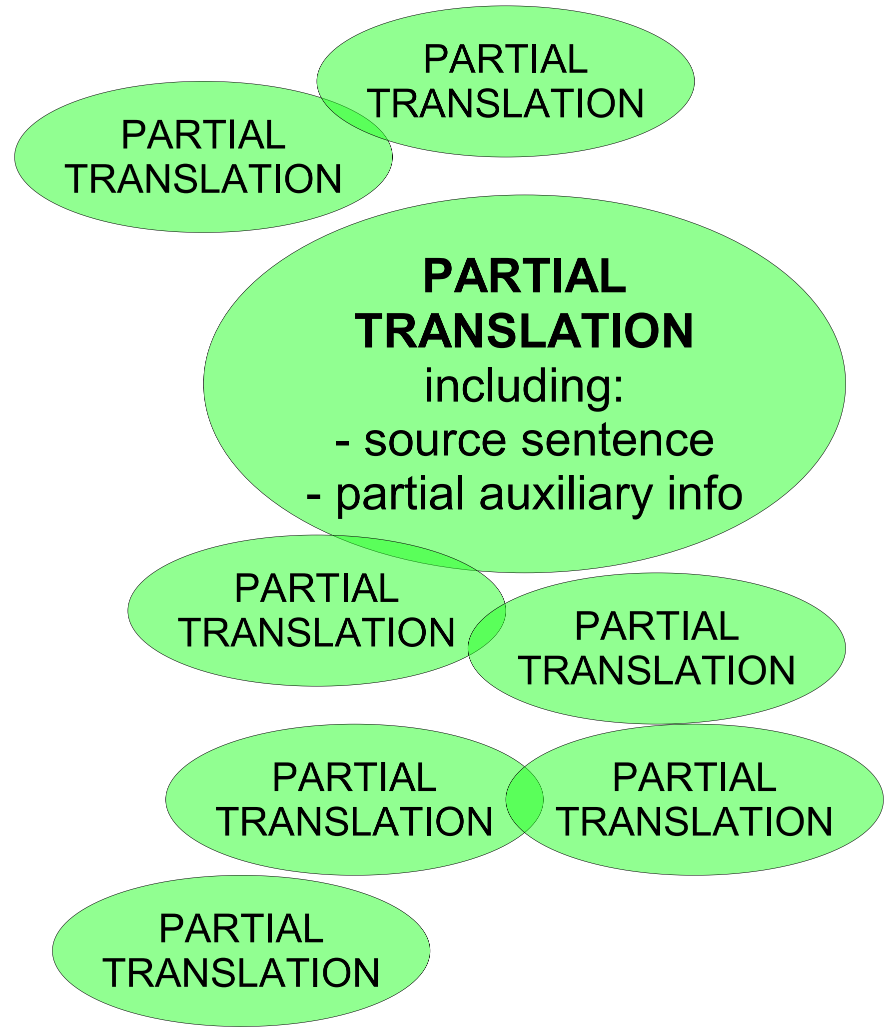
The familiar goal: use syntax in a meaningful, central way to genuinely help machine translation. Our hypothesis is that by using syntactic information intelligently enough, it should be possible to learn the basics of translation with only a small quantity of training data.

The general set-up:

TRAINING DATA

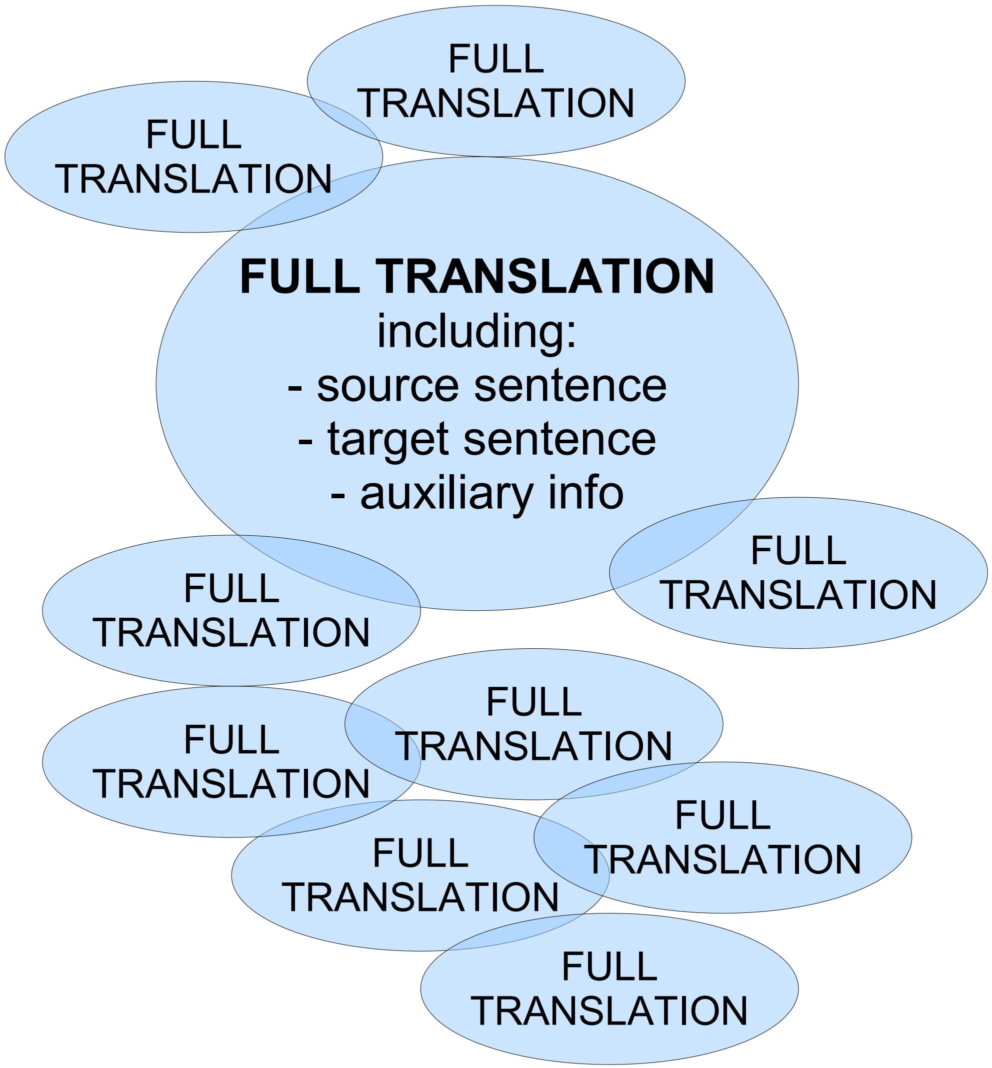


EVALUATION DATA

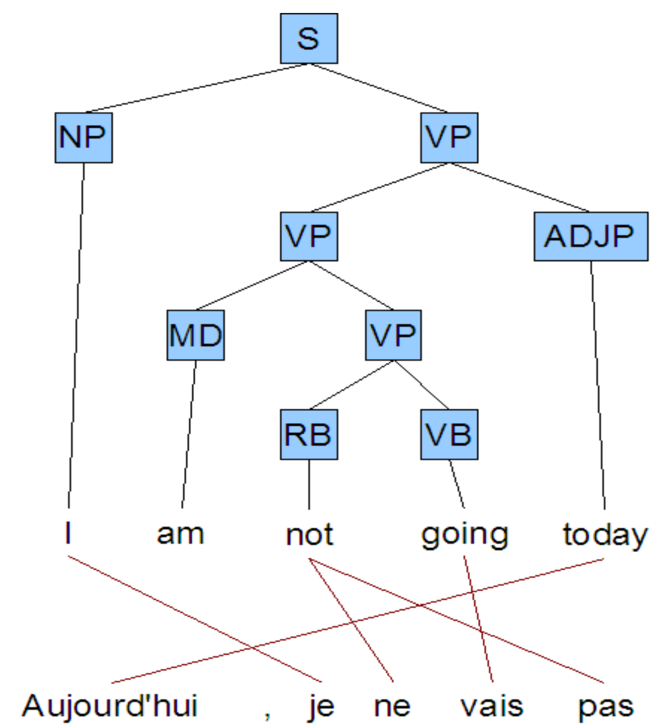


The general set-up:

TRAINING DATA

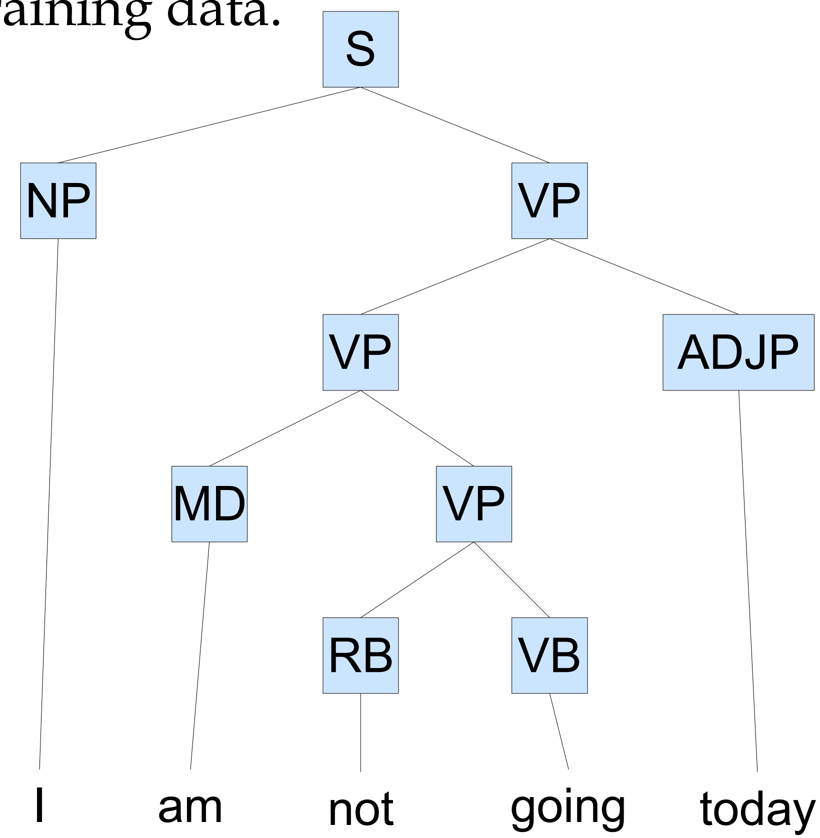


The first decision is to determine what these "full translations" are going to look like. Namely, what kind of auxiliary info are we going to rely on? Our assumption is that they will look as follows:

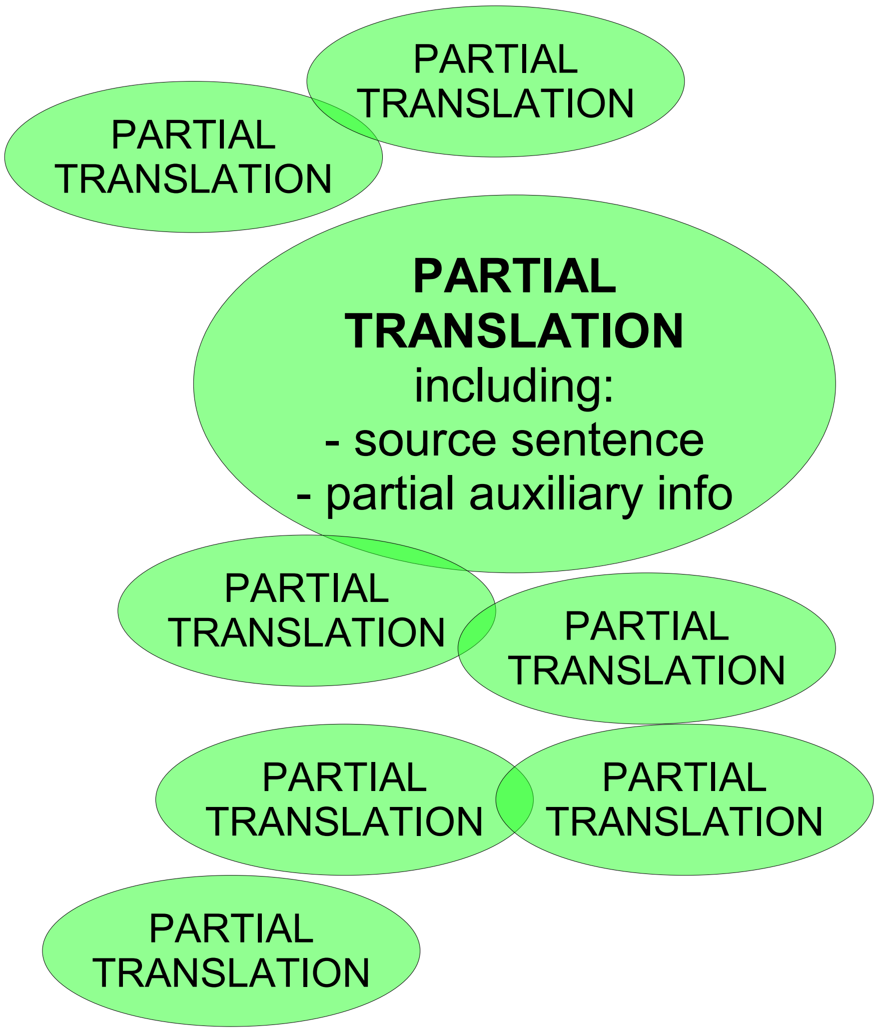


The general set-up:

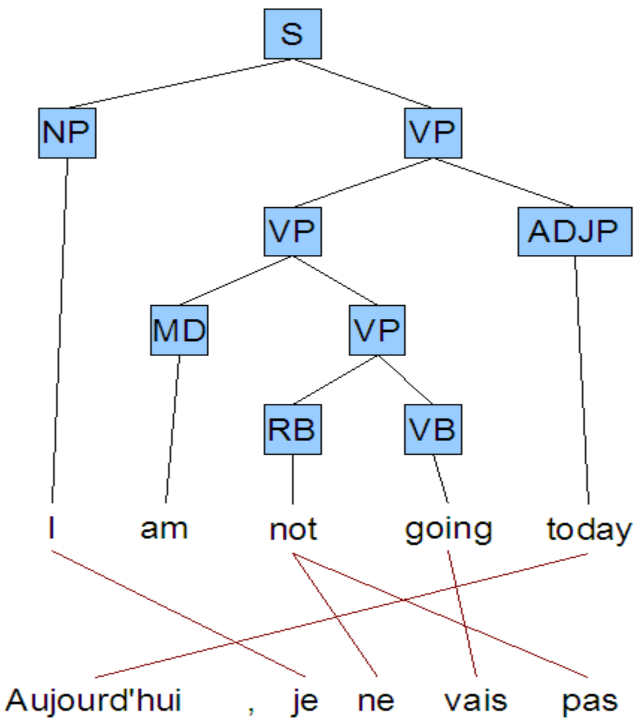
Next decision: what will the "partial translations" look like? We will assume that they are source sentences which are parsed and annotated in the same manner as the training data.



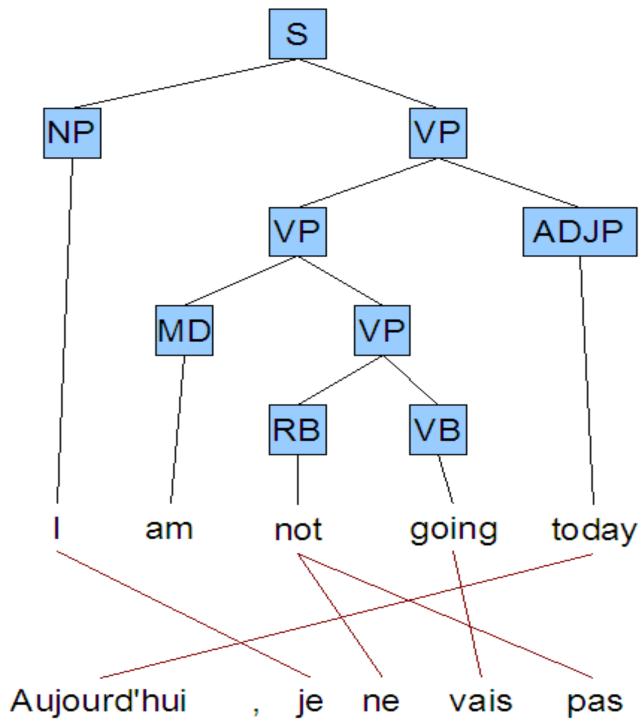
EVALUATION DATA



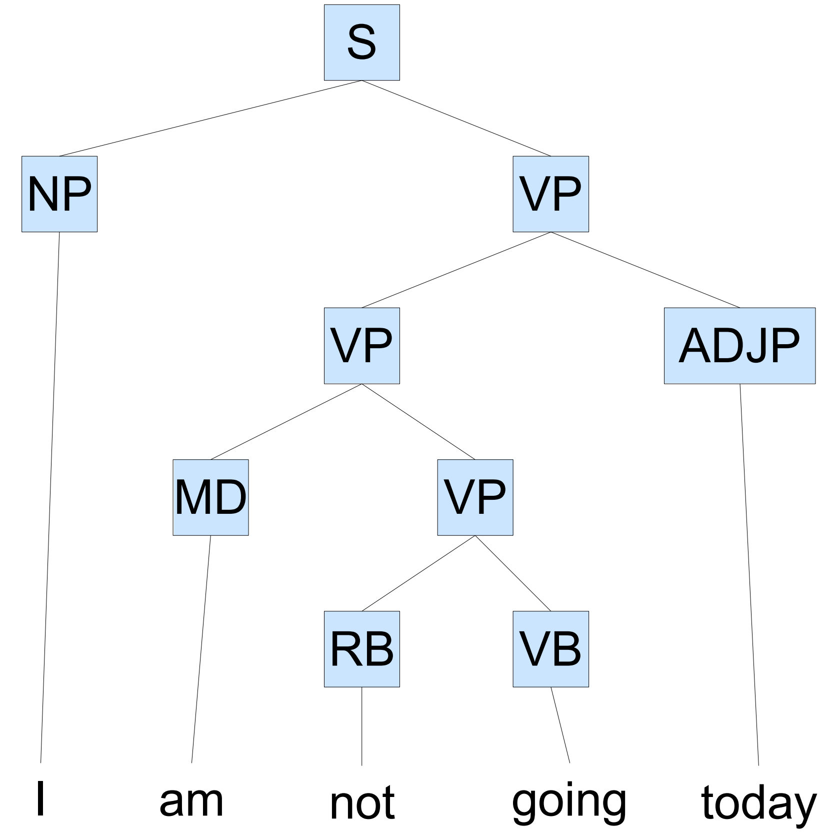
To summarize... from a training set of full translations:



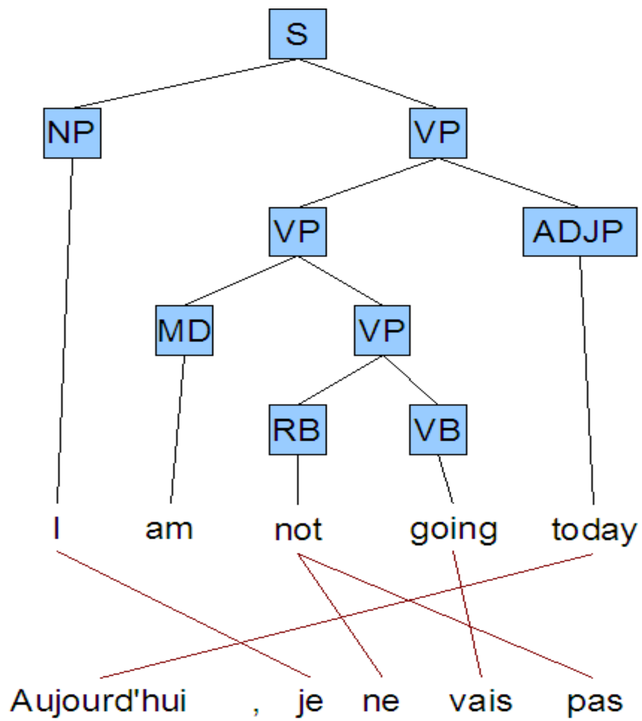
To summarize... from a training set of full translations:



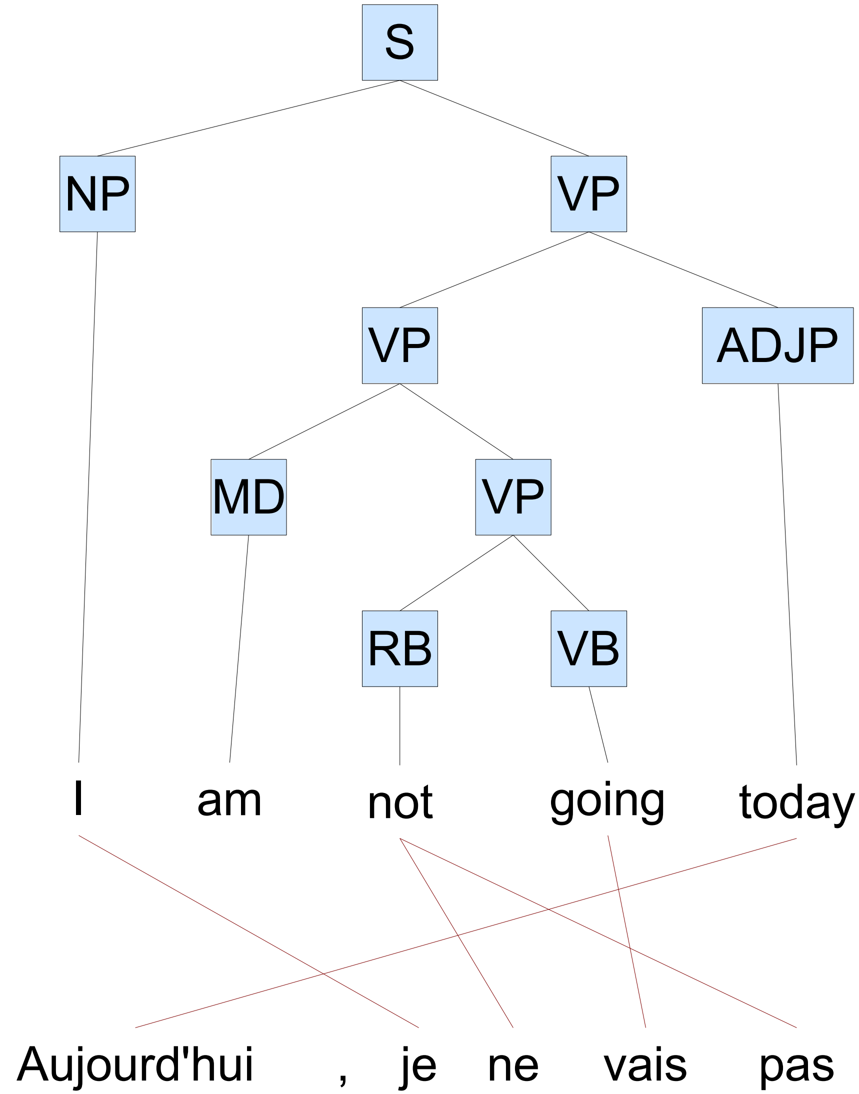
...we want to learn how to go from a partial translation...



To summarize... from a training set of full translations:



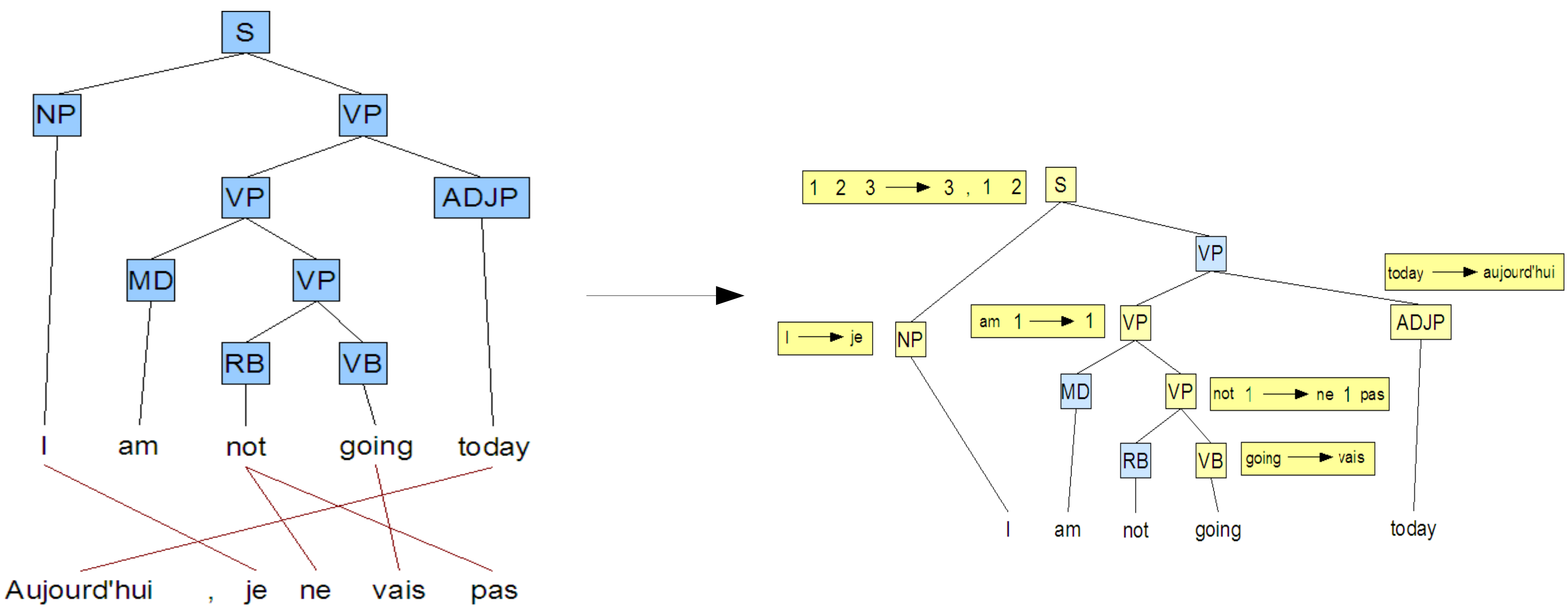
...we want to learn how to go from a partial translation...



...to a full translation.

We will show:

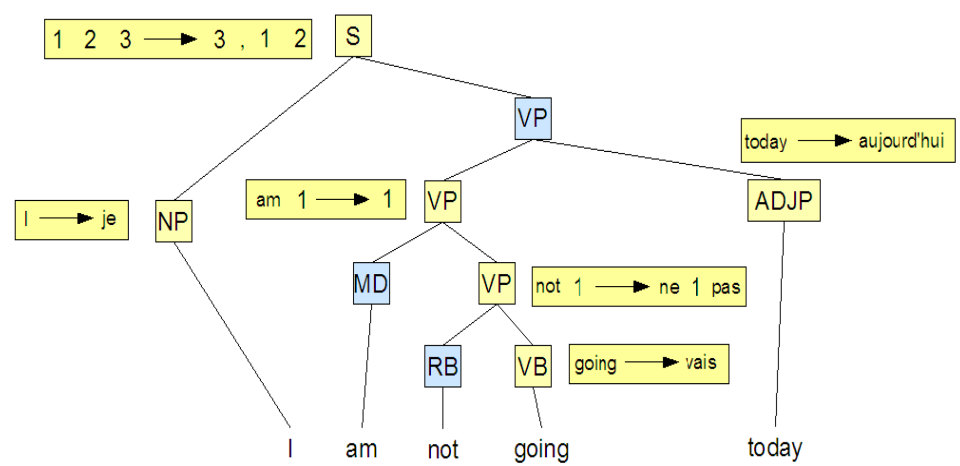
(1) how to take an aligned tree-string pair in our training corpus, and convert it to an equivalent labeled tree representation.



We will show:

(1) how to take an aligned tree-string pair in our training corpus, and convert it to an equivalent labeled tree representation.
(2) how to take this labeled tree representation, and convert it to a series of (mostly binary) decisions – a generative story

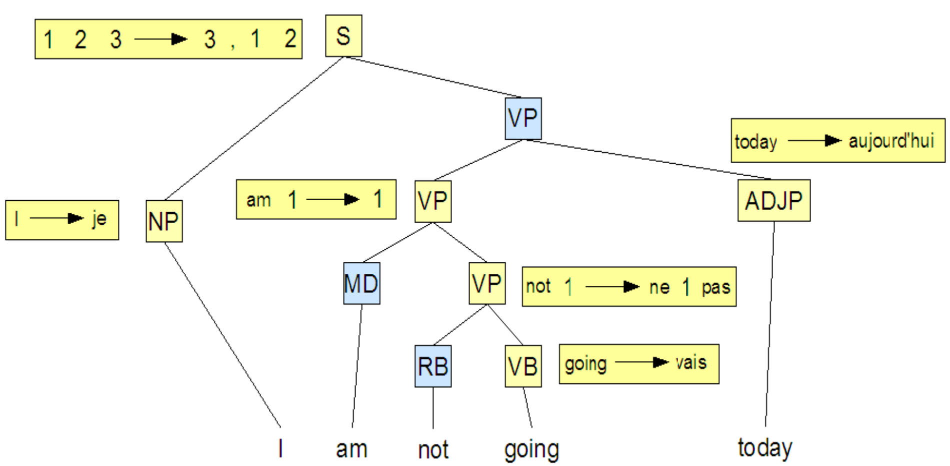
RULE (root): **YES**
RULE (NP): **YES**
RULE (VP1): **NO**
RULE (VP2): **YES**
...
RHS TEMPLATE(root): **X , X**
VAR 1 RIGHT(root)? **YES**
VAR 2 LEFT(root)? **NO**
VAR 3 LEFT(root)? **YES**
VAR 3 LEFT(root)? **YES**
VAR 3 LEFT(root)? **YES**
RHS TEMPLATE(NP): **je**
RHS TEMPLATE(VP2): **X**
RHS TEMPLATE(VP3): **ne X pas**
RHS TEMPLATE(VB): **vais**
RHS TEMP(ADJP): **aujourd'hui**



After converting each item in our training corpus to such a generative story:

- training will consist of learning how to make each of these decisions in the generative story (discriminatively)
- decoding will consist of computing the lowest cost generative story according to our learned distributions.

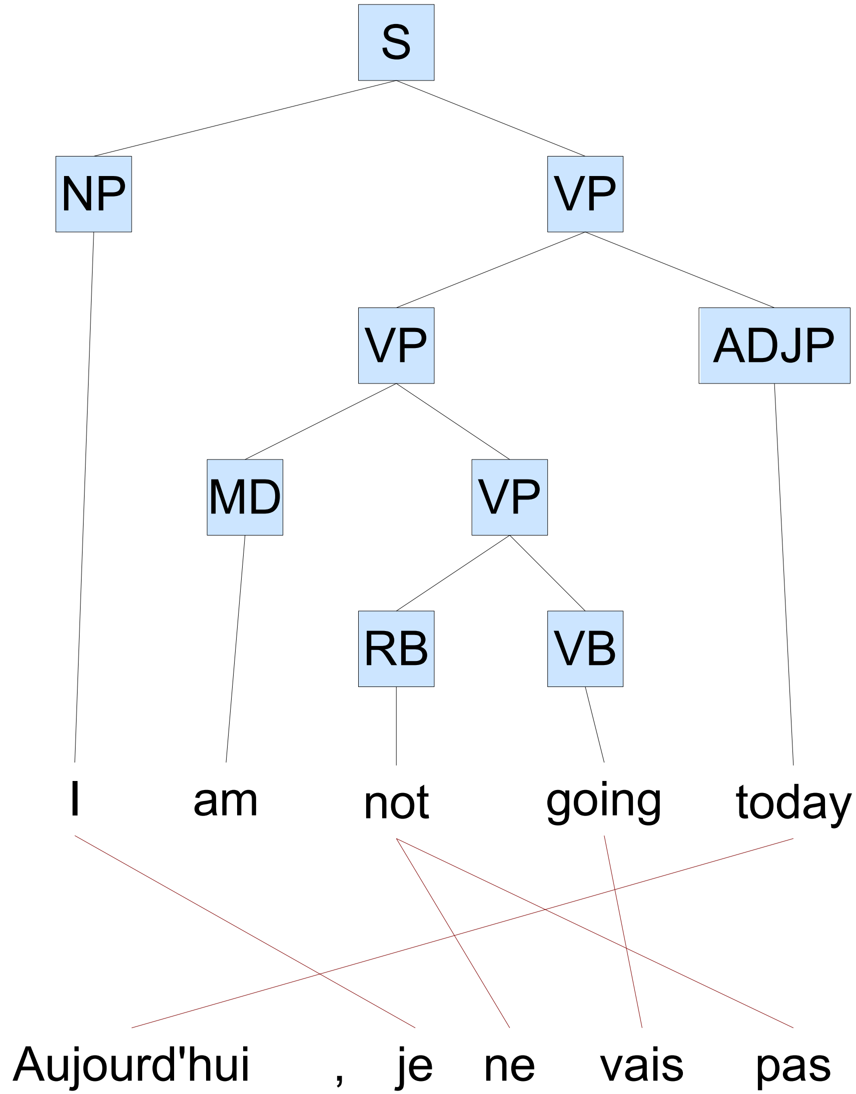
RULE (root): **YES**
RULE (NP): **YES**
RULE (VP1): **NO**
RULE (VP2): **YES**
...
RHS TEMPLATE(root): **X , X**
VAR 1 RIGHT(root)? **YES**
VAR 2 LEFT(root)? **NO**
VAR 3 LEFT(root)? **YES**
VAR 3 LEFT(root)? **YES**
VAR 3 LEFT(root)? **YES**
RHS TEMPLATE(NP): **je**
RHS TEMPLATE(VP2): **X**
RHS TEMPLATE(VP3): **ne X pas**
RHS TEMPLATE(VB): **vais**
RHS TEMP(ADJP): **aujourd'hui**



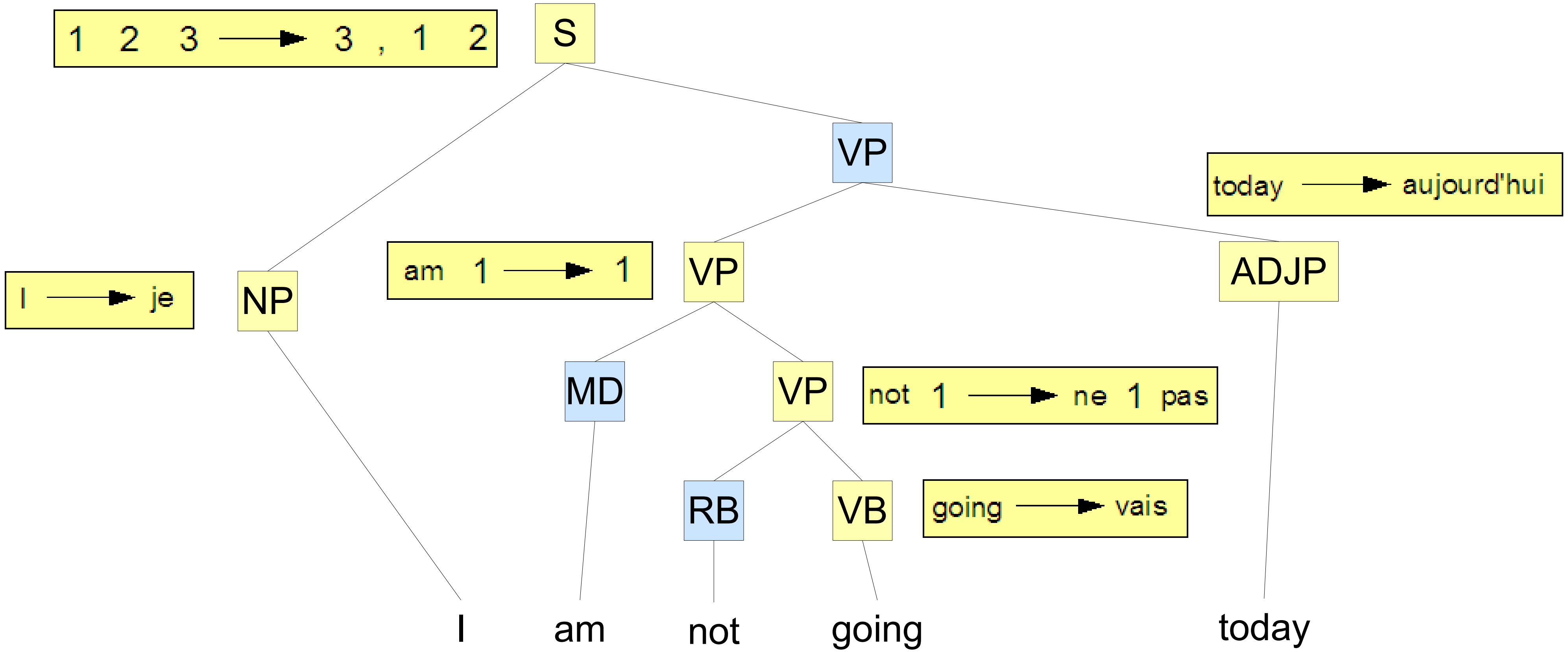
GHKM RULES

In (Galley et al., 2003), the authors propose a way of interpreting any aligned tree-string pair (t,s,a) as the tree t, labeled with a particular type of rule (the so-called GHKM rule).

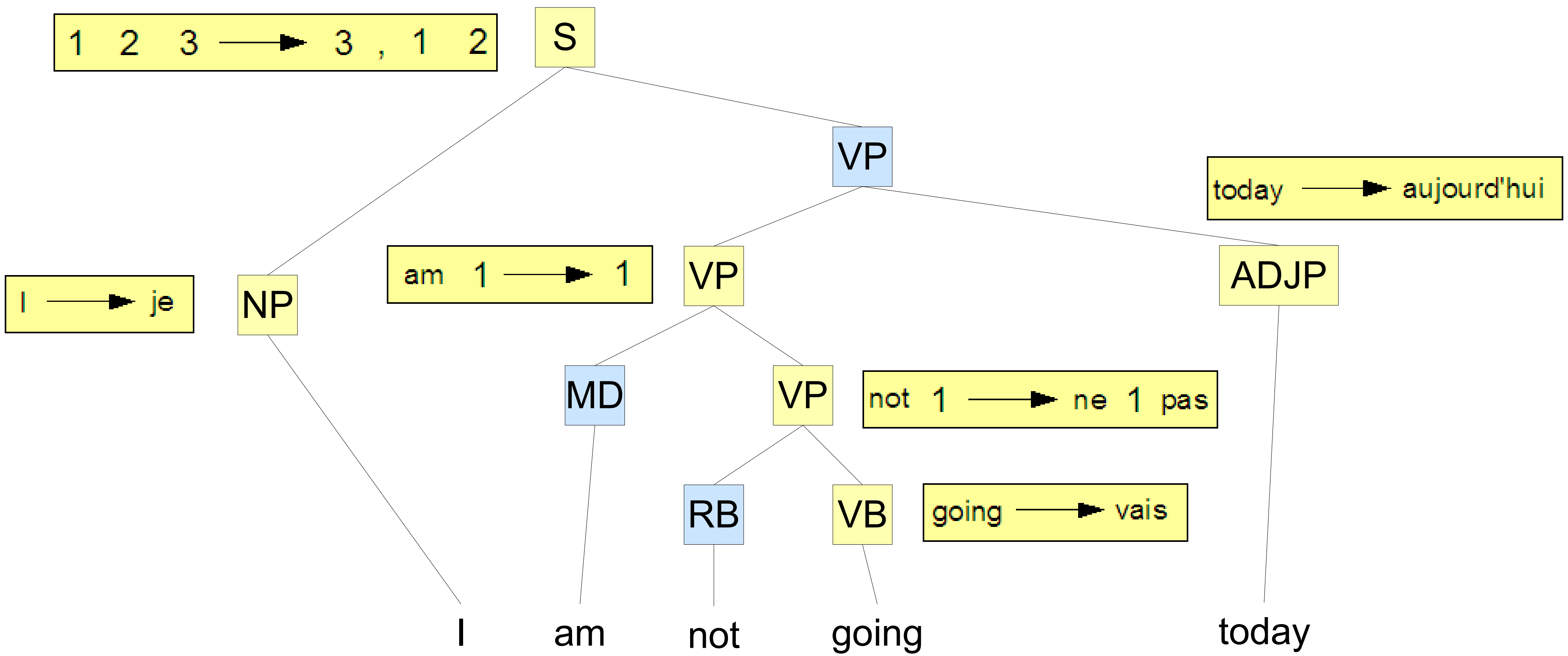
For instance...



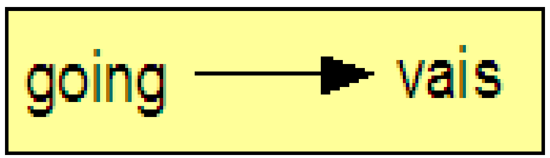
This aligned tree-sentence pair turns into...

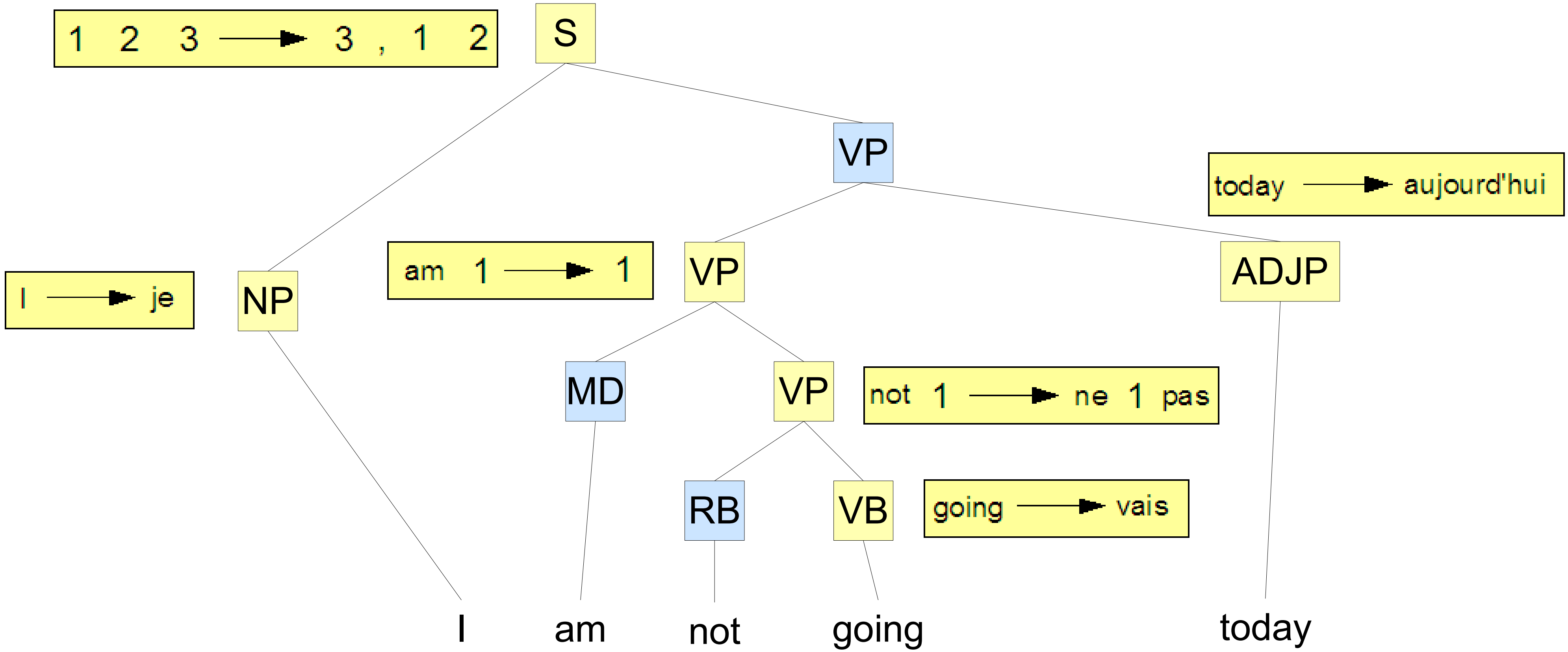


... this labeled tree.

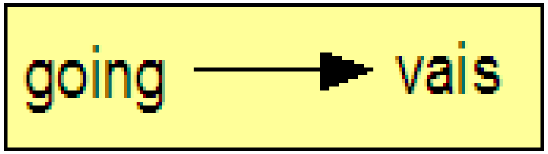


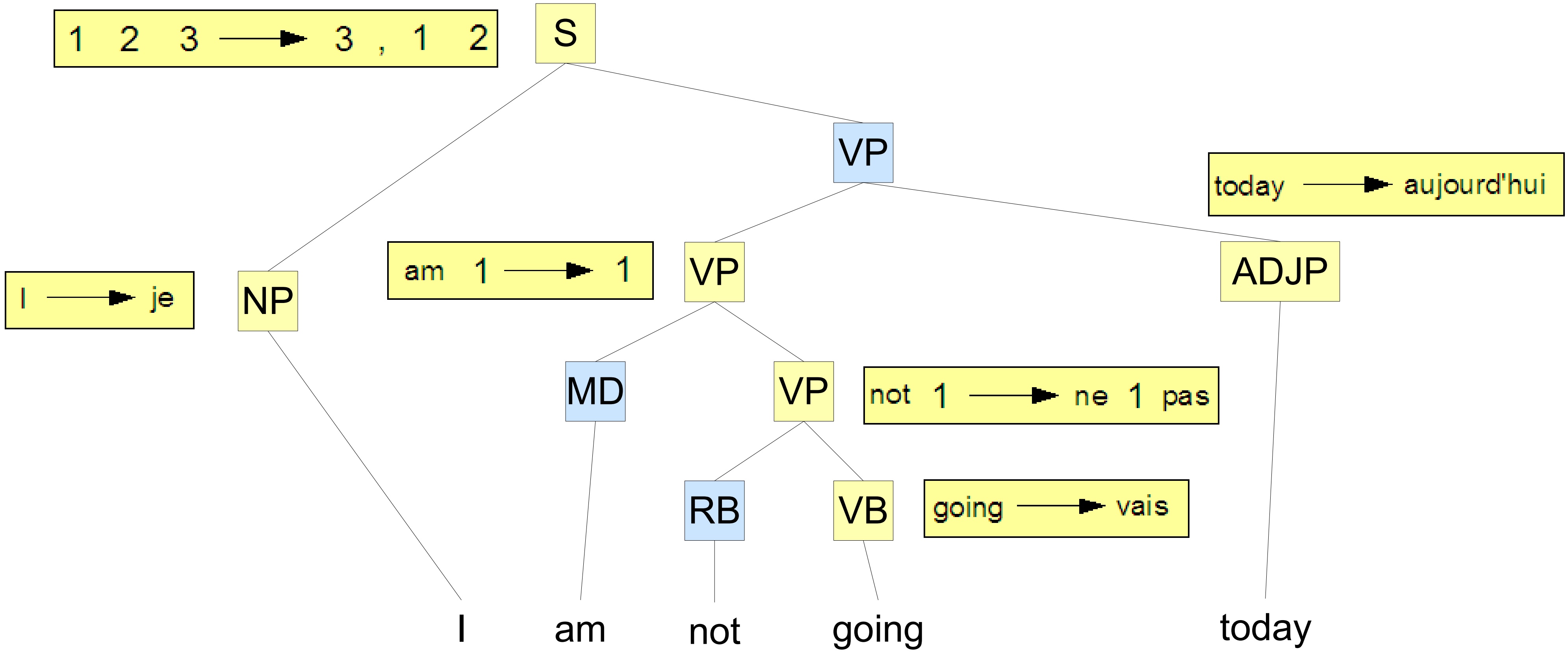
Roughly speaking, what do these rules mean? Consider one of the simplest rules, the one labeling the VB node:





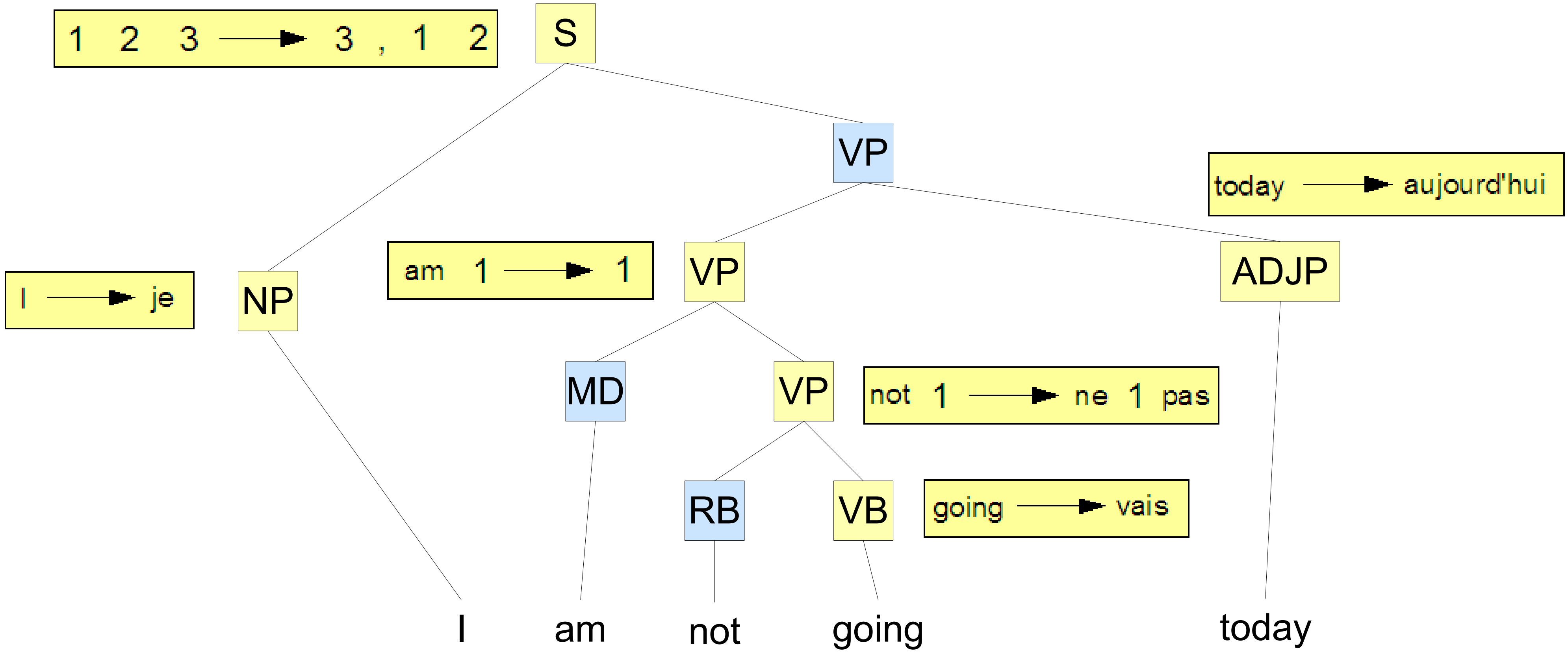
This simply directs you to do the following: if you see the word **"going"** then translate it as **"vais"**.



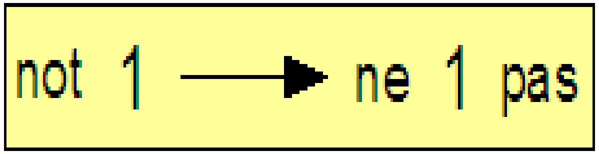


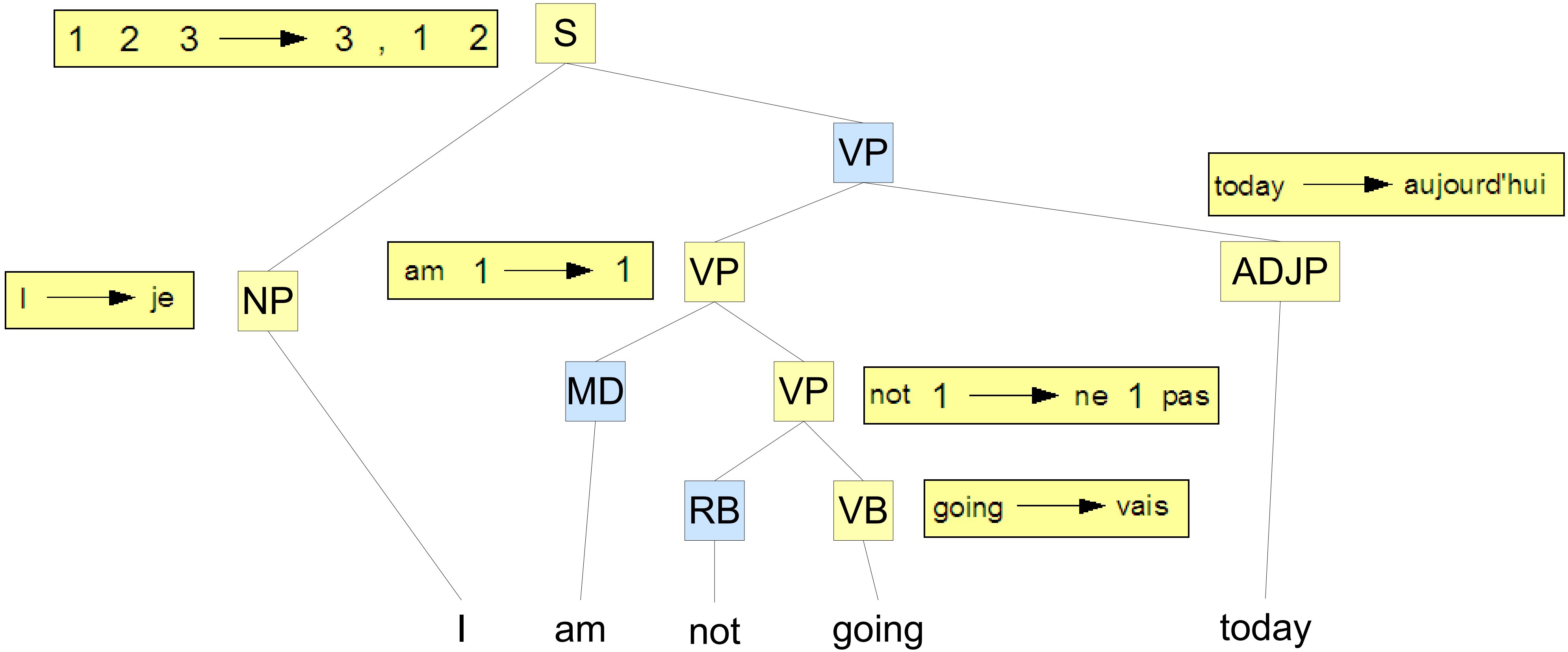
Let's go one level up in the tree and consider a slightly more complicated rule:

not 1 ⟶ ne 1 pas



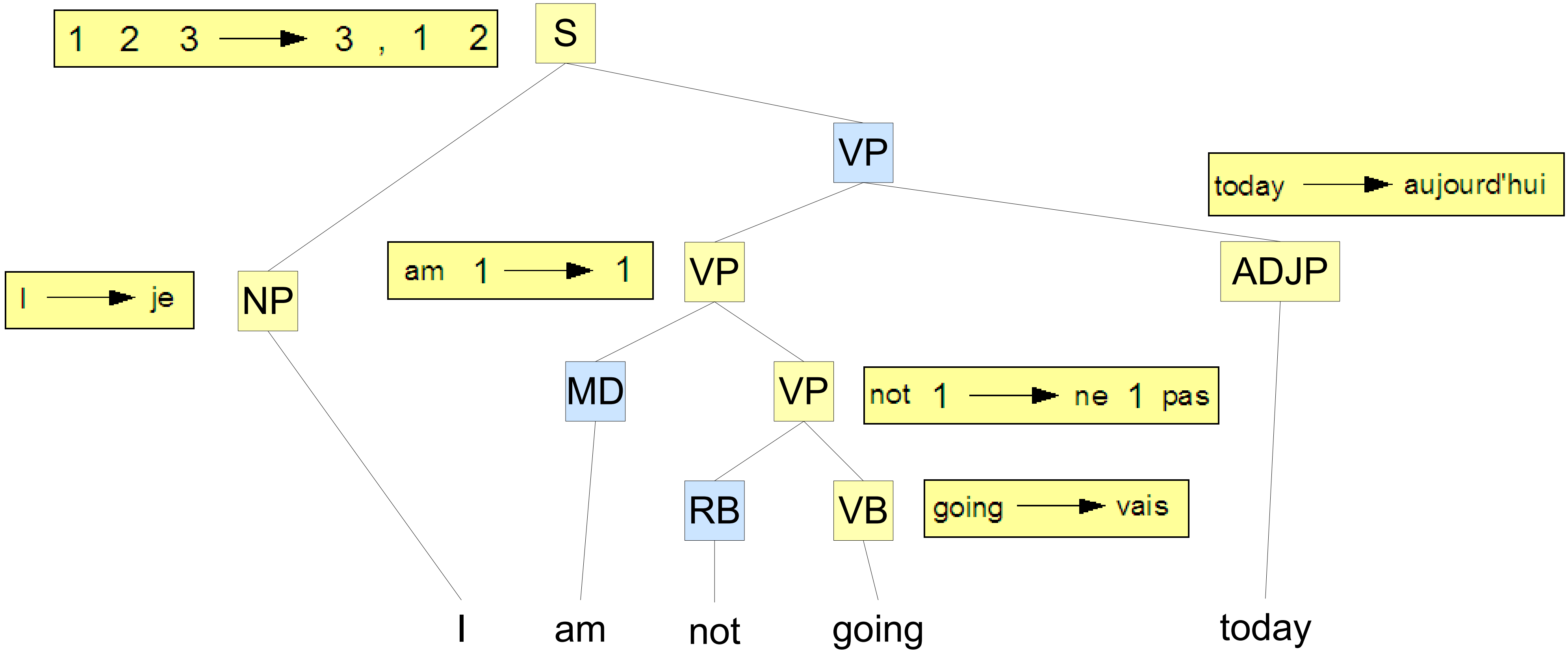
This states:
IF you see the word **“not”**, followed by something you've already translated
THEN translate the whole thing as **“ne”** + the existing translation + **“pas”**





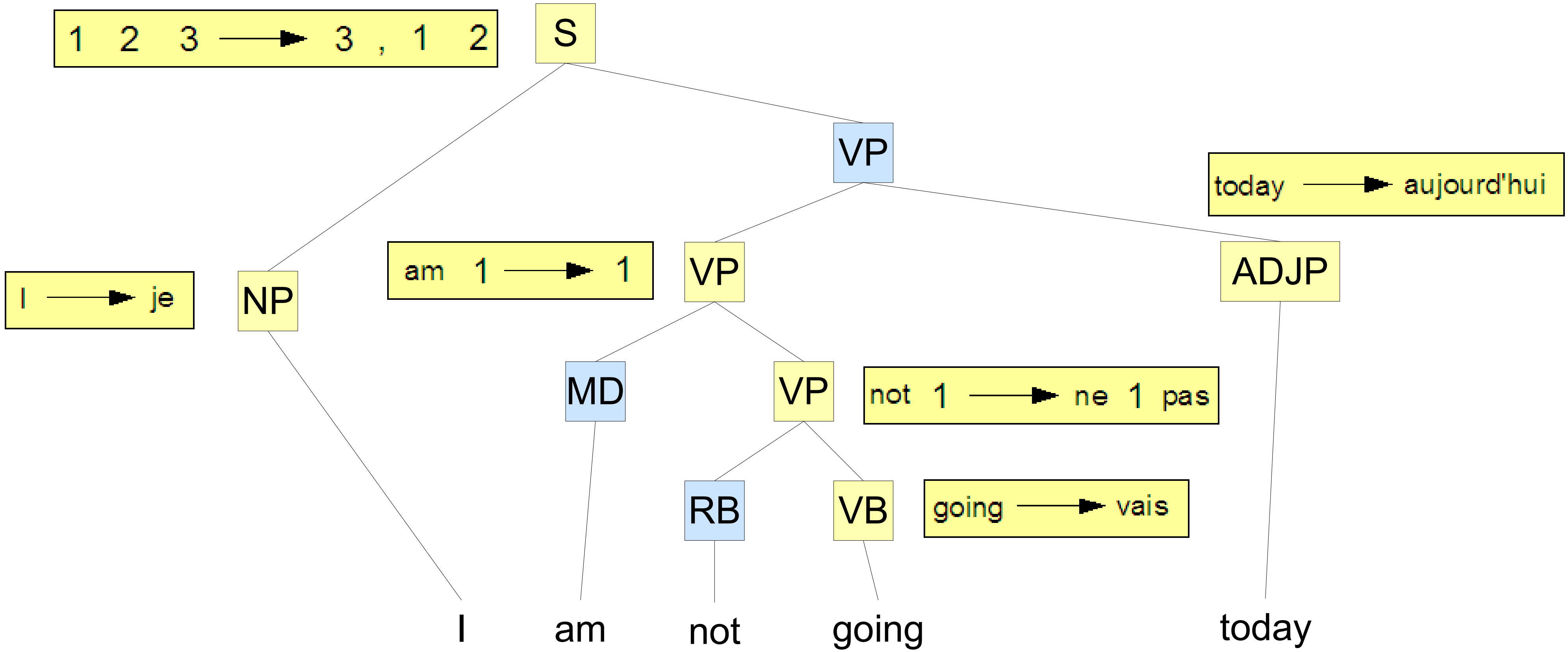
In this particular tree, we've already translated **“going”** as **“vais”**, so we're applying this rule to: **“not vais”**, which therefore becomes **“ne vais pas”**.

not 1 → ne 1 pas



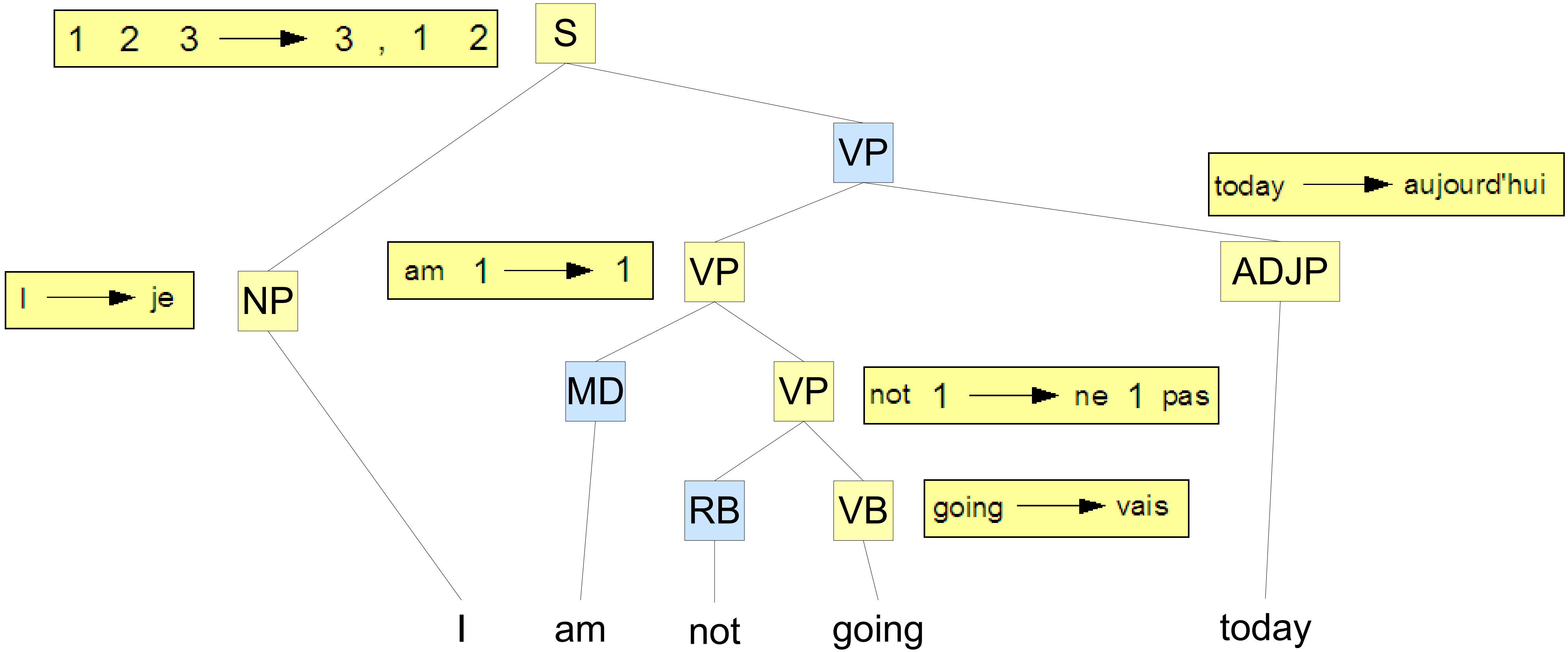
By the time we get to the top of the tree, we've translated the three immediate "rule node" descendants (those descendants annotated with rules) as **"je"**, **"ne vais pas"**, and **"aujourd'hui"**, respectively.

1 2 3 → 3 , 1 2

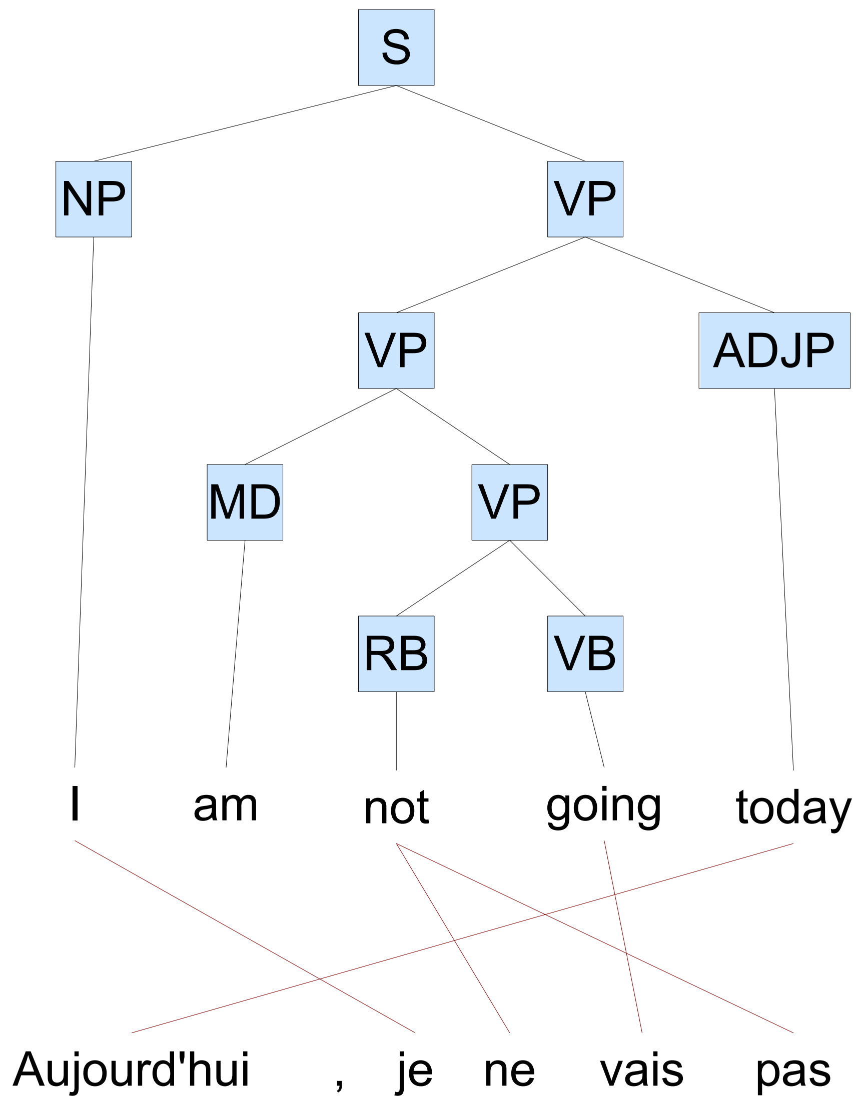


Our translations so far are: **"je"**, **"ne vais pas"**, **"aujourd'hui"** (in that order).
Thus the top rule directs us to reorder these, such that we get as output:
aujourd'hui , je ne vais pas (notice the comma that the rule inserts)

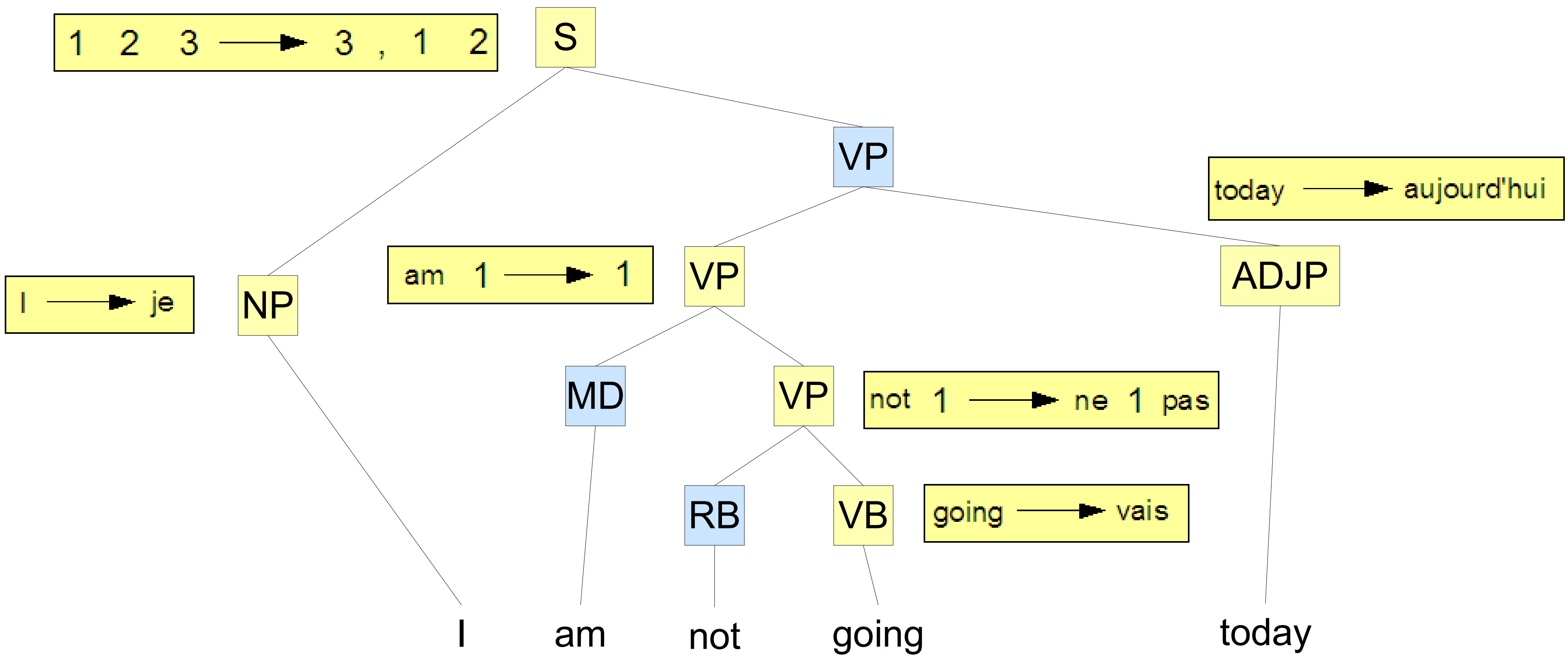
1 2 3 → 3 , 1 2



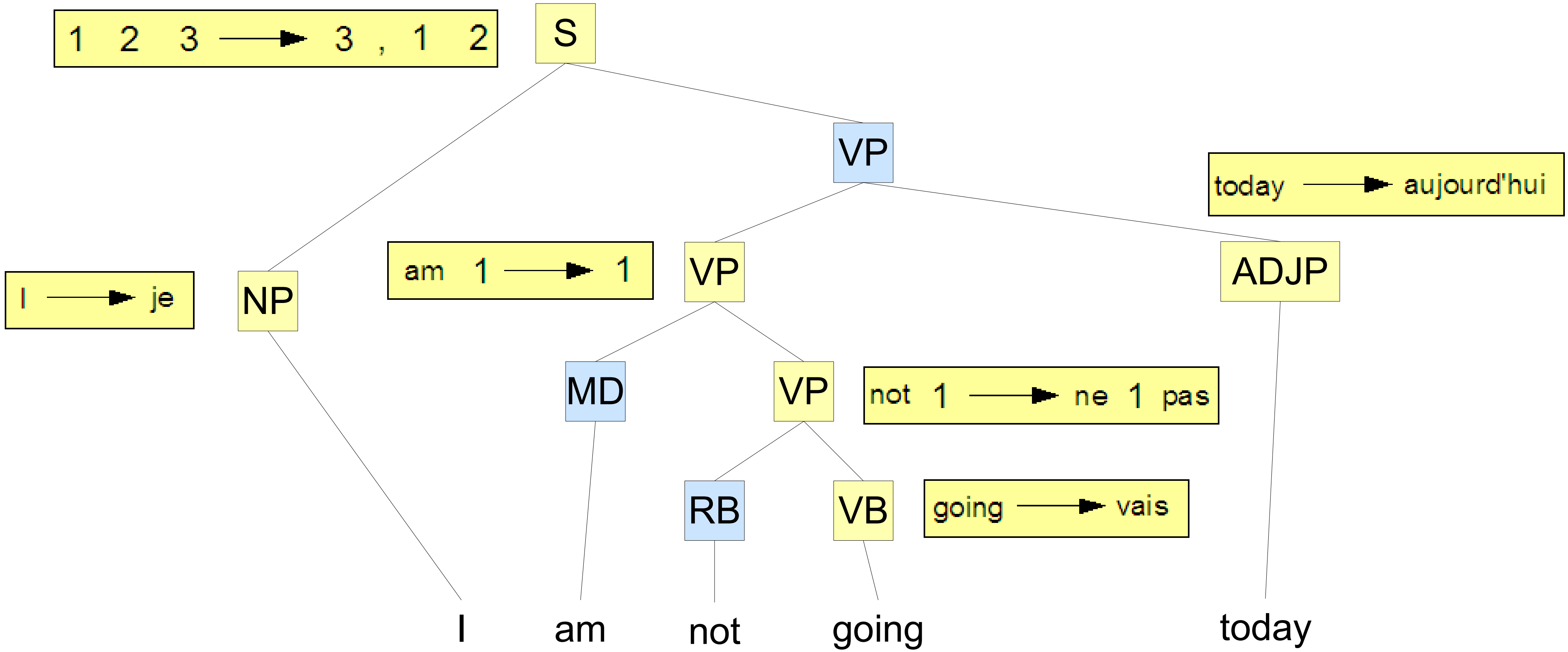
In short, the "GHKM tree" representation implicitly captures the target language translation of this source sentence, which is explicitly represented in the original aligned tree-string pair.



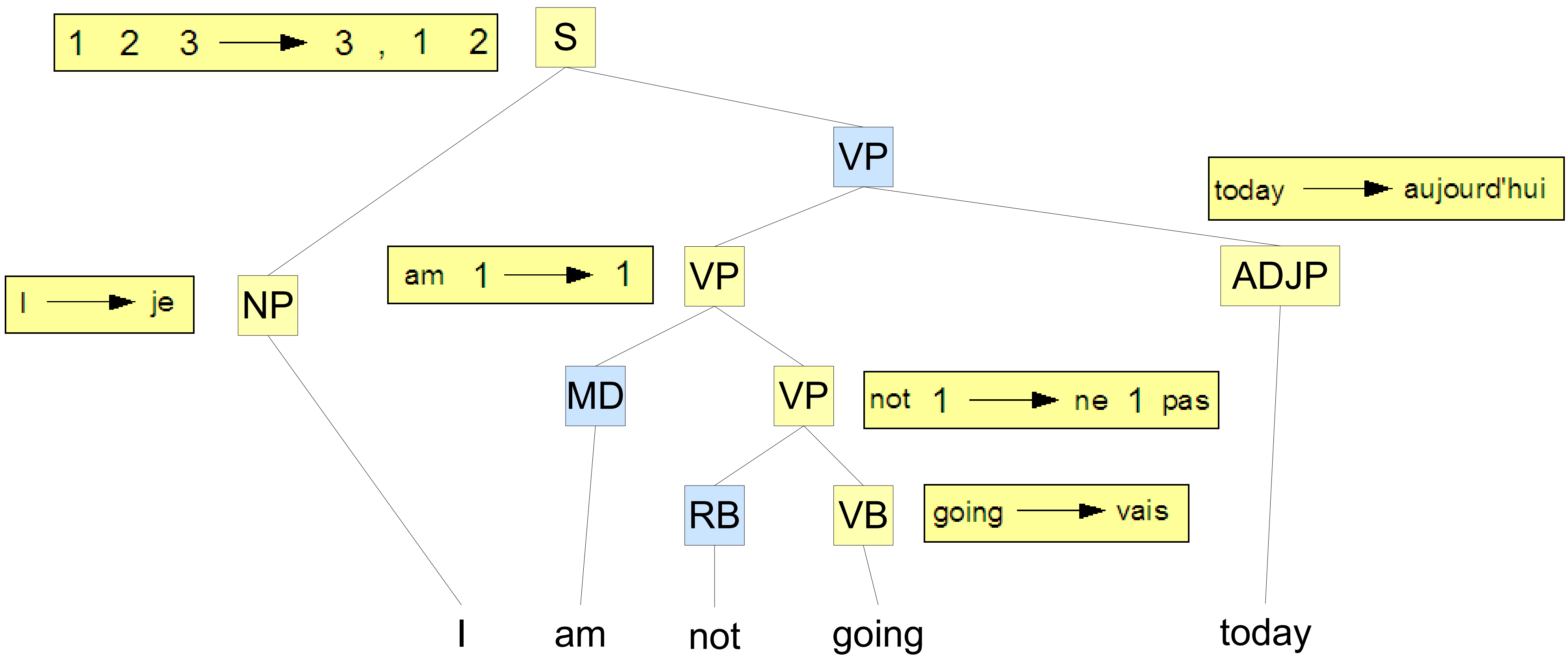
Although we won't go into detail about how (Galley et al., 2003) convert this...


1 2 3 → 3 , 1 2
S
VP
today → aujourd'hui
I → je
NP
am 1 → 1
VP
ADJP
MD
VP
not 1 → ne 1 pas
RB
VB
going → vais
I
am
not
going
today

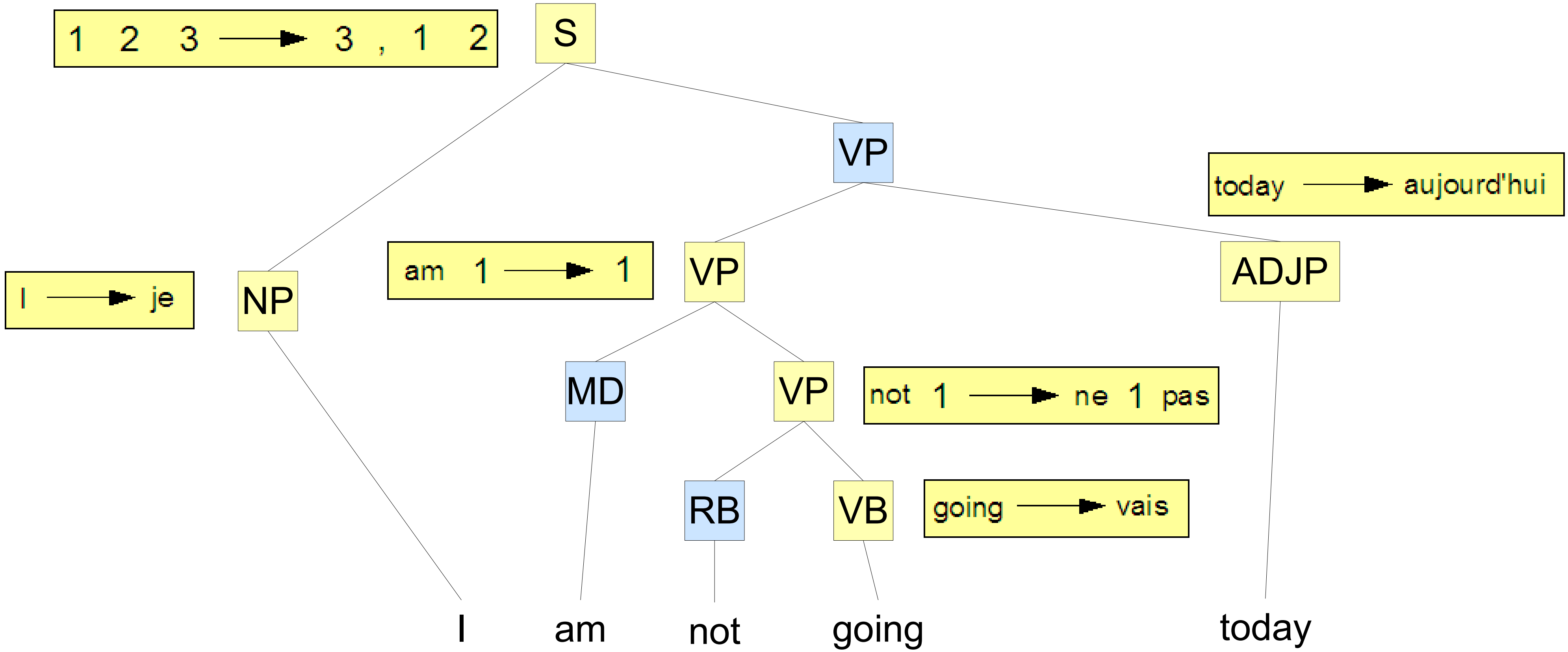
... to this...



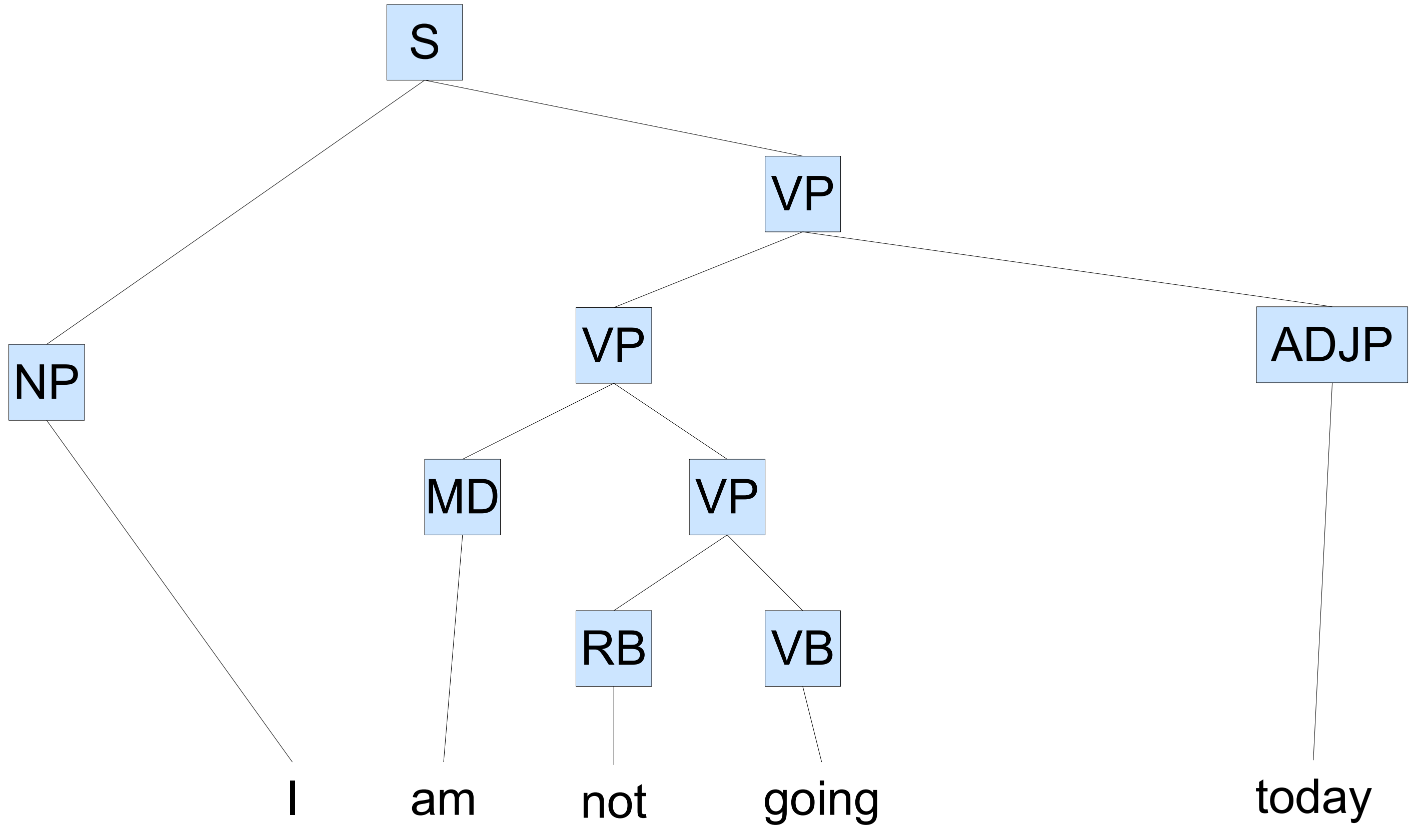
...suffice it to say that this can be done deterministically and efficiently, in such a way that the rules of the GHKM tree "respect" the original alignment (where this idea of "respect" is defined formally by (Galley et al., 2003)).



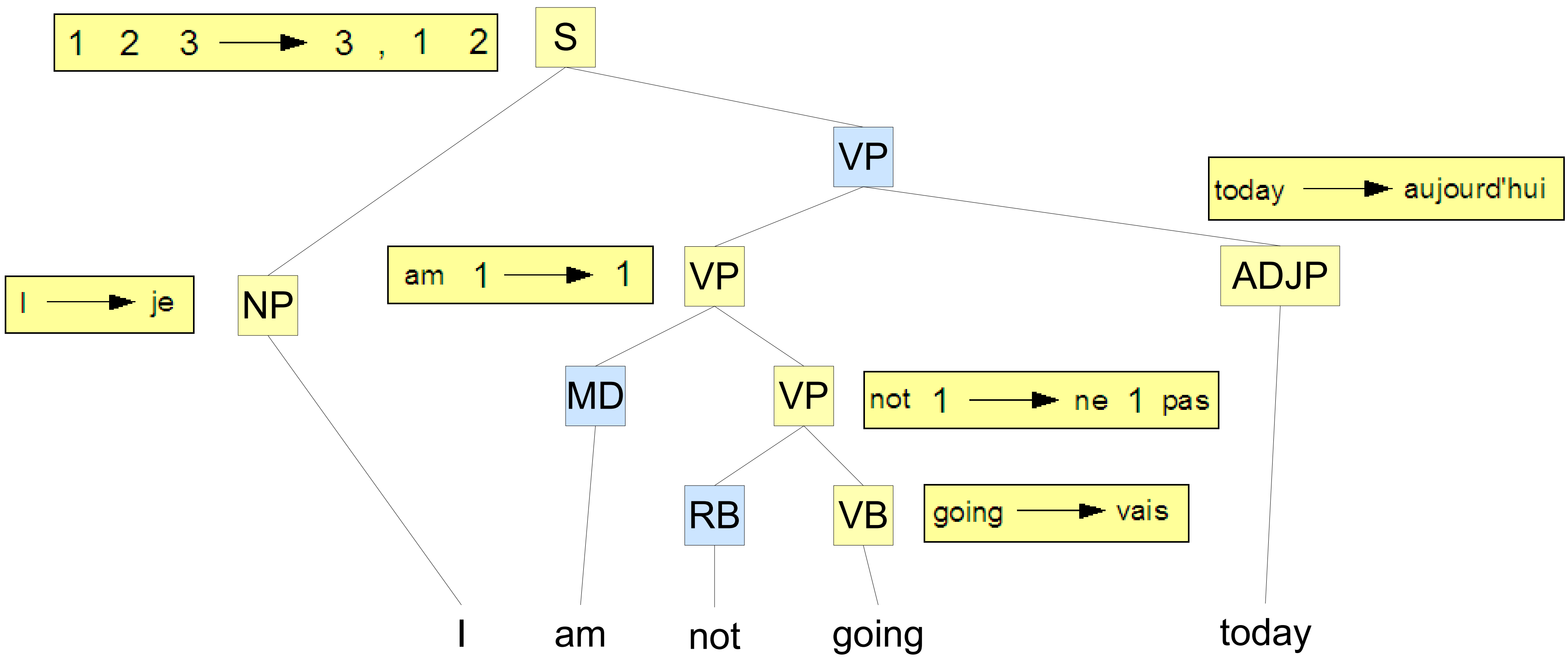
Hence, as a preprocessing step, we can convert every aligned tree-sentence pair in our training corpus to its equivalent GHKM tree representation, and learn from these instead.



Given this representation, what does it mean to translate? It means: we start with a parsed sentence...

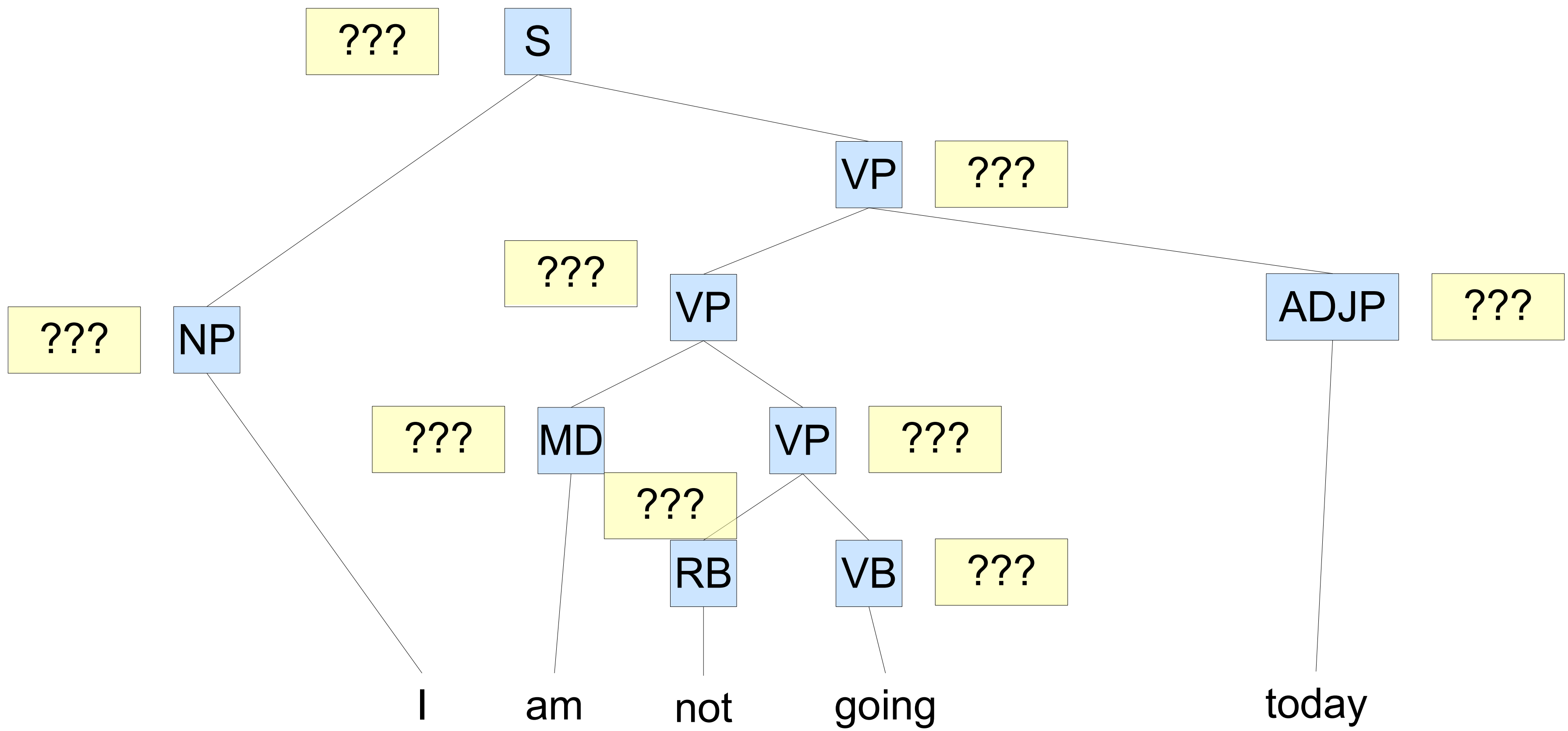


Given this representation, what does it mean to translate? It means: we start with a parsed sentence...

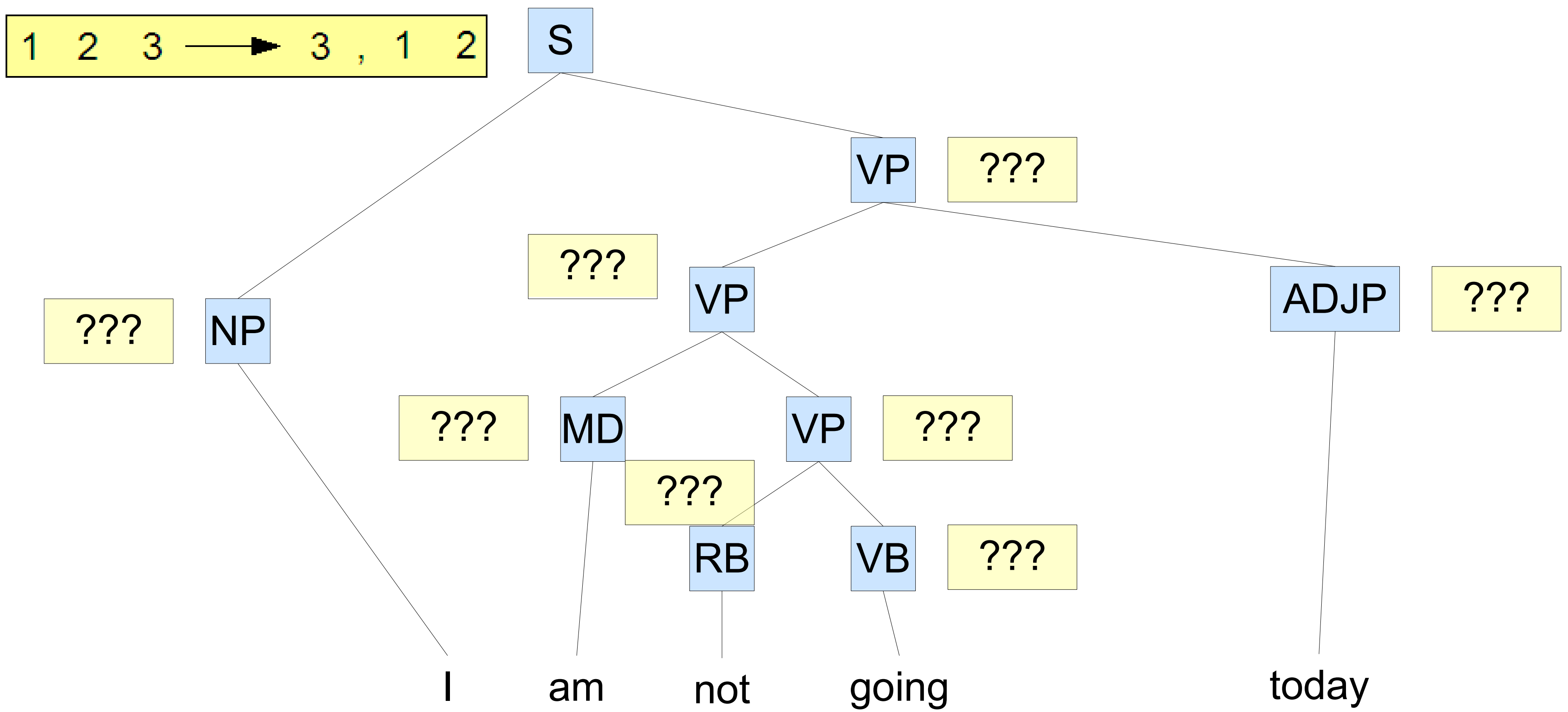


Given this representation, what does it mean to translate? It means: we start with a parsed sentence...

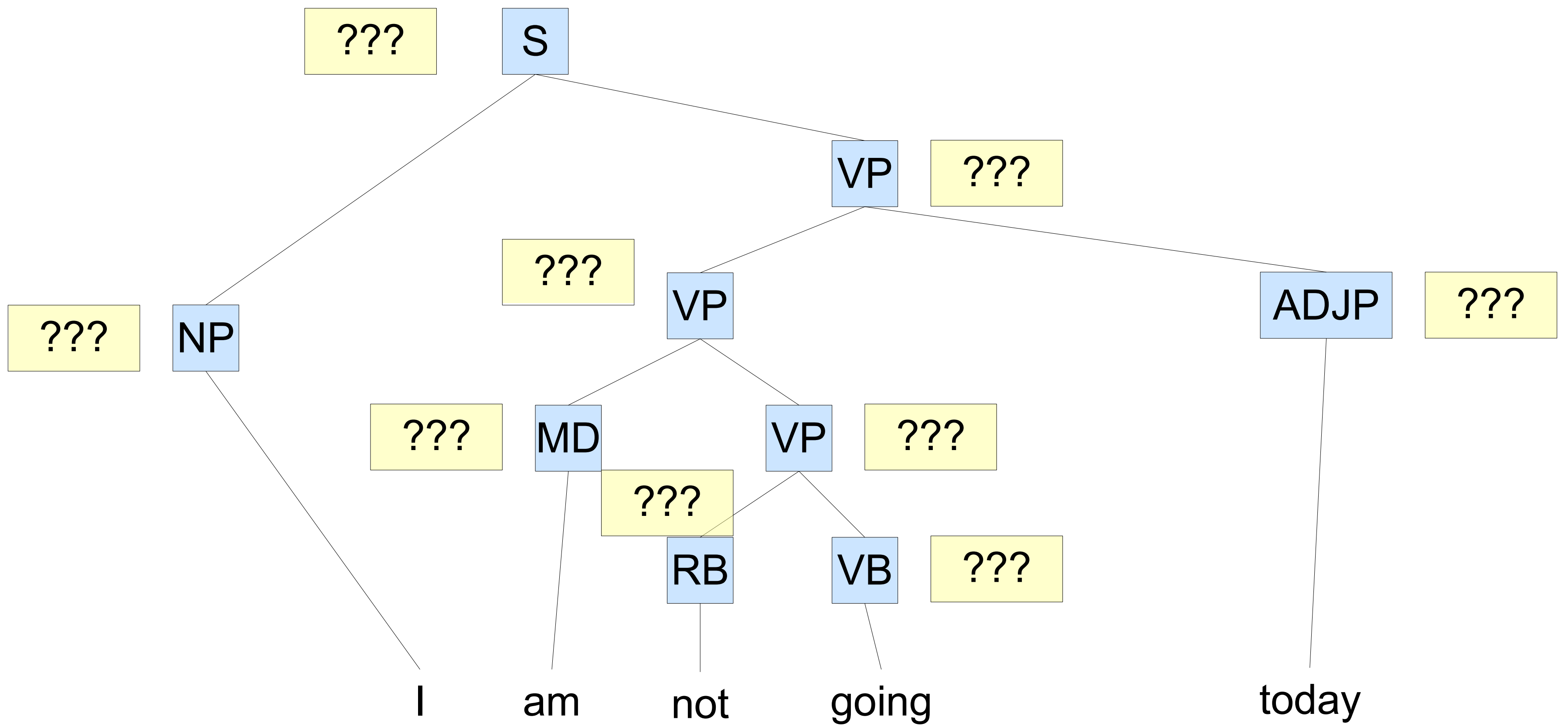
... and we label (some subset) of the nodes with rules.



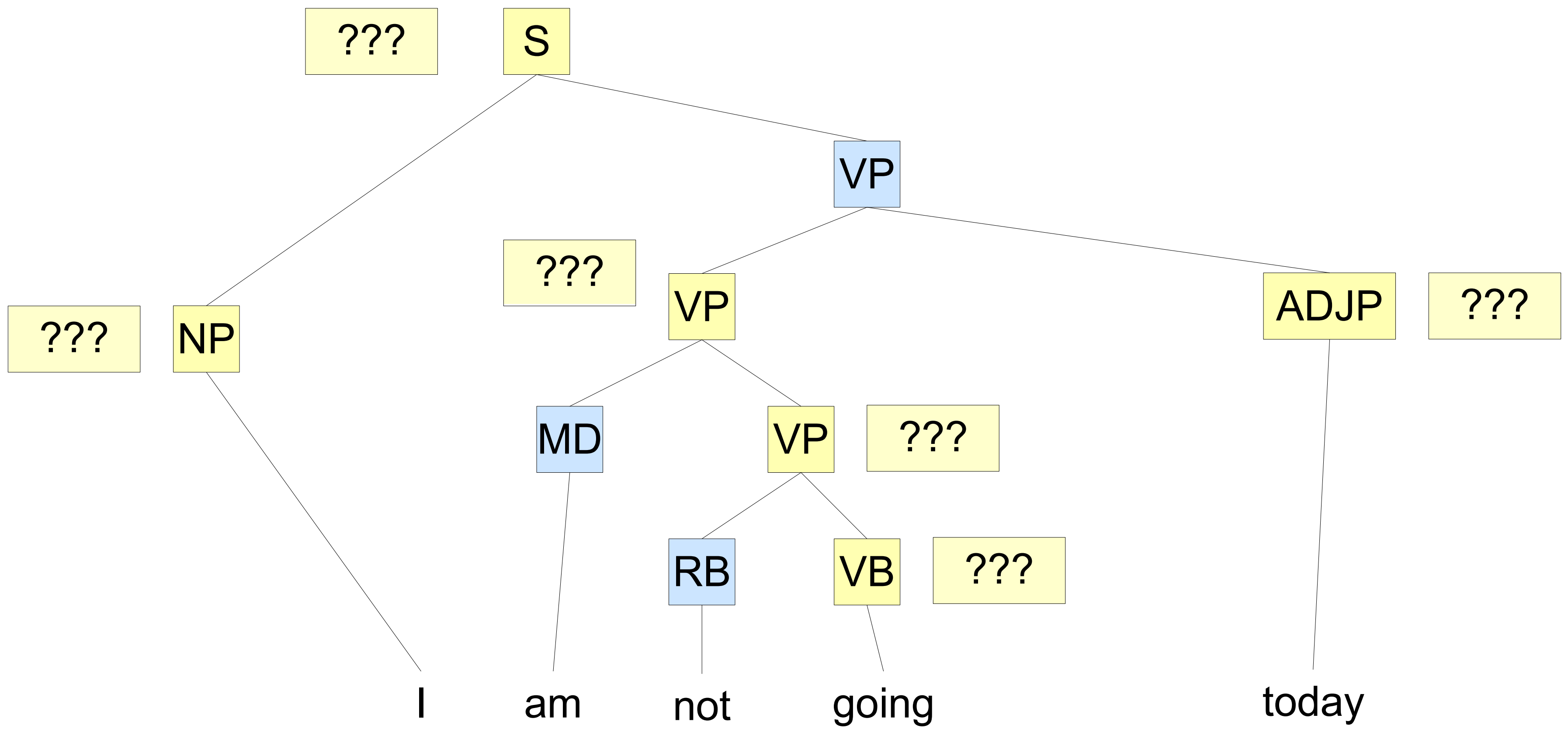
In this way, we've reduced the somewhat ambiguous translation task to the more concrete task of labeling the nodes of a tree.



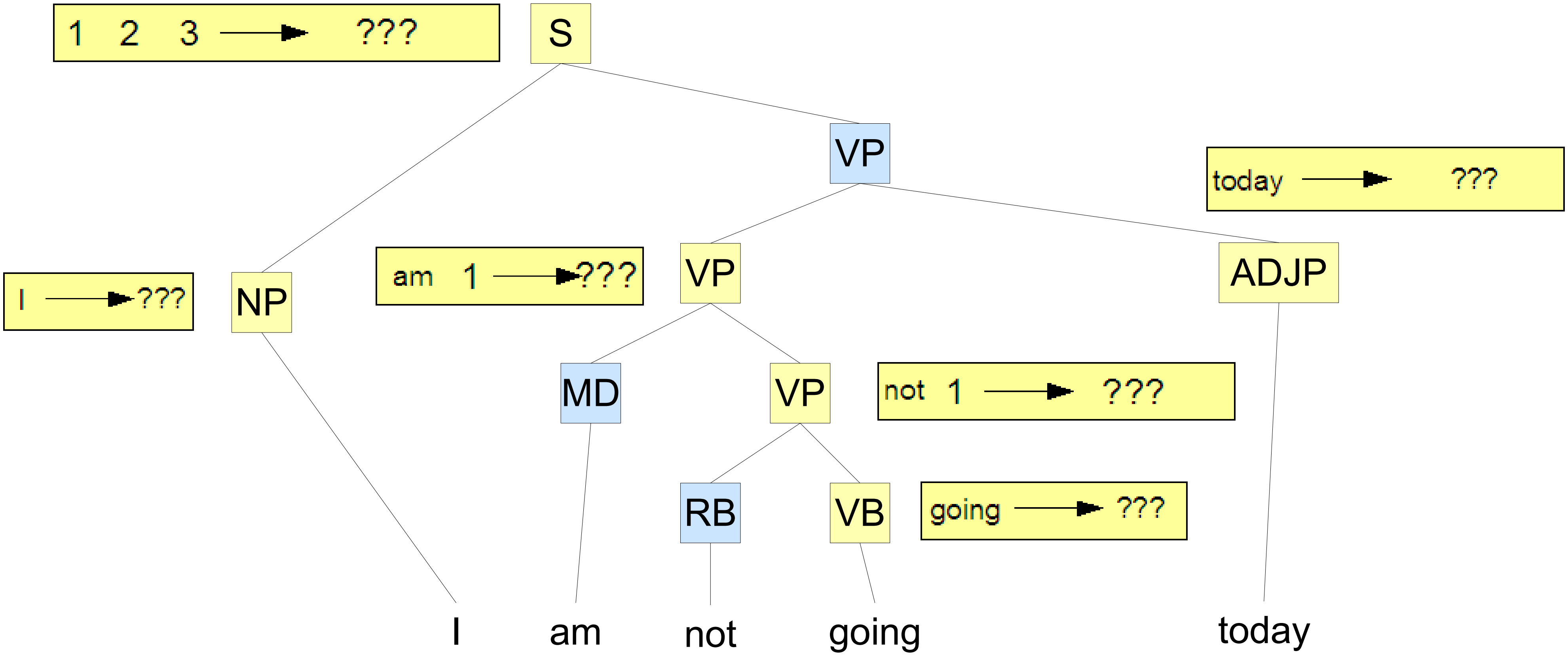
But we still need to break down the task a bit further. We can't really ask a generative process, as its first decision, to guess with a high degree of accuracy that it should label the root node in the correct way. It doesn't even know yet how many variables there will be (which will only be clear once we know which nodes will actually be receiving rule labels).



Instead, we'll begin the generative story with a simpler task. We'll simply go to each node, and decide whether it will **have a rule** or whether it will **not have a rule**.

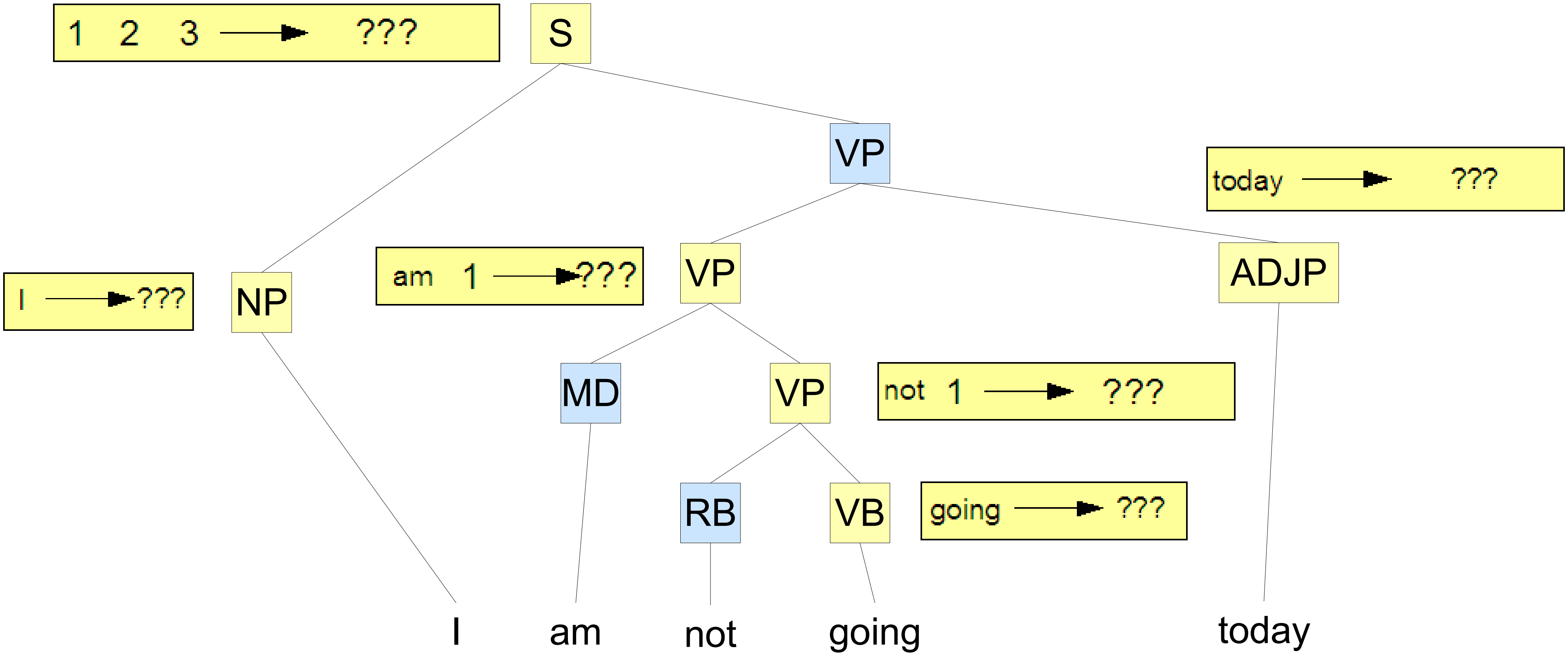


Note that this is a binary decision. Once we've done that, notice that our job is suddenly easier, because:



Note that this is a binary decision. Once we've done that, notice that our job is suddenly easier, because:

The LHS of every rule **is determined**.



Now, what do we have to do to complete our generative story? Simply fill in the **right-hand side** of each rule.

So let's devise a simple generative story for the RHS of a rule...

Consider two of the rules from our sample tree, and take particular notice of the RHS.

$1\ 2\ 3 \longrightarrow 3\ ,\ 1\ 2$

not $1 \longrightarrow$ ne 1 pas

Conceivably in our corpus we will find many very similar looking right hand sides.

1 2 3 → 3 , 1 2	not 1 → ne 1 pas
1 2 3 → 1 , 2 3	not 1 2 → ne 1 2 pas
1 2 3 4 → 4 , 1 2 3	not 1 2 → ne 2 1 pas
1 2 → 1 , 2	

We can view these RHS as instances of a general schema. We begin to create such a RHS by choosing a template.

1 2 3 $\longrightarrow$ 3 , 1 2	not 1 $\longrightarrow$ ne 1 pas
1 2 3 $\longrightarrow$ 1 , 2 3	not 1 2 $\longrightarrow$ ne 1 2 pas
1 2 3 4 $\longrightarrow$ 4 , 1 2 3	not 1 2 $\longrightarrow$ ne 2 1 pas
1 2 $\longrightarrow$ 1 , 2	

Rather than memorizing all of these rules separately, let's view them as instances of a more general rule. We begin to create such a rule by choosing a template.

1 2 3 → 3 , 1 2	not 1 → ne 1 pas
1 2 3 → 1 , 2 3	not 1 2 → ne 1 2 pas
1 2 3 4 → 4 , 1 2 3	not 1 2 → ne 2 1 pas
1 2 → 1 , 2	
→ X , X	→ ne X pas

Once we've chosen the RHS template, we have partially completed rules that look like the following examples. Our next task is to fill each RHS will the variables of the LHS. Let's see how this works.

1 2 3 → X , X

not 1 → ne X pas

We began the generative story of a rule RHS by choosing the RHS template.

1 2 3 ⟶ X , X

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**

Now we place (by default) the first variable in the first RHS slot.

1 2 3 $\longrightarrow$ X , X

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**

Now we ask: do we want to push this variable further to the right? Note that this is a yes/no decision. Suppose we say yes.

$$1 \quad 2 \quad 3 \longrightarrow 1 \; , \; X$$

“Do we want to push this variable further to the right?”

“Yes.”

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**
CHOOSE TO PUSH VAR 1 RIGHT? **YES**

At this point, we cannot push variable 1 further to the right, so we won't ask again. Let's move on to variable 2.

$1 \quad 2 \quad 3 \longrightarrow X \,, \; 1$

“Hmm... it looks like we can't go any further to the right.”

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**
CHOOSE TO PUSH VAR 1 RIGHT? **YES**

By default, we'll start by placing variables immediately after the previously placed variable.

$$1 \quad 2 \quad 3 \longrightarrow X \; , \; 1 \, 2$$

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**
CHOOSE TO PUSH VAR 1 RIGHT? **YES**

We can't move variable 2 further to the right, so we'll start by asking whether we want to move it further to the left.

$1 \quad 2 \quad 3 \longrightarrow X\ ,\ 1\ 2$

“Do we want to push this variable further to the left?”

“No.”

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**
CHOOSE TO PUSH VAR 1 RIGHT? **YES**
CHOOSE TO PUSH VAR 2 LEFT? **NO**

Once we've declined to move variable 2 to the right and to the left, we go on to consider variable 3.

$1 \quad 2 \quad 3 \longrightarrow X \;,\; 1\,2\,3$

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**
CHOOSE TO PUSH VAR 1 RIGHT? **YES**
CHOOSE TO PUSH VAR 2 LEFT? **NO**

We can't move to the right, so we begin by asking whether we want to move it left.

1 2 3 ⟶ X , 1 3 2

"Do we want to push this variable further to the left?"

"Yes."

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**
CHOOSE TO PUSH VAR 1 RIGHT? **YES**
CHOOSE TO PUSH VAR 2 LEFT? **NO**
CHOOSE TO PUSH VAR 3 LEFT? **YES**

Until we decline to move it left, or are unable to move the variable left, we continue to ask this question.

1 2 3 ⟶ X , 3 1 2

"Do we want to push this variable further to the left?"

"Yes."

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**
CHOOSE TO PUSH VAR 1 RIGHT? **YES**
CHOOSE TO PUSH VAR 2 LEFT? **NO**
CHOOSE TO PUSH VAR 3 LEFT? **YES**
CHOOSE TO PUSH VAR 3 LEFT? **YES**

Now that we have taken care of all of the LHS variables, we have completed our rule.

$1 \quad 2 \quad 3 \longrightarrow 3 \ , \quad 1 \ 2$

DECISIONS MADE THUS FAR:

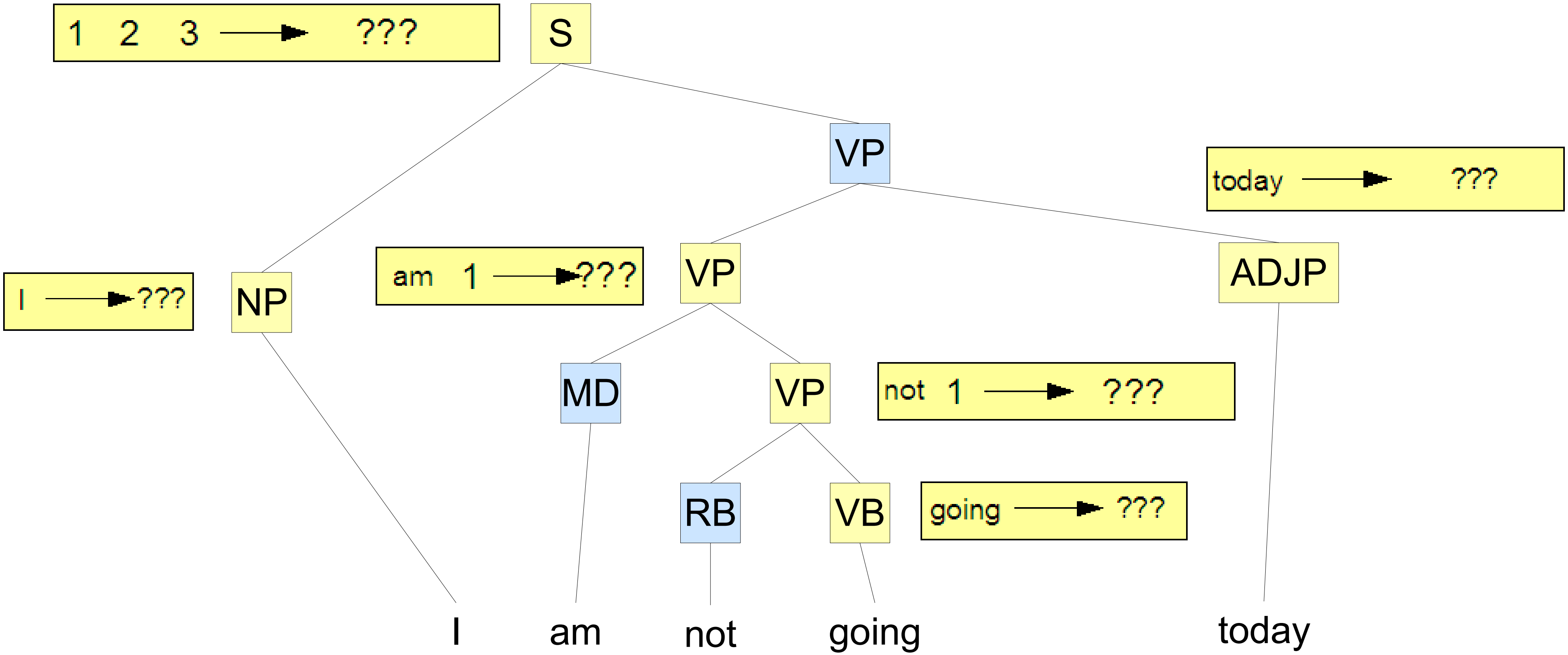
CHOOSE TEMPLATE RHS: **X , X**
CHOOSE TO PUSH VAR 1 RIGHT? **YES**
CHOOSE TO PUSH VAR 2 LEFT? **NO**
CHOOSE TO PUSH VAR 3 LEFT? **YES**
CHOOSE TO PUSH VAR 3 LEFT? **YES**
CHOOSE TO PUSH VAR 3 LEFT? **YES**

Observe that the RHS of this rule was created using a series of three basic types of decisions (template, push left, push right), two of which are binary.

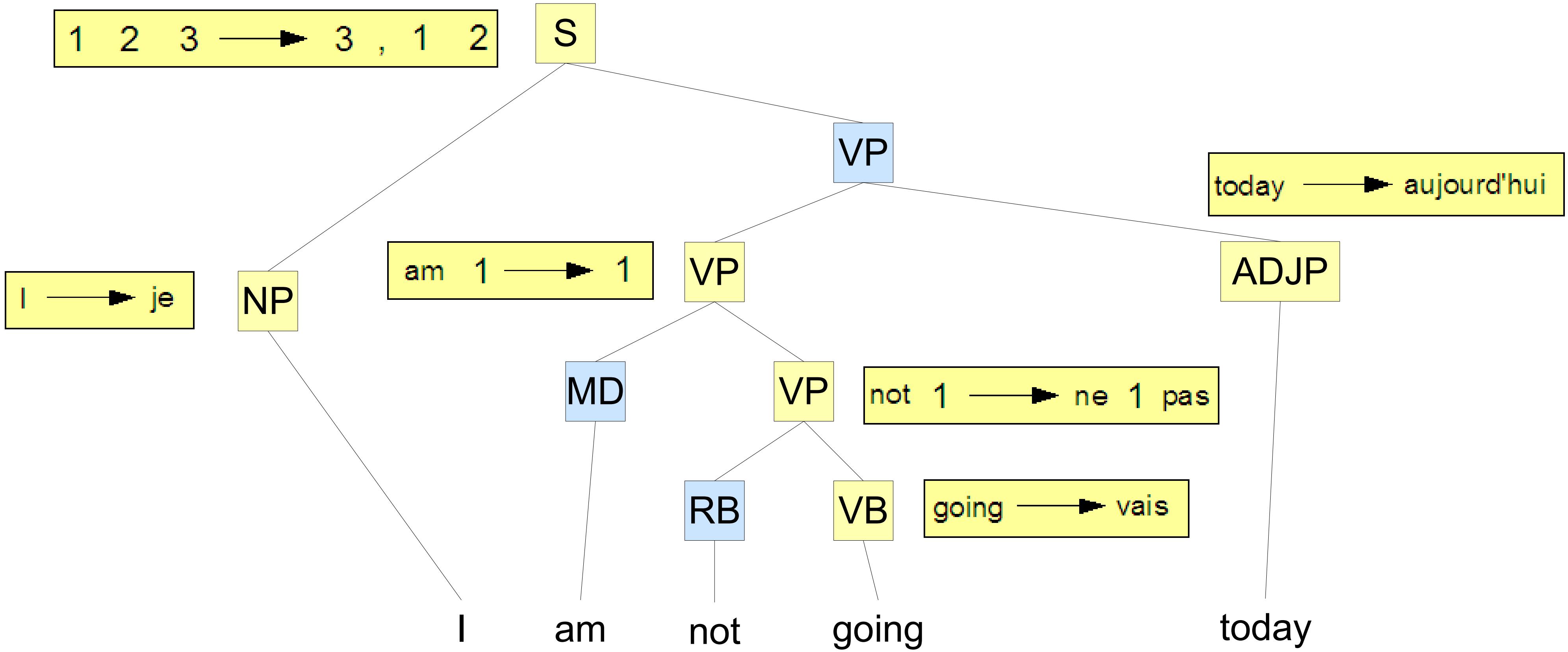
1 2 3 $\longrightarrow$ 3 , 1 2

DECISIONS MADE THUS FAR:

CHOOSE TEMPLATE RHS: **X , X**
CHOOSE TO PUSH VAR 1 RIGHT? **YES**
CHOOSE TO PUSH VAR 2 LEFT? **NO**
CHOOSE TO PUSH VAR 3 LEFT? **YES**
CHOOSE TO PUSH VAR 3 LEFT? **YES**
CHOOSE TO PUSH VAR 3 LEFT? **YES**



Now that we have a generative story for the generation of the right-hand side of a rule, we use this to go from here...



... to here. Which is exactly where we were trying to get to.

Hence, our overall process of labeling a tree with GHKM rules then boils down to four types of decisions:

(1) Whether a node should be labeled with a rule (true/false)

(2) The RHS template of the rule (open ended)

(3) Whether a variable should be pushed left in a given context (true/false)

(4) Whether a variable should be pushed right in a given context (true/false)

Except for decision (2), all of these decisions are binary, and easy for a classifier to crunch on.

(1) Whether a node should be labeled with a rule (true/false)

(2) The RHS template of the rule (open ended)

(3) Whether a variable should be pushed left in a given context (true/false)

(4) Whether a variable should be pushed right in a given context (true/false)

Except for decision (2), all of these decisions are binary, and easy for a classifier to crunch on.

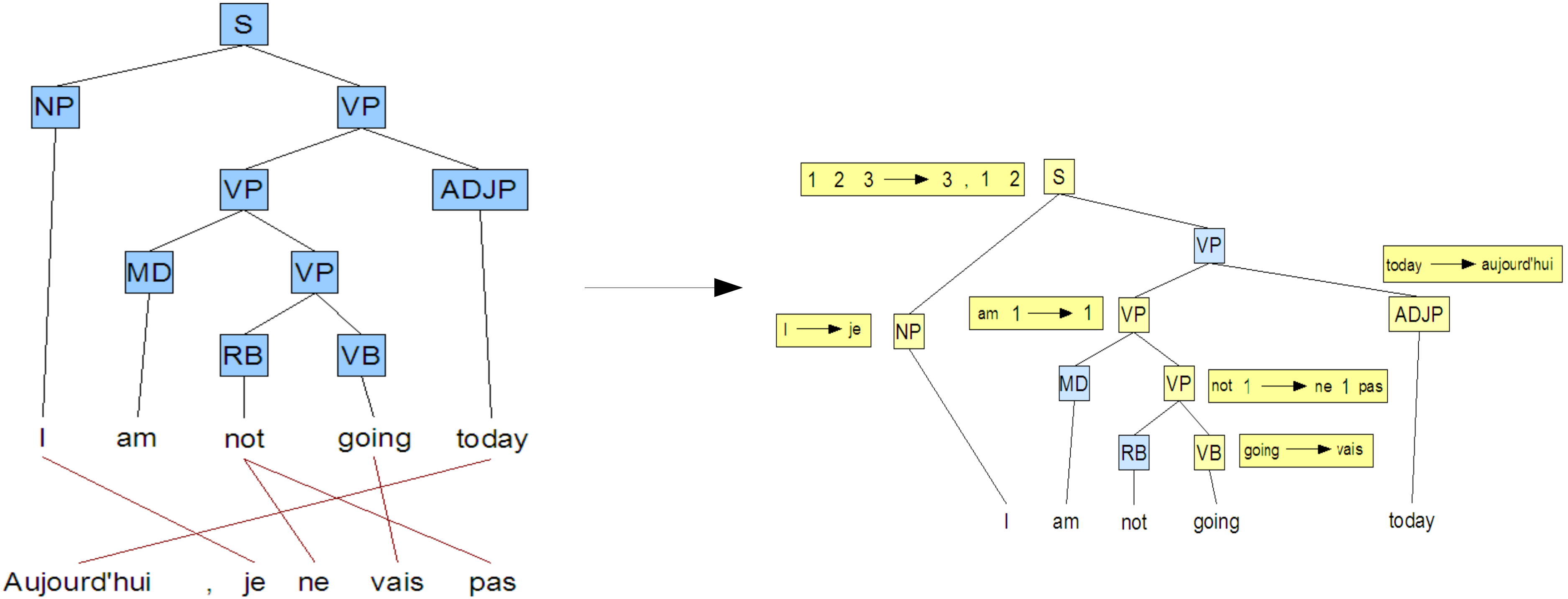
Even decision (2) is not an extremely difficult decision, if we learn a separate distribution for each LHS template.

(2) The RHS template of the rule (open ended)

Consider LHS template “X”. Much of the time the corresponding RHS template will simply be “X” as well. Or consider LHS template “the”. The number of RHS template possibilities is not enormous (for French, the major candidates would be “le” and “la”).

At this point we have shown:

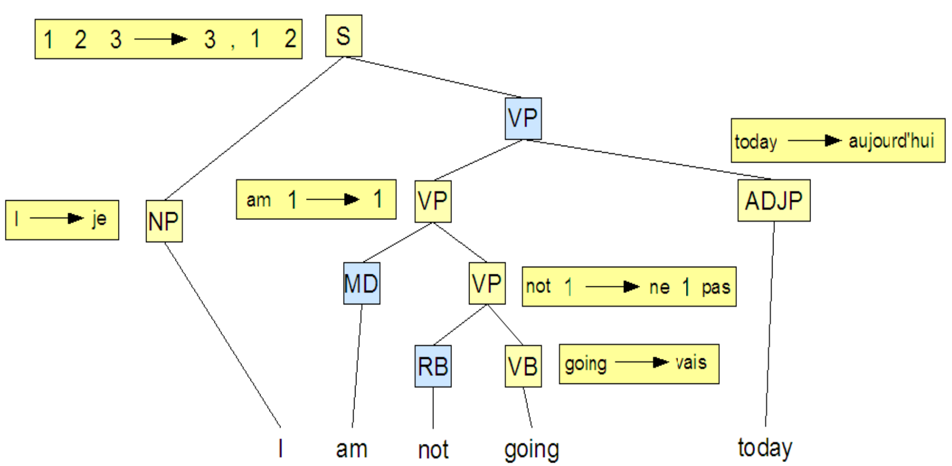
(1) how to take an aligned tree-string pair in our training corpus, and convert it to a GHKM tree representation.



At this point we have shown:

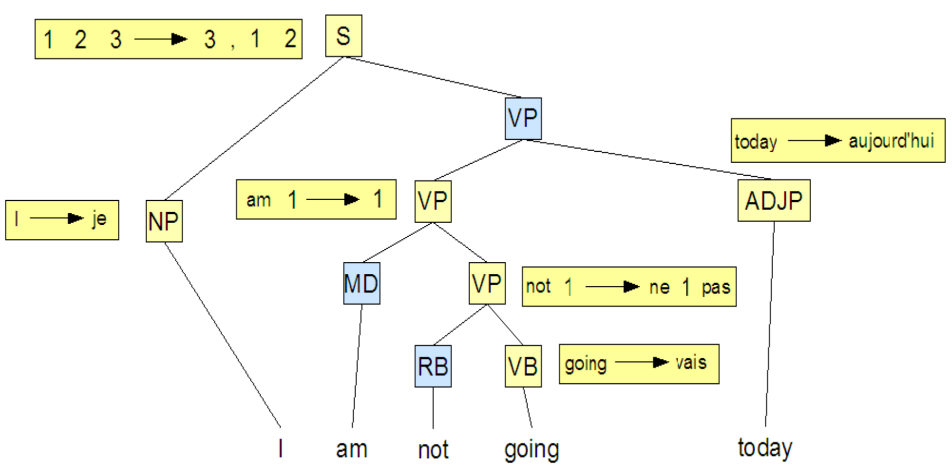
(1) how to take an aligned tree-string pair in our training corpus, and convert it to a GHKM tree representation.
(2) how to take this GHKM tree presentation, and convert it to a series of (mostly binary) decisions

RULE (root): **YES**
RULE (NP): **YES**
RULE (VP1): **NO**
RULE (VP2): **YES**
...
RHS TEMPLATE(root): **X , X**
VAR 1 RIGHT(root)? **YES**
VAR 2 LEFT(root)? **NO**
VAR 3 LEFT(root)? **YES**
VAR 3 LEFT(root)? **YES**
VAR 3 LEFT(root)? **YES**
RHS TEMPLATE(NP): **je**
RHS TEMPLATE(VP2): **X**
RHS TEMPLATE(VP3): **ne X pas**
RHS TEMPLATE(VB): **vais**
RHS TEMP(ADJP): **aujourd'hui**



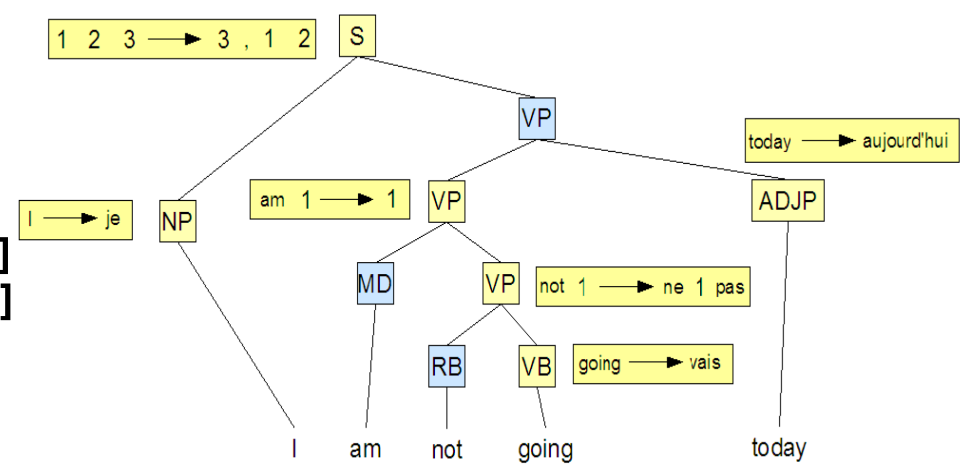
We can easily turn these into feature vectors, by jotting down relevant feature information at the point when these decisions were made.

RULE (root): **YES [NT = S; HEAD = am]**
RULE (NP): **YES [NT = NP; HEAD = I]**
RULE (VP1): **NO [NT = VP; HEAD = am]**
RULE (VP2): **YES [NT = VP; HEAD = am]**
...
RHS TEMPLATE(root): **X , X [NT=S]**
VAR 1 RIGHT(root)? **YES [VARNT=NP; PUSH=,]**
VAR 2 LEFT(root)? **NO [VARNT=VP; PUSH = NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=VP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=,]**
RHS TEMPLATE(NP): **je [NT=NP; WD=I]**
RHS TEMPLATE(VP2): **X [NT=VP]**
RHS TEMPLATE(VP3): **ne X pas [NT=VP; WD=not]**
RHS TEMPLATE(VB): **vais [NT=VB; WD=going]**
RHS TEMP(ADJP): **aujourd'hui [WD=today]**



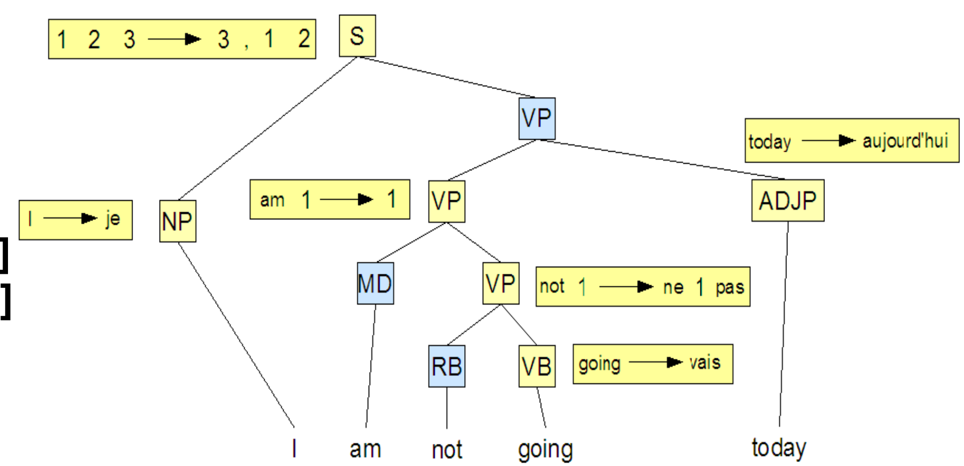
Here we content ourselves with simple features. However there are no restrictions. We can provide appropriate features to enable an intelligent and informed decision about whether to translate “the” as “le” or “la,” given simple clues (e.g. whether the associated noun ends in “ion”) or more structured information that we might choose to annotate the parse tree with (like the gender of the associated noun).

RULE (root): **YES [NT = S; HEAD = am]**
RULE (NP): **YES [NT = NP; HEAD = I]**
RULE (VP1): **NO [NT = VP; HEAD = am]**
RULE (VP2): **YES [NT = VP; HEAD = am]**
...
RHS TEMPLATE(root): **X , X [NT=S]**
VAR 1 RIGHT(root)? **YES [VARNT=NP; PUSH=,]**
VAR 2 LEFT(root)? **NO [VARNT=VP; PUSH = NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=VP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=,]**
RHS TEMPLATE(NP): **je [NT=NP; WD=I]**
RHS TEMPLATE(VP2): **X [NT=VP]**
RHS TEMPLATE(VP3): **ne X pas [NT=VP; WD=not]**
RHS TEMPLATE(VB): **vais [NT=VB; WD=going]**
RHS TEMP(ADJP): **aujourd'hui [WD=today]**



The system can also incorporate intelligent context for the reordering (RIGHT/LEFT) decisions. For instance, if I have a variable corresponding to an adjective, should I push it past its associated noun, based on what I know of this language?

RULE (root): **YES [NT = S; HEAD = am]**
RULE (NP): **YES [NT = NP; HEAD = I]**
RULE (VP1): **NO [NT = VP; HEAD = am]**
RULE (VP2): **YES [NT = VP; HEAD = am]**
...
RHS TEMPLATE(root): **X , X [NT=S]**
VAR 1 RIGHT(root)? **YES [VARNT=NP; PUSH=,]**
VAR 2 LEFT(root)? **NO [VARNT=VP; PUSH = NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=VP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=,]**
RHS TEMPLATE(NP): **je [NT=NP; WD=I]**
RHS TEMPLATE(VP2): **X [NT=VP]**
RHS TEMPLATE(VP3): **ne X pas [NT=VP; WD=not]**
RHS TEMPLATE(VB): **vais [NT=VB; WD=going]**
RHS TEMP(ADJP): **aujourd'hui [WD=today]**



Training at this stage is simple. First we divide the feature vectors up according to what type of decision they're making.

RULE (root): **YES [NT = S; HEAD = am]**
RULE (NP): **YES [NT = NP; HEAD = I]**
RULE (VP1): **NO [NT = VP; HEAD = am]**
RULE (VP2): **YES [NT = VP; HEAD = am]**
...
RHS TEMPLATE(root): **X , X [NT=S]**
VAR 1 RIGHT(root)? **YES [VARNT=NP; PUSH=,]**
VAR 2 LEFT(root)? **NO [VARNT=VP; PUSH = NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=VP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=,]**
RHS TEMPLATE(NP): **je [NT=NP; WD=I]**
RHS TEMPLATE(VP2): **X [NT=VP]**
RHS TEMPLATE(VP3): **ne X pas [NT=VP; WD=not]**
RHS TEMPLATE(VB): **vais [NT=VB; WD=going]**
RHS TEMP(ADJP): **aujourd'hui [WD=today]**

Training at this stage is simple. First we divide the feature vectors up according to what type of decision they're making.

RULE (root): **YES [NT = S; HEAD = am]**
RULE (NP): **YES [NT = NP; HEAD = I]**
RULE (VP1): **NO [NT = VP; HEAD = am]**
RULE (VP2): **YES [NT = VP; HEAD = am]**
...
RHS TEMPLATE(root): **X , X [NT=S]**
VAR 1 RIGHT(root)? **YES [VARNT=NP; PUSH=,]**
VAR 2 LEFT(root)? **NO [VARNT=VP; PUSH = NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=VP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=,]**
RHS TEMPLATE(NP): **je [NT=NP; WD=I]**
RHS TEMPLATE(VP2): **X [NT=VP]**
RHS TEMPLATE(VP3): **ne X pas [NT=VP; WD=not]**
RHS TEMPLATE(VB): **vais [NT=VB; WD=going]**
RHS TEMP(ADJP): **aujourd'hui [WD=today]**

Training at this stage is simple. First we divide the feature vectors up according to what type of decision they're making.

RULE (root): **YES [NT = S; HEAD = am]**
RULE (NP): **YES [NT = NP; HEAD = I]**
RULE (VP1): **NO [NT = VP; HEAD = am]**
RULE (VP2): **YES [NT = VP; HEAD = am]**

RHS TEMPLATE(root): **X , X [NT=S]**
RHS TEMPLATE(NP): **je [NT=NP; WD=I]**
RHS TEMPLATE(VP2): **X [NT=VP]**
RHS TEMPLATE(VP3): **ne X pas [NT=VP; WD=not]**
RHS TEMPLATE(VB): **vais [NT=VB; WD=going]**
RHS TEMP(ADJP): **aujourd'hui [WD=today]**

VAR 2 LEFT(root)? **NO [VARNT=VP; PUSH = NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=VP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=,]**

VAR 1 RIGHT(root)? **YES [VARNT=NP; PUSH=,]** Then we train a classifier for each decision type, based on these training vectors. We can use any off-the-shelf learning software, and any learning method, though we prefer "soft" learners that induce a probability distribution rather than a hard decision.

RULE (root): **YES [NT = S; HEAD = am]**
RULE (NP): **YES [NT = NP; HEAD = I]**
RULE (VP1): **NO [NT = VP; HEAD = am]**
RULE (VP2): **YES [NT = VP; HEAD = am]**

RHS TEMPLATE(root): **X , X [NT=S]**
RHS TEMPLATE(NP): **je [NT=NP; WD=I]**
RHS TEMPLATE(VP2): **X [NT=VP]**
RHS TEMPLATE(VP3): **ne X pas [NT=VP; WD=not]**
RHS TEMPLATE(VB): **vais [NT=VB; WD=going]**
RHS TEMP(ADJP): **aujourd'hui [WD=today]**

VAR 2 LEFT(root)? **NO [VARNT=VP; PUSH = NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=VP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=NP]**
VAR 3 LEFT(root)? **YES [VARNT=ADJP; PUSH=,]**

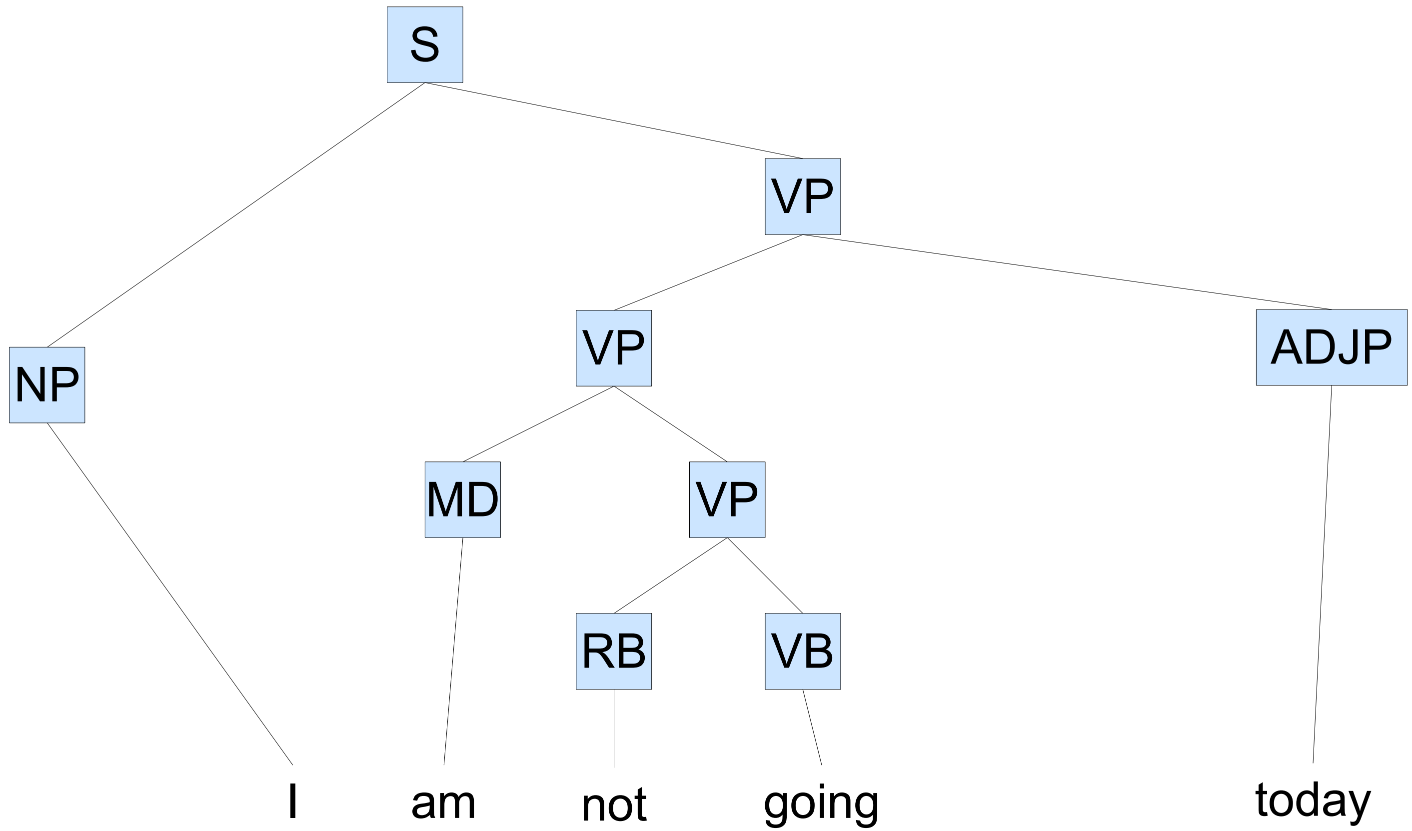
VAR 1 RIGHT(root)? **YES [VARNT=NP; PUSH=,]** Then we train a classifier for each decision type, based on this training feature vectors. We can use any off-the-shelf learning software, and any learning method, though we prefer “soft” learners that induce a probability distribution rather than a hard decision.

RULE CLASSIFIER

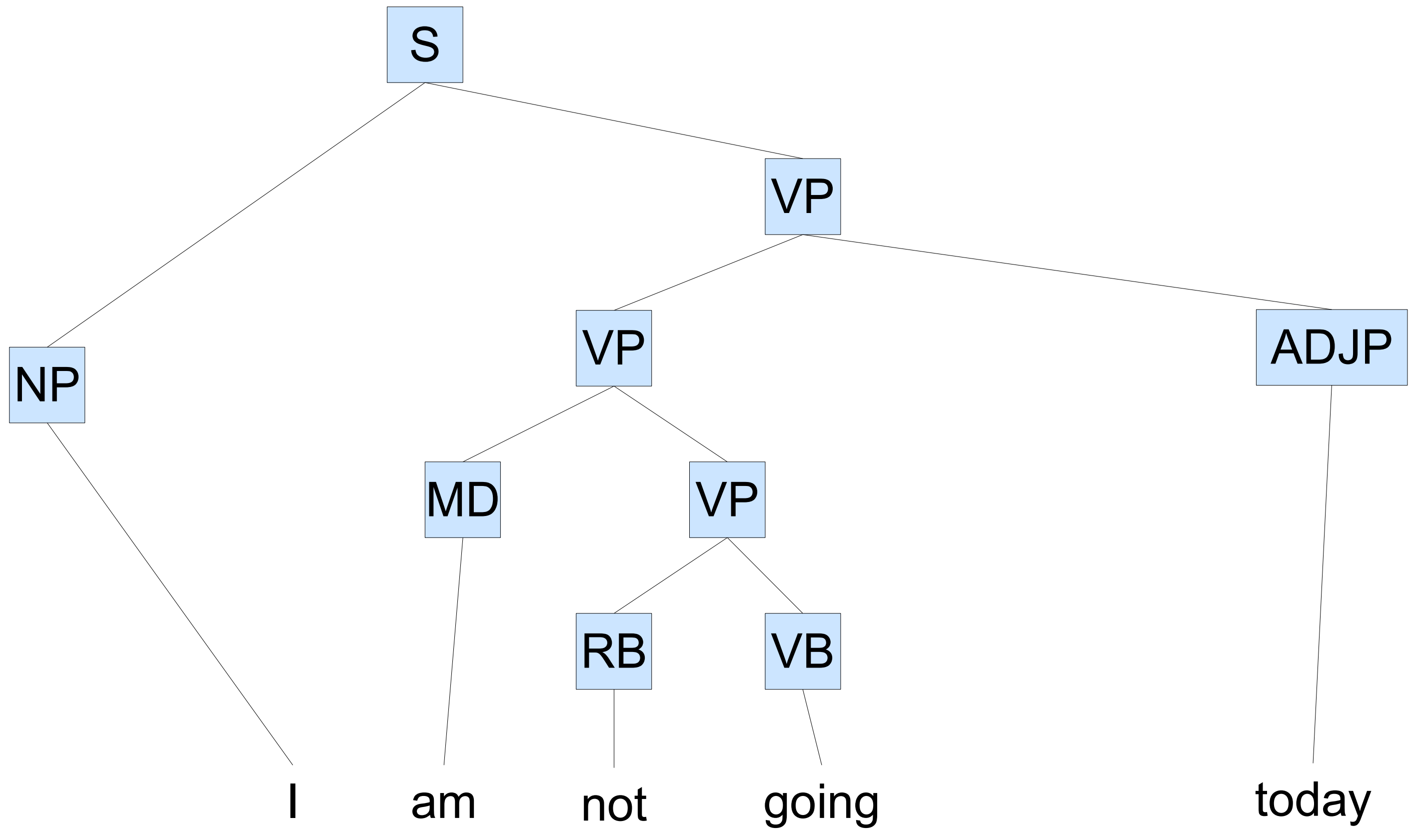
RHS TEMPLATE CLASSIFIER

VAR LEFT CLASSIFIER

VAR RIGHT CLASSIFIER



Decoding is also straightforward. We simply find the best overall assignment to our generative model according to the classifiers we've learned.



Because of the generality (and therefore flexibility) of our approach, the optimal solution cannot always be determined in polynomial time. We settle for a brute-force search to find the best solution.

However... what we can do is use depth-first branch-and-bound search, i.e. greedily find a good solution in linear time, and then use this solution to help prune the search space as we attempt to find better solutions.

The sharper the distributions of the assignment process, the faster the search process will go. In the extreme case, where all distributions have all of their probability weight concentrated on a single domain element (a hard classifier), DFBnB is linear-time. In any event, DFBnB can be cut off at any time (once the initial greedy solution is found) and used as a heuristic search.

The main advantages of using this probabilistic approach over, say, some constrained model for which we can use dynamic programming to decode polynomially are:

We have a **practically unlimited flexibility** to include history into our decisions.

We can use **plug-and-play discriminative training software** to train our system, adopting decision trees or MaxEnt regression with equal ease.

Heuristic decoding is faster than using a polynomial dynamic programming approach.

So:

We've developed an early version of a translator which casts the machine translation problem as a graph labeling task.

This system has the potential to use syntactic information freely and flexibly to make key decisions during the translation process.

Early results seem to indicate that the basics of reordering and syntax can be learned with a relatively small training corpus.

Preliminary results

- Comparison (English $\rightarrow$ German):

  (1) "Phrasal" StatMT approach (Pharaoh)

  (2) Tree labeling approach, based on Collins' parser

  (3) Tree labeling approach based on English ParGram grammar

  (4) Professional translation

Uses language model of German in addition (trained on large corpus)

Translation models trained on 50,000 sentence pairs

Preliminary training on 5000 sentence pairs

Corpus: subset of Europarl consisting of 60000 sentences of length 8-17 words – divided into training (50000), dev (5000), and test (5000).

We believe that this is a fundamental element .

(1) wir halten dies für eine grundlegende element .
(1) we believe that a fundamental element .
(2) wir glauben , dass diesen ist ein grundlegendes element .
(2) we believe that this is a fundamental element .
(3) wir meinen , dass dies eine grundlegende element ist.
(3) We believe that this a fundamental element is .
(4) wir denken , dass dies ein grundlegender aspekt ist .
(4) we believe that this a fundamental element is .

- system uses an incorrect morphological form
- word position is incorrect
- a word .... is missing
- untranslated source **word**

It is true that Lisbon is a programme for ten years .

(1) es ist richtig , dass lissabon ist eine programm für zehn jahren .
(1) it is true that lisbon is a program for ten years .
(2) es ist richtig , dass lissabon ist eine programm für zehn jahren .
(2) it is true that lisbon is a program for ten years .
(3) es ist **true** , dass **lisbon** eine programm für zehn jahren ist.
(3) It is true that lisbon a program for ten years is .
(4) nun ist lissabon ein programm für zehn jahre .
(4) is lisbon a program for ten years .

- system uses an incorrect morphological form
- word position is incorrect
- a word .... is missing
- untranslated source **word**

I completely agree with each of these points .

(1) ich ..... völlig einverstanden mit jedem dieser punkte .
(1) i completely in agreement with each of these points .

(2) ich bin völlig mit jedes diese fragen einer Meinung .
(2) i am completely with each these questions in agreement .

(3) ich **agree completely** mit jeder dieser punkte .
(3) i agree completely with each of these points .

(4) ich bin mit jeder einzelnen dieser aussagen voll und ganz einverstanden .
i am with each point of this statement completely in agreement .

- system uses an incorrect morphological form
- word position is incorrect
- a word .... is missing
- untranslated source **word**

However , I would like to add one point .

(1) allerdings möchte ich noch eines sagen .
(1) however would like I another thing say .
(2) ich möchte jedoch an noch einen punkt hinzufügen .
(2) i would like however another one point to add .
(3) allerdings möchte ich einen punkt **add** .
(3) however would like i one point add .
(4) aber ich möchte gern einen punkt hinzufügen .
(4) however i would like one point to add .

- system uses an incorrect morphological form
- word position is incorrect
- a word .... is missing
- untranslated source **word**

This is undoubtedly a point which warrants attention .

(1) das ist sicherlich eine punkt rechtfertigt das aufmerksamkeit .
(1) this is undoubtedly a point justifies the attention .

(2) das ist ohne zweifel eine punkt , die **warrants** beachtung .
(2) this is undoubtedly a point , which warrants attention .

(3) das ist **undoubtedly** .... sache , die **attention warrants** .
(3) this is undoubtedly point , which attention warrants .

(4) ohne jeden zweifel ist dies ein punkt , der aufmerksamkeit verdient .
(4) undoubtedly is this a point , which attention warrants .

- system uses an incorrect morphological form
- word position is incorrect
- a word .... is missing
- untranslated source **word**

Results: Summary

- Syntax-based approach seems to avoid some of the more serious ordering mistakes of phrase-based translation
- Phrasal translation incurs fewer agreement mistakes (presumably due to the target language model trained on a large corpus)
- Quantitative scores (BLEU) for syntax-based approach are statistically indistinguishable from phrase-based approach
- Although XLE-based model was trained on just 10% of the data, there are some encouraging advantages in picking up clause structure facts

Thank you!