Language and Statistics II

Lecture 14: Practical Dynamic Programming

Noah Smith

Where we left off …

- Shieber, Schabes, and Pereira: logic programming (deduction) as a way to think about and implement parsers.
- Goodman: add weights!
- No implementation.

Meanwhile, in the “real” world of parsing …

- People were actually building weighted parsers!
- Crucial: good search strategy.

Beyond Goodman (1999)

- Goodman's Algorithm: carry out deduction to build the chart (i.e., fill in items with nonzero value); then compute their values.
  - Tough part: **efficient** ordering of items.
  - For Forward/Viterbi: order by position
  - For CKY: order by width
  - In general?
- Would like efficient **execution strategy** for **arbitrary** programs.
  - Key idea: avoid unnecessary work and repropagation.

Indeed!

- The logic programs *don’t tell us how to implement the parser*!

- Is there a **generic** way to go about “compiling” a weighted logic program into a dynamic programming algorithm?
  - Yes: agenda-based DP (for Viterbi).
  - Does this generalize to arbitrary semirings?

Agenda

$S \rightarrow^{.8} NP\ VP$

$NNS \rightarrow^{.0002}$ quitters

$VP \rightarrow^{.6} RB\ VB$

$NP \rightarrow^{.1} NNS$

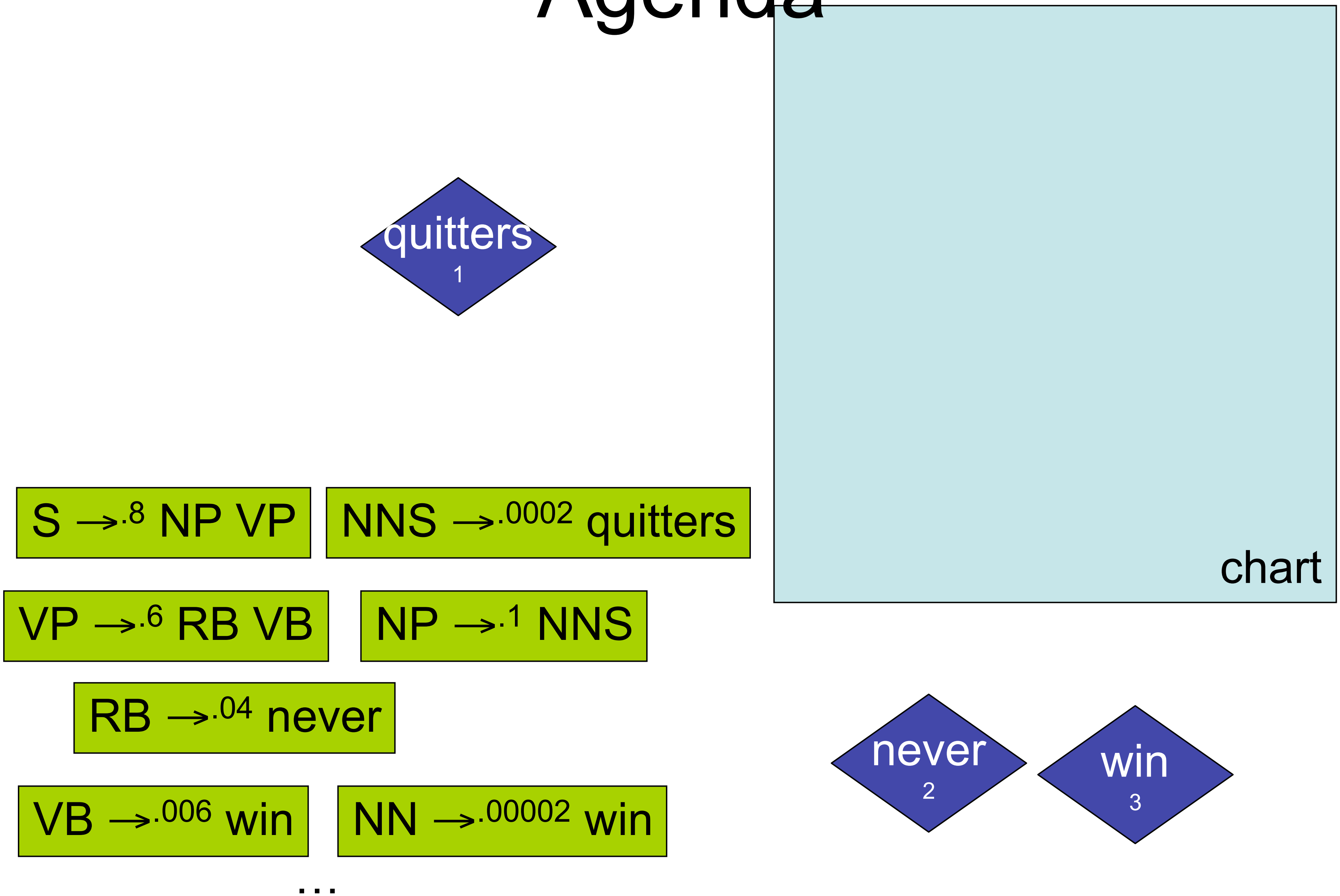
$RB \rightarrow^{.04}$ never

$VB \rightarrow^{.006}$ win

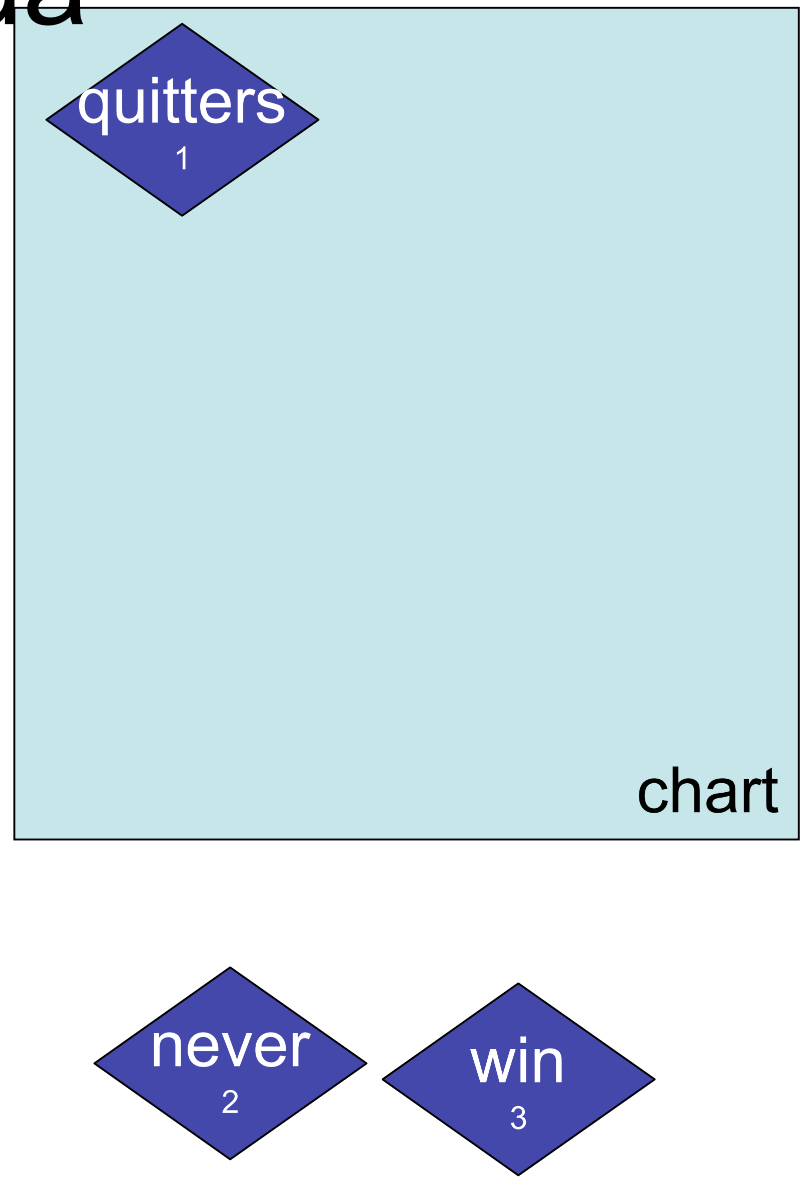
$NN \rightarrow^{.00002}$ win

…

Agenda


quitters
1
chart
$S \rightarrow^{.8} NP\ VP$
$NNS \rightarrow^{.0002}$ quitters
$VP \rightarrow^{.6} RB\ VB$
$NP \rightarrow^{.1} NNS$
$RB \rightarrow^{.04}$ never
$VB \rightarrow^{.006}$ win
$NN \rightarrow^{.00002}$ win
…
never
2
win
3

Agenda



$S \rightarrow^{.8}$ NP VP

$NNS \rightarrow^{.0002}$ quitters

$VP \rightarrow^{.6}$ RB VB

$NP \rightarrow^{.1}$ NNS

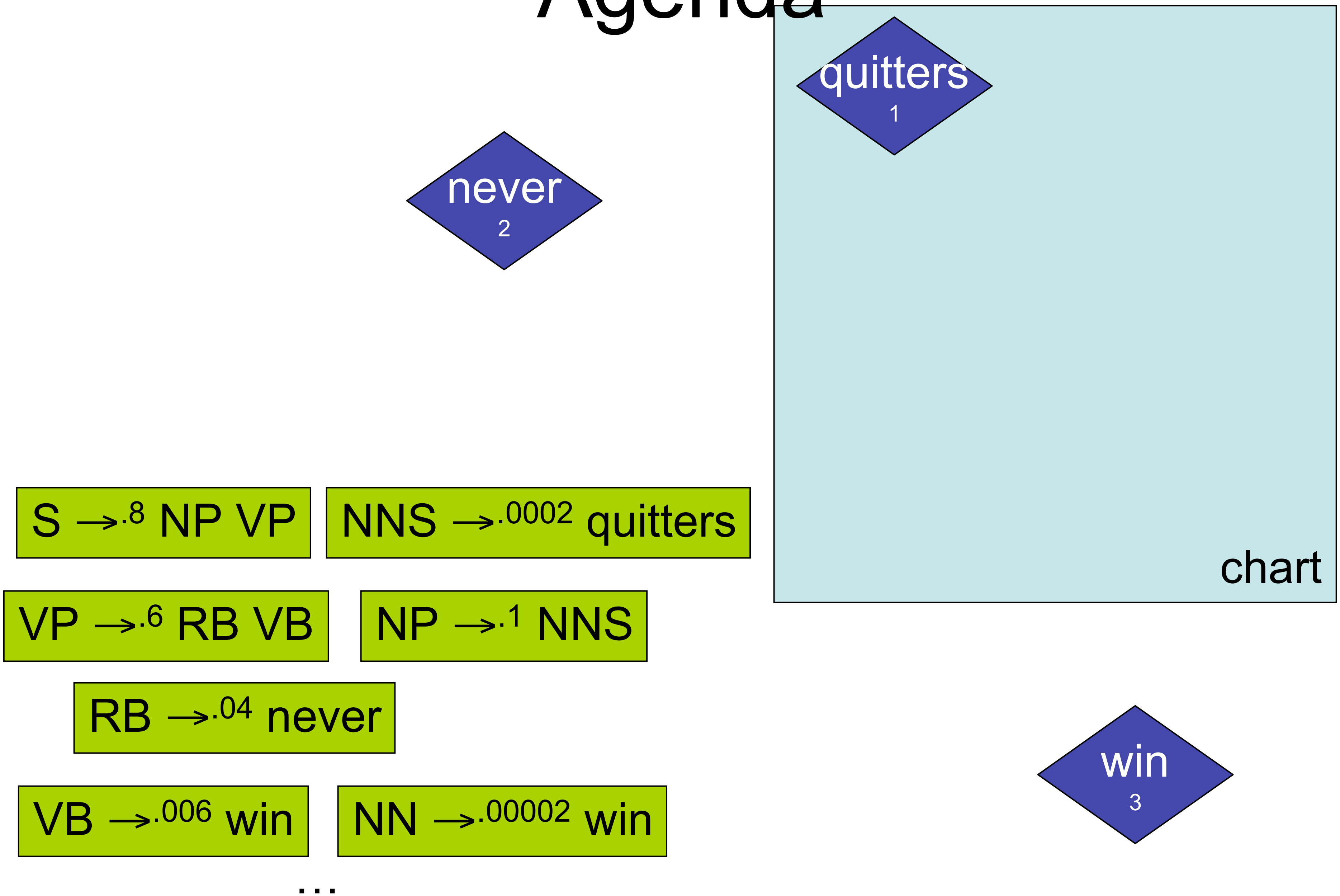
$RB \rightarrow^{.04}$ never

$VB \rightarrow^{.006}$ win

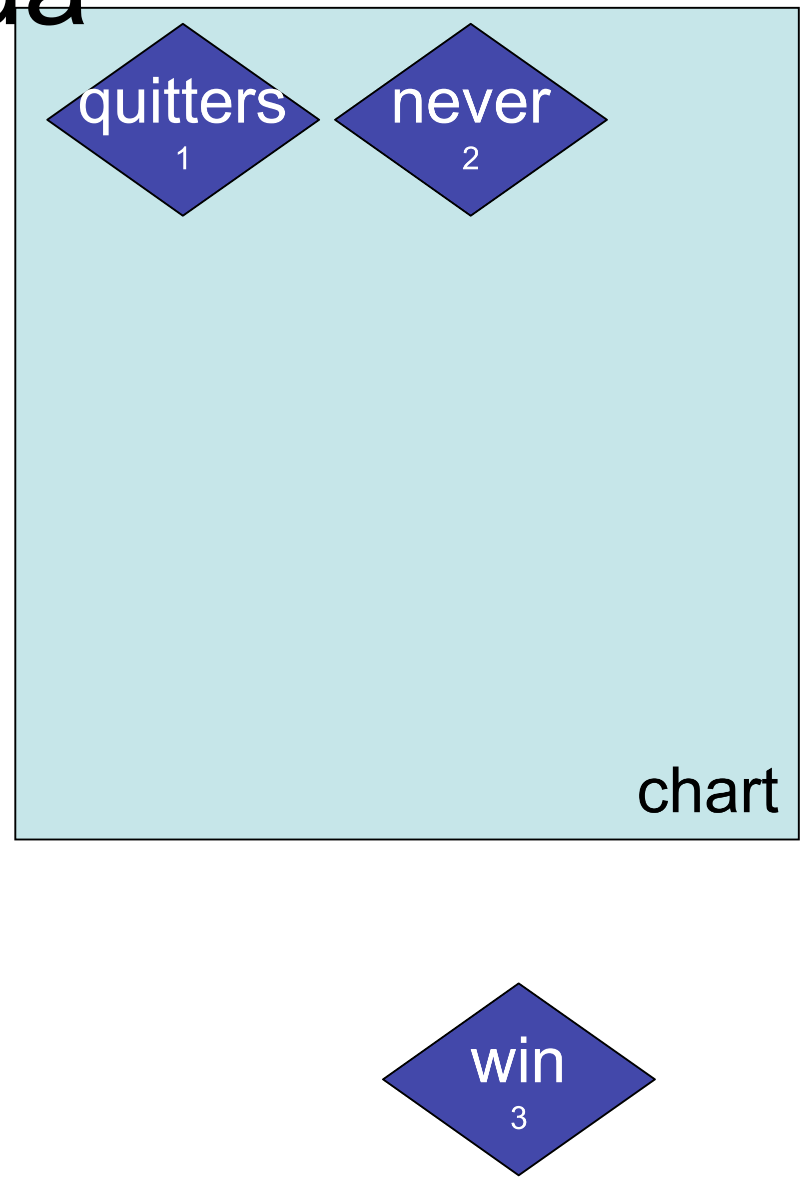
$NN \rightarrow^{.00002}$ win

…

Agenda


quitters
1
never
2
chart
S →.8 NP VP
NNS →.0002 quitters
VP →.6 RB VB
NP →.1 NNS
RB →.04 never
VB →.006 win
NN →.00002 win
…
win
3

Agenda



$S \rightarrow^{.8} NP\ VP$

$NNS \rightarrow^{.0002}$ quitters

$VP \rightarrow^{.6} RB\ VB$

$NP \rightarrow^{.1} NNS$

$RB \rightarrow^{.04}$ never

$VB \rightarrow^{.006}$ win

$NN \rightarrow^{.00002}$ win

…

Agenda

$S \rightarrow^{.8} NP\ VP$

$NNS \rightarrow^{.0002} quitters$

$VP \rightarrow^{.6} RB\ VB$

$NP \rightarrow^{.1} NNS$

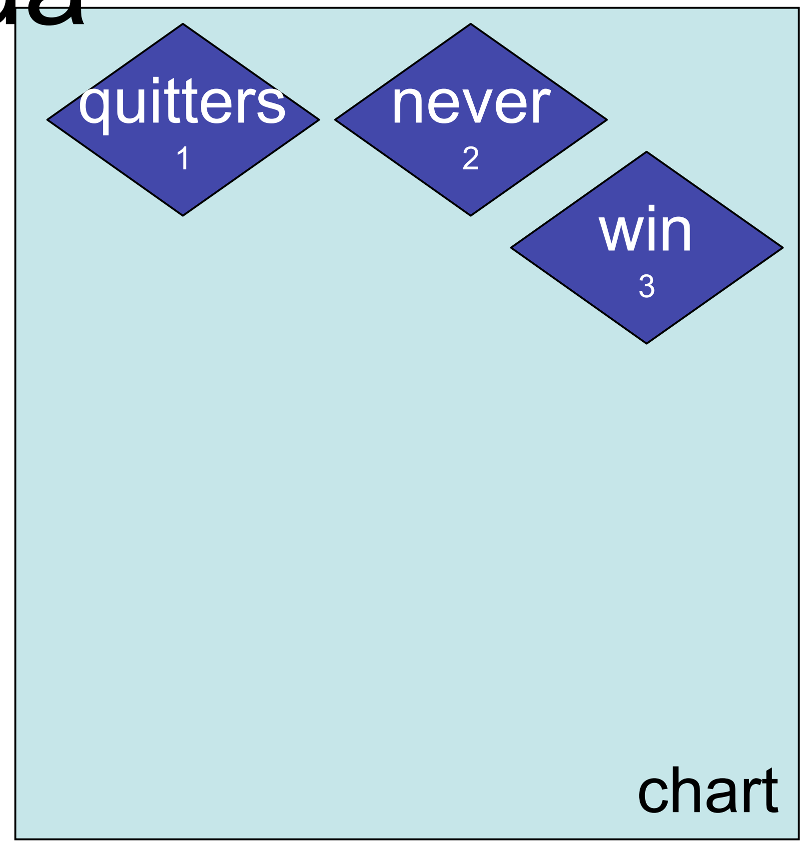
$RB \rightarrow^{.04} never$

$VB \rightarrow^{.006} win$

$NN \rightarrow^{.00002} win$

…

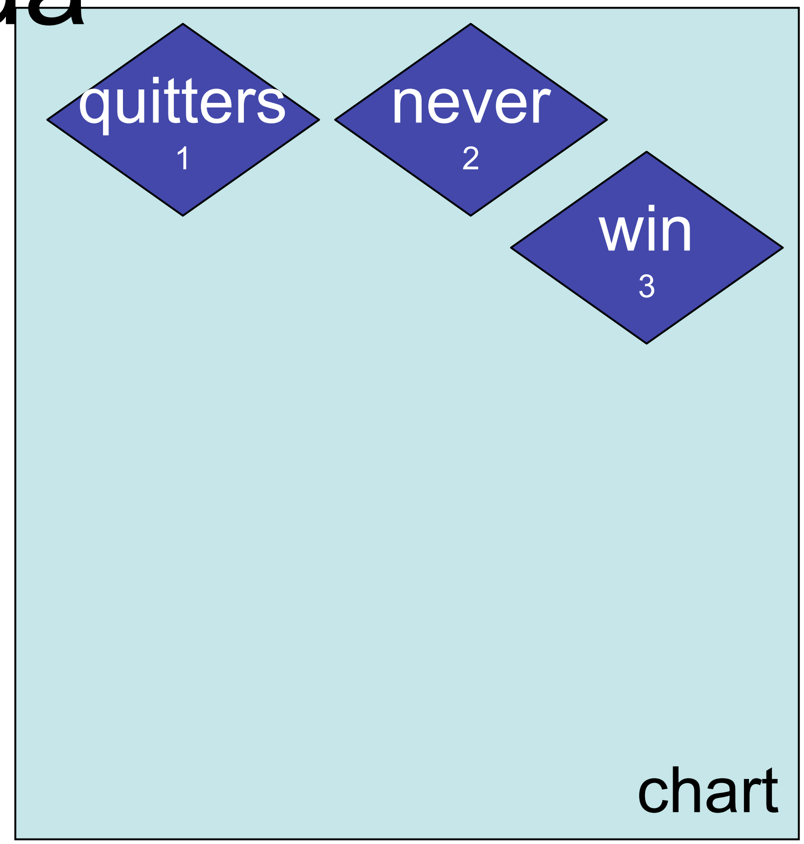
Agenda



- S $\rightarrow^{.8}$ NP VP
- NNS $\rightarrow^{.0002}$ quitters
- VP $\rightarrow^{.6}$ RB VB
- NP $\rightarrow^{.1}$ NNS
- RB $\rightarrow^{.04}$ never
- VB $\rightarrow^{.006}$ win
- NN $\rightarrow^{.00002}$ win

…

Agenda



$S \rightarrow^{.8} NP\ VP$

$NNS \rightarrow^{.0002}$ quitters

$VP \rightarrow^{.6} RB\ VB$

$NP \rightarrow^{.1} NNS$

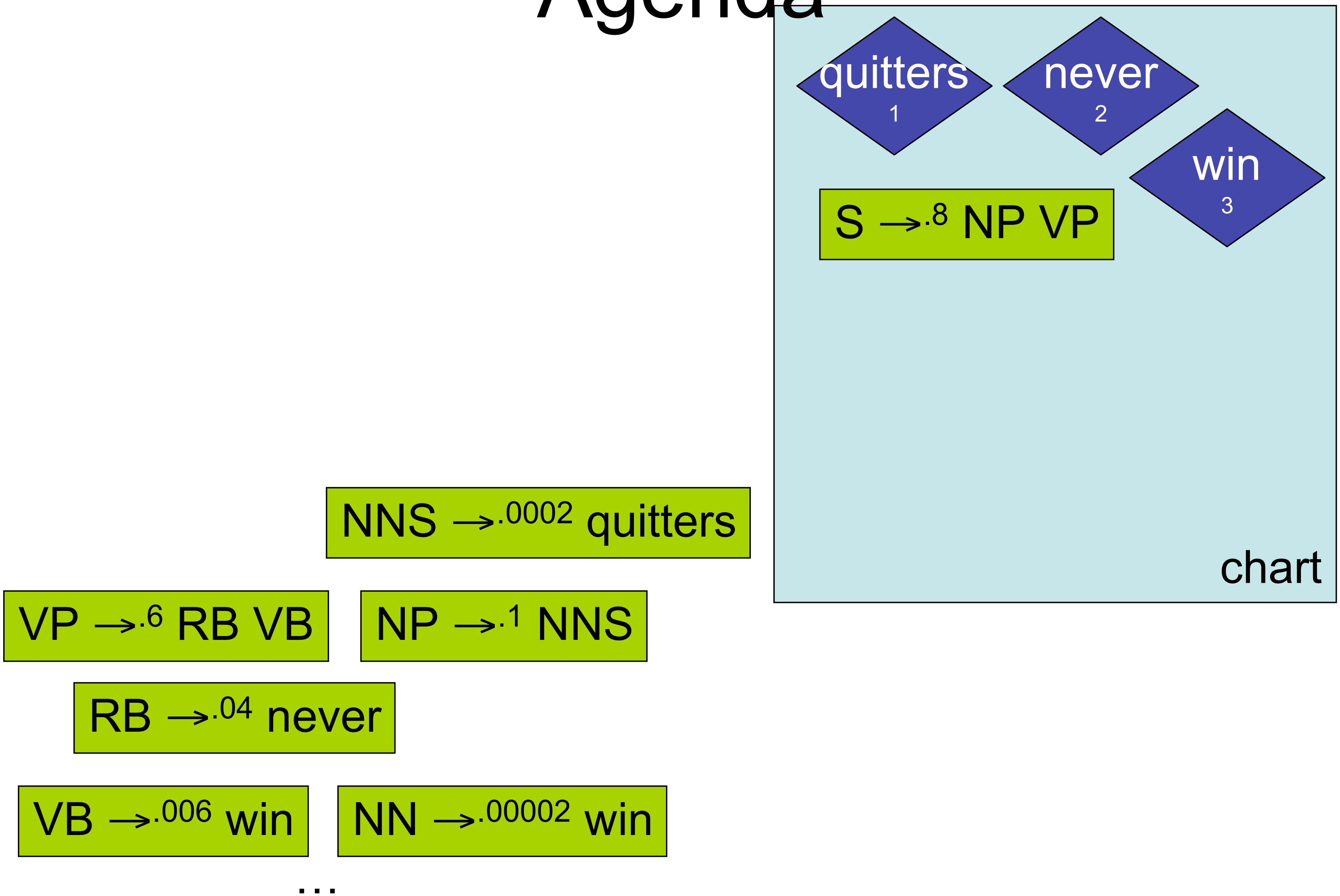
$RB \rightarrow^{.04}$ never

$VB \rightarrow^{.006}$ win

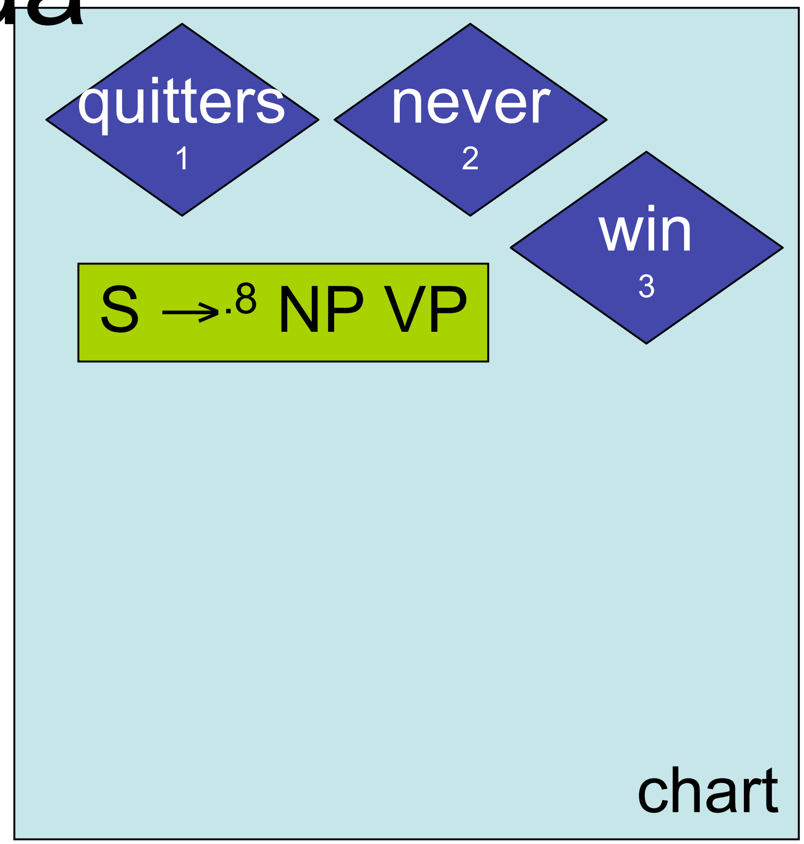
$NN \rightarrow^{.00002}$ win

…

Agenda


quitters
1
never
2
win
3
S $\rightarrow^{.8}$ NP VP
chart
NNS $\rightarrow^{.0002}$ quitters
VP $\rightarrow^{.6}$ RB VB
NP $\rightarrow^{.1}$ NNS
RB $\rightarrow^{.04}$ never
VB $\rightarrow^{.006}$ win
NN $\rightarrow^{.00002}$ win
…

Agenda



VP $\rightarrow^{.6}$ RB VB

NNS $\rightarrow^{.0002}$ quitters

NP $\rightarrow^{.1}$ NNS

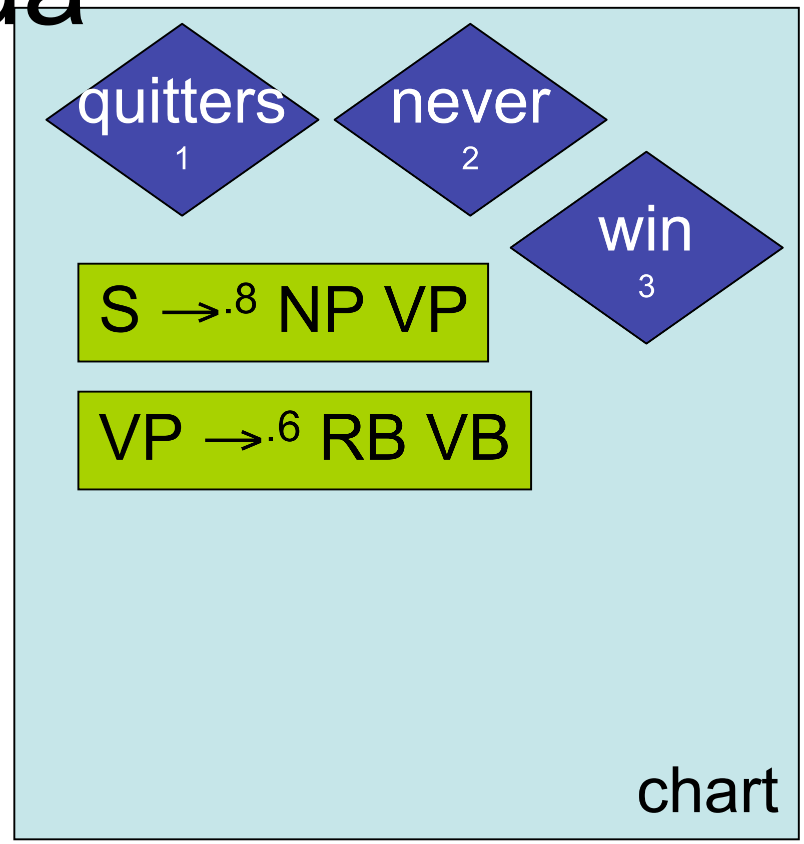
RB $\rightarrow^{.04}$ never

VB $\rightarrow^{.006}$ win

NN $\rightarrow^{.00002}$ win

…

Agenda



NNS $\rightarrow^{.0002}$ quitters

NP $\rightarrow^{.1}$ NNS

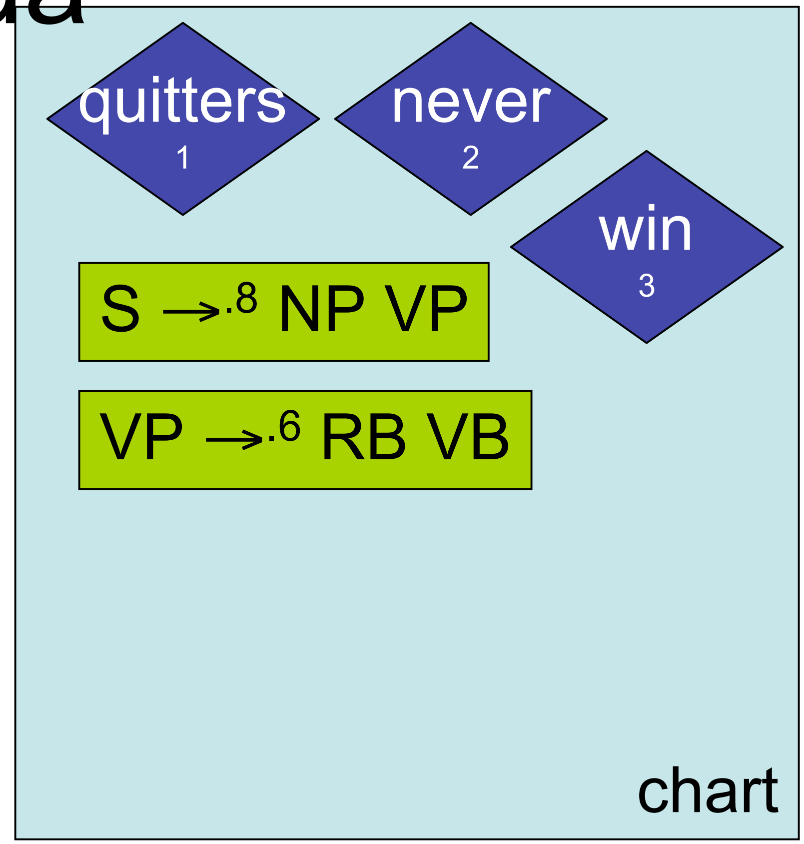
RB $\rightarrow^{.04}$ never

VB $\rightarrow^{.006}$ win

NN $\rightarrow^{.00002}$ win

…

Agenda



NP $\rightarrow^{.1}$ NNS

NNS $\rightarrow^{.0002}$ quitters

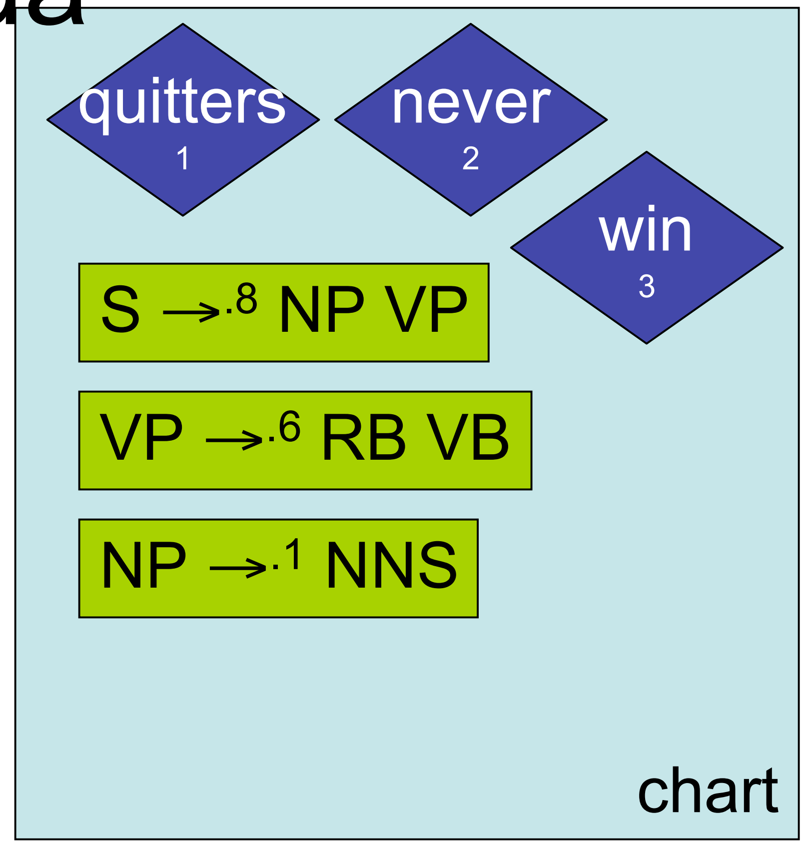
RB $\rightarrow^{.04}$ never

VB $\rightarrow^{.006}$ win

NN $\rightarrow^{.00002}$ win

…

Agenda



NNS $\rightarrow^{.0002}$ quitters

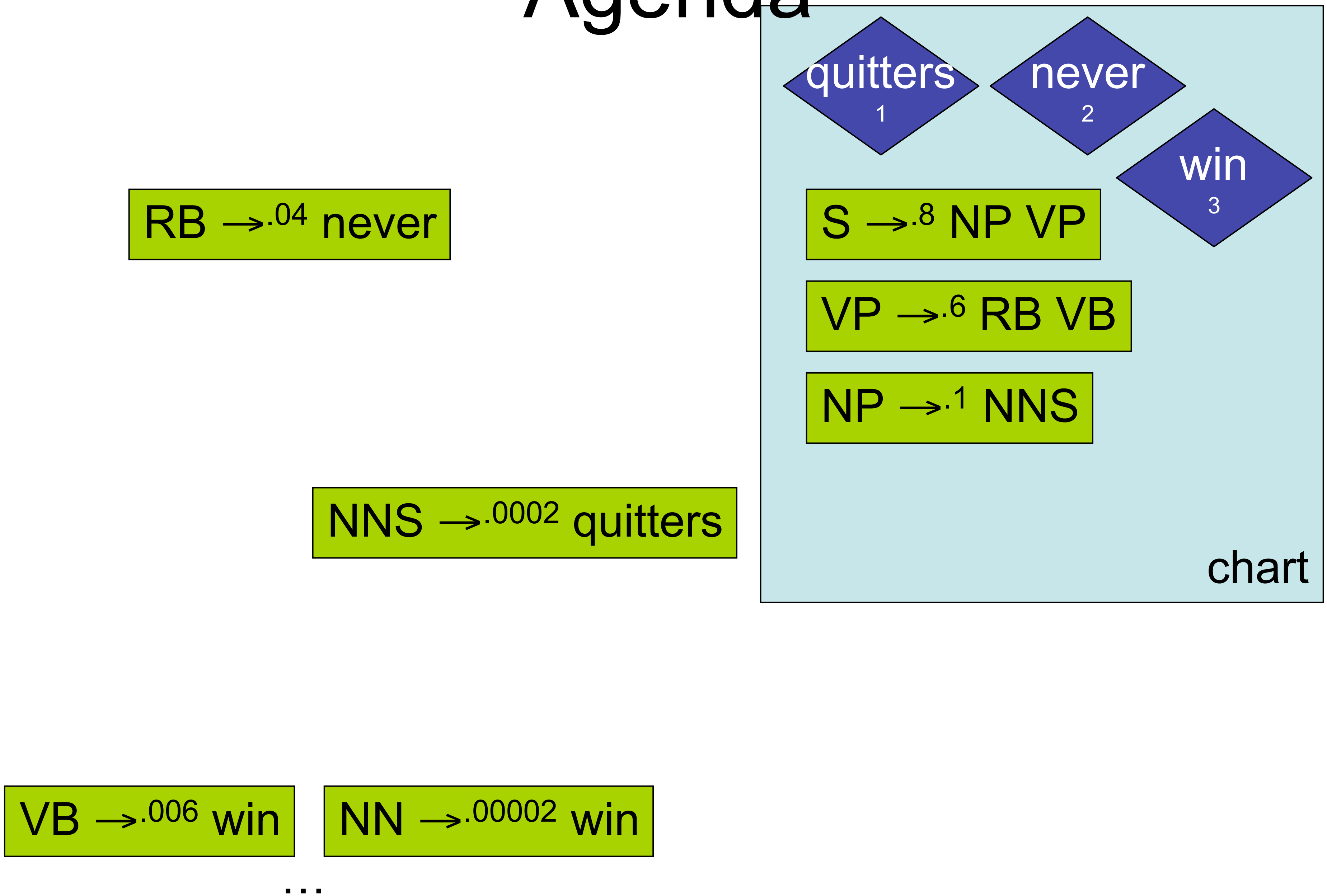
RB $\rightarrow^{.04}$ never

VB $\rightarrow^{.006}$ win

NN $\rightarrow^{.00002}$ win

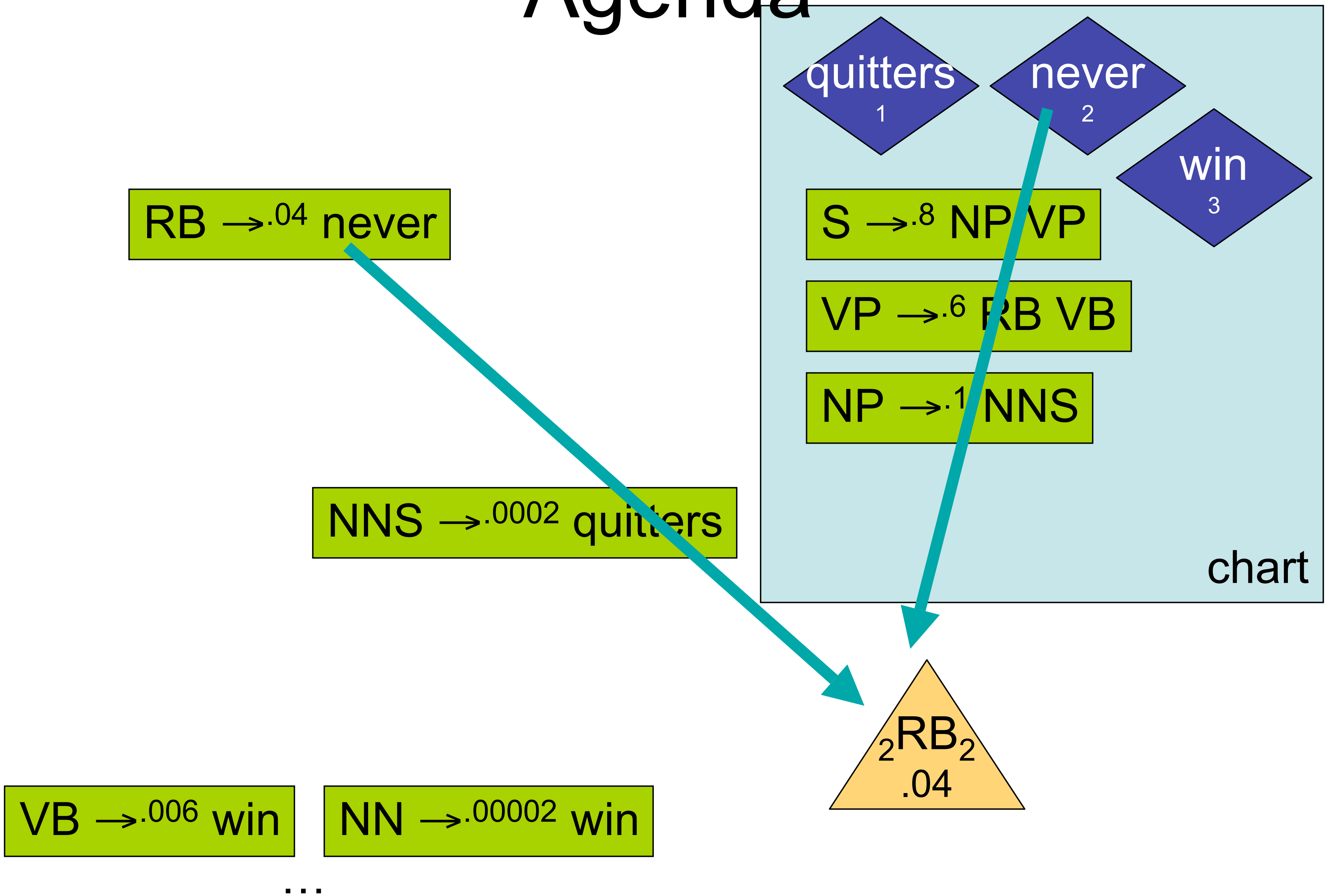
…

Agenda

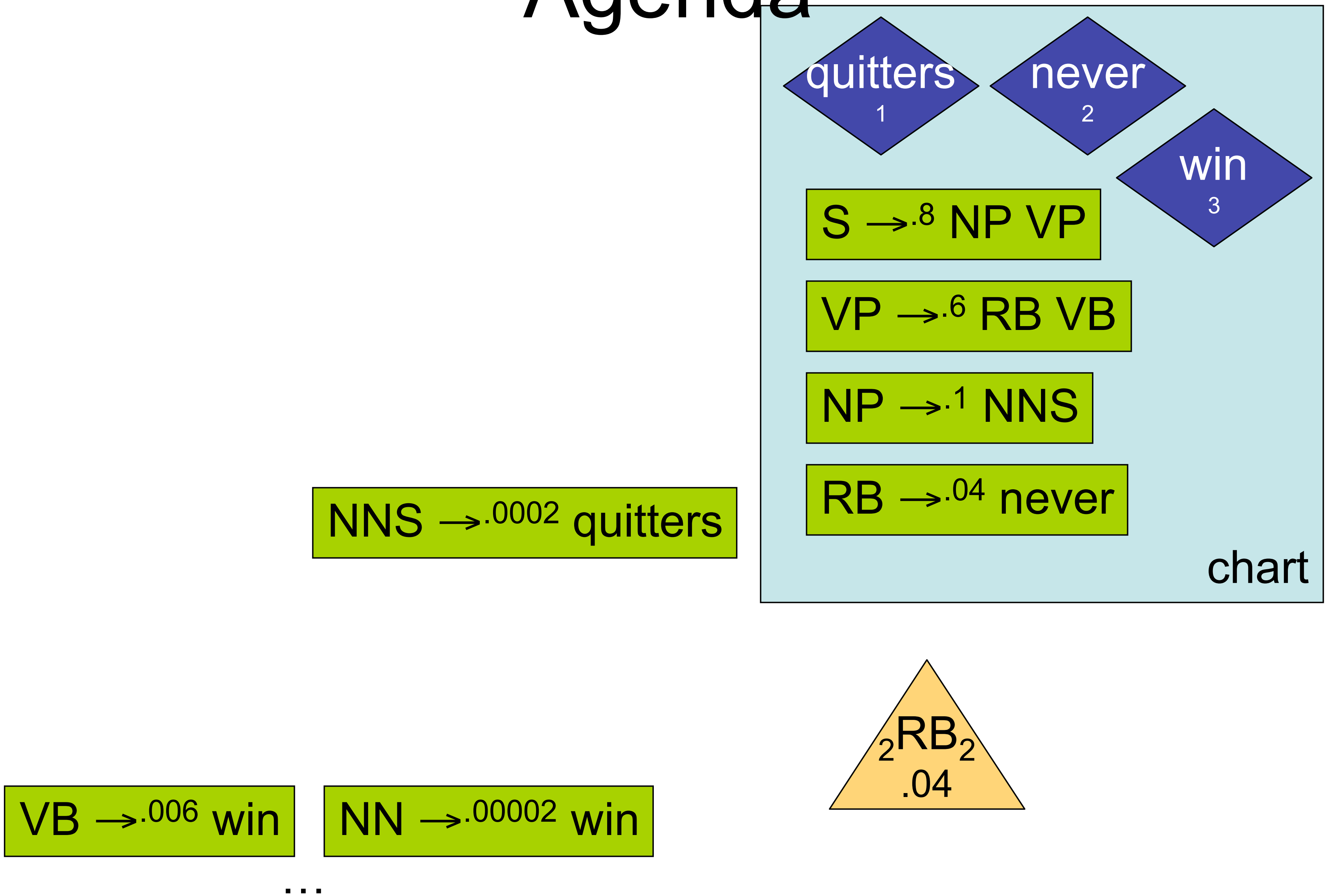


quitters 1
never 2
win 3
$S \rightarrow^{.8}$ NP VP
$VP \rightarrow^{.6}$ RB VB
$NP \rightarrow^{.1}$ NNS
chart
$RB \rightarrow^{.04}$ never
$NNS \rightarrow^{.0002}$ quitters
$VB \rightarrow^{.006}$ win
$NN \rightarrow^{.00002}$ win
…

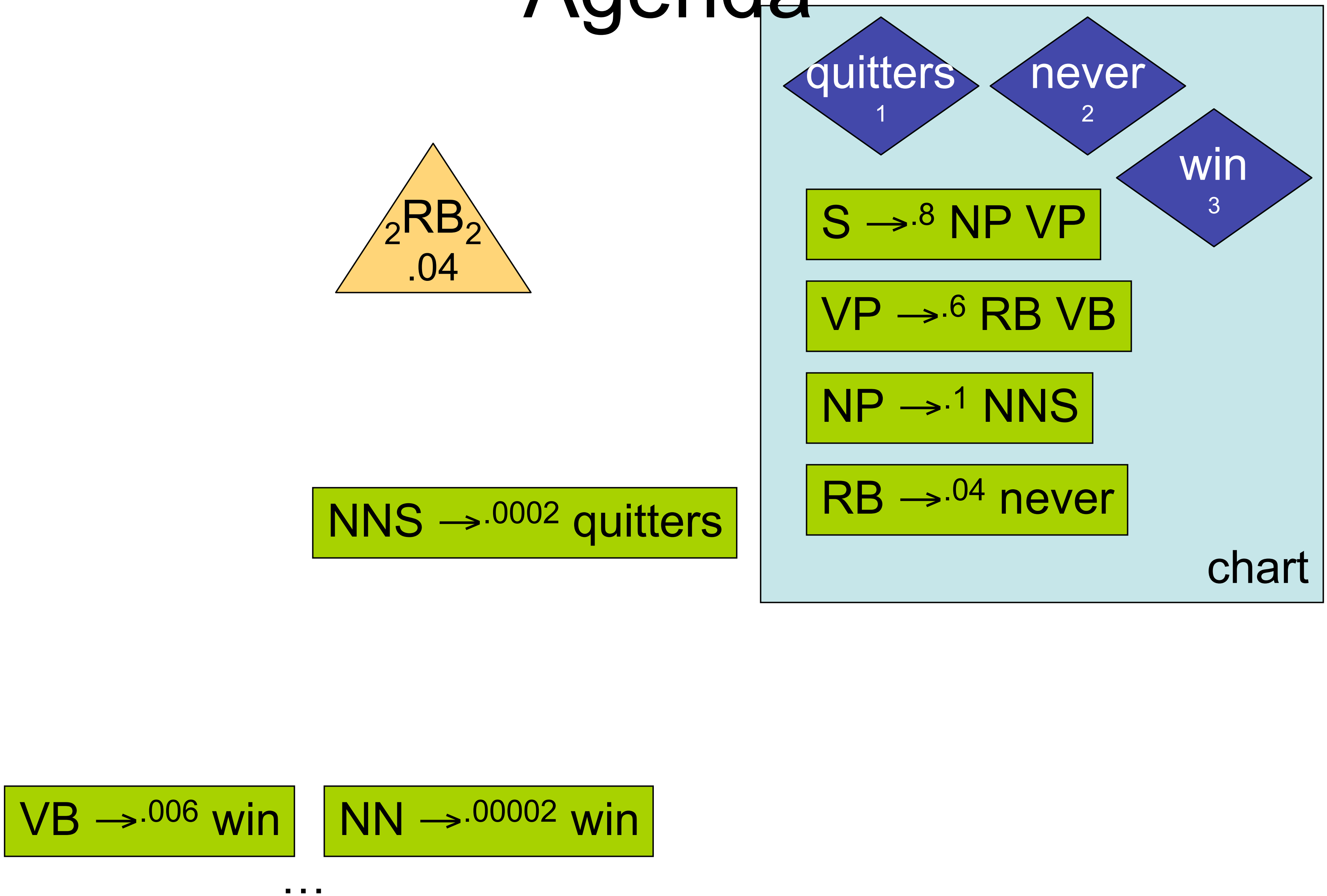
Agenda


quitters
1
never
2
win
3
S →.8 NP VP
VP →.6 RB VB
NP →.1 NNS
chart
RB →.04 never
NNS →.0002 quitters
2RB2
.04
VB →.006 win
NN →.00002 win
…

Agenda

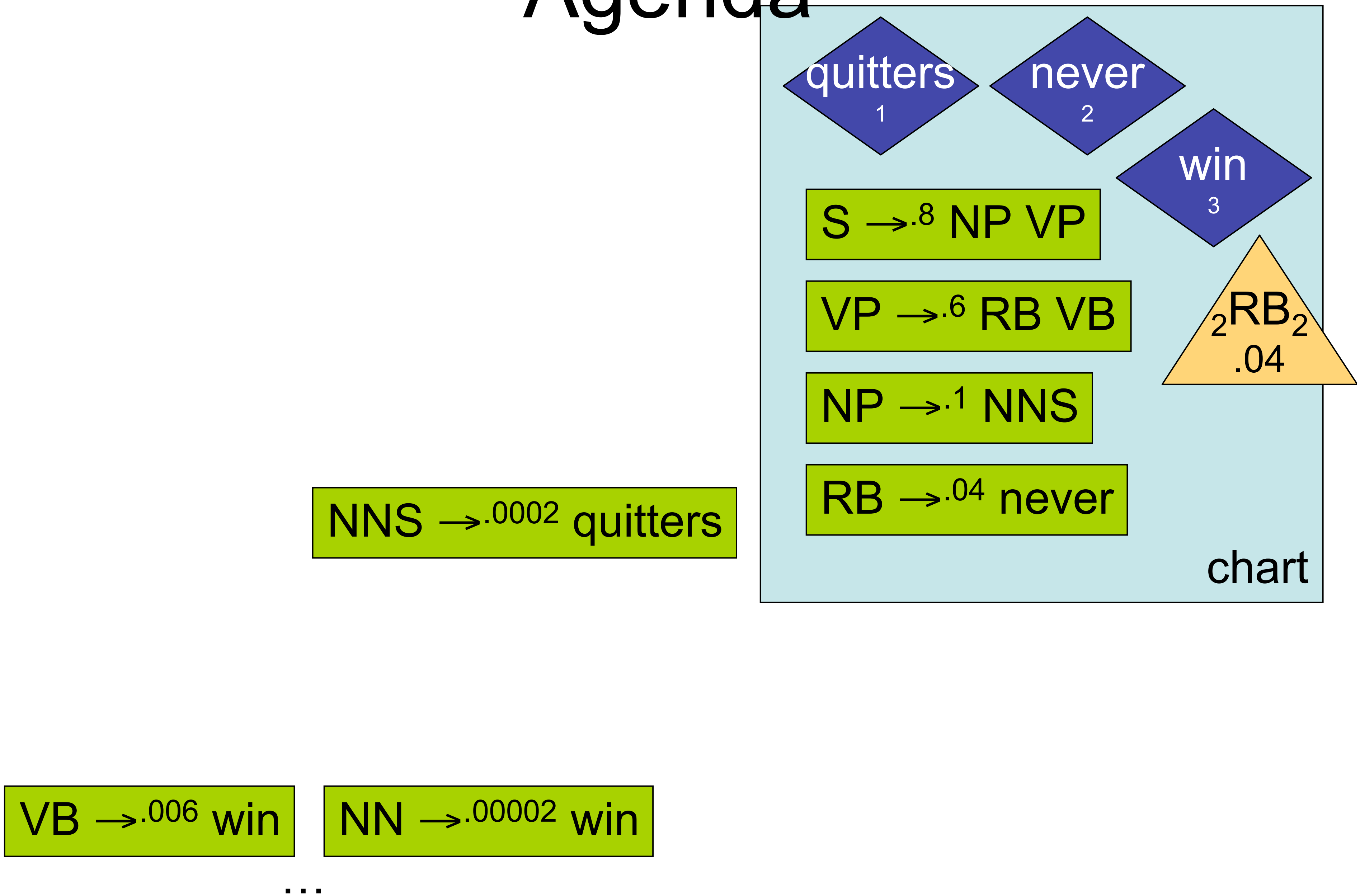

quitters
1
never
2
win
3
$S \rightarrow^{.8} NP\ VP$
$VP \rightarrow^{.6} RB\ VB$
$NP \rightarrow^{.1} NNS$
$RB \rightarrow^{.04}$ never
chart
$NNS \rightarrow^{.0002}$ quitters
$_2RB_2$
.04
$VB \rightarrow^{.006}$ win
$NN \rightarrow^{.00002}$ win
…

Agenda

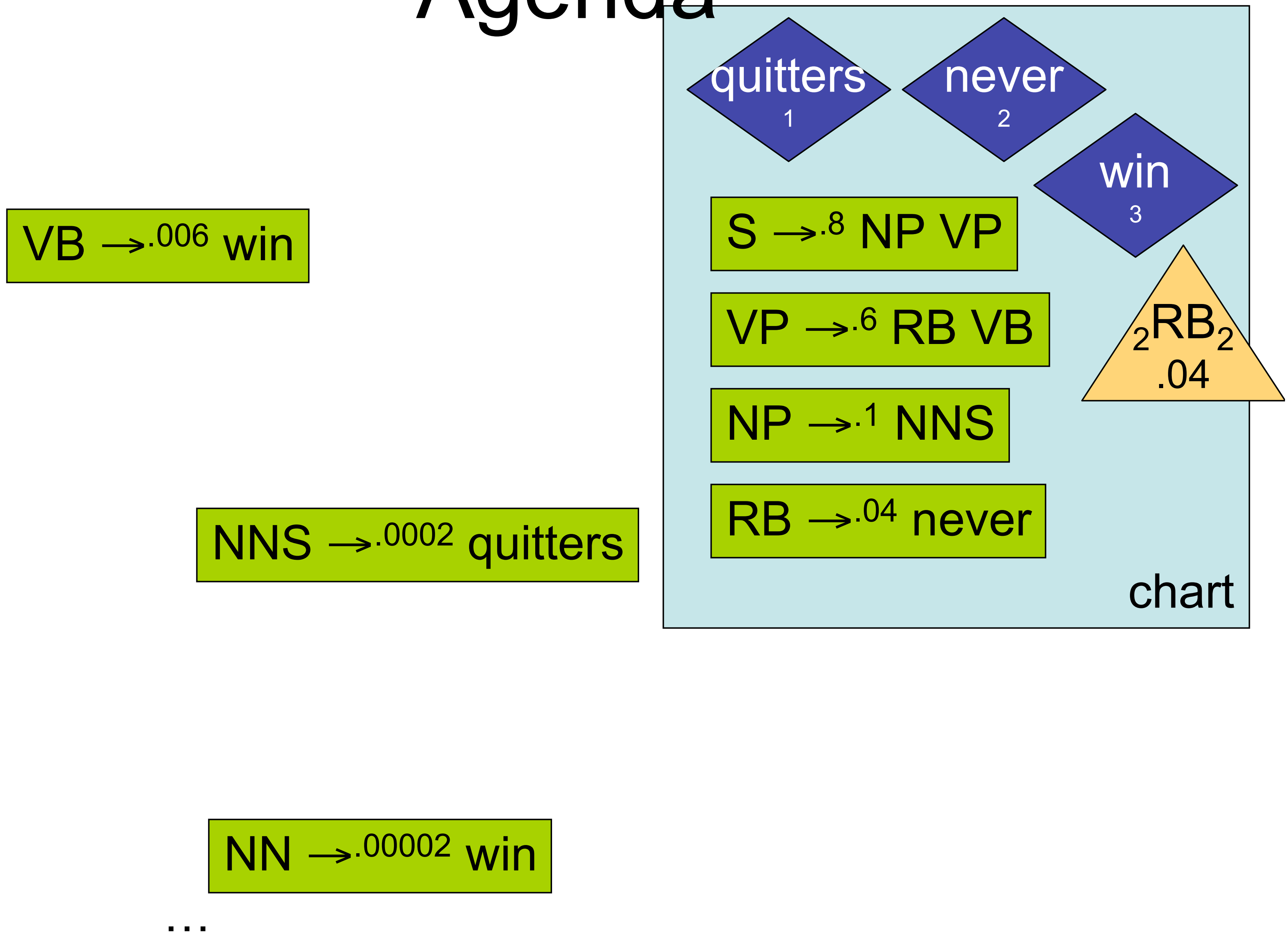


quitters
1
never
2
win
3
S $\to^{.8}$ NP VP
VP $\to^{.6}$ RB VB
NP $\to^{.1}$ NNS
RB $\to^{.04}$ never
chart
$_{2}RB_{2}$
.04
NNS $\to^{.0002}$ quitters
VB $\to^{.006}$ win
NN $\to^{.00002}$ win
…

Agenda

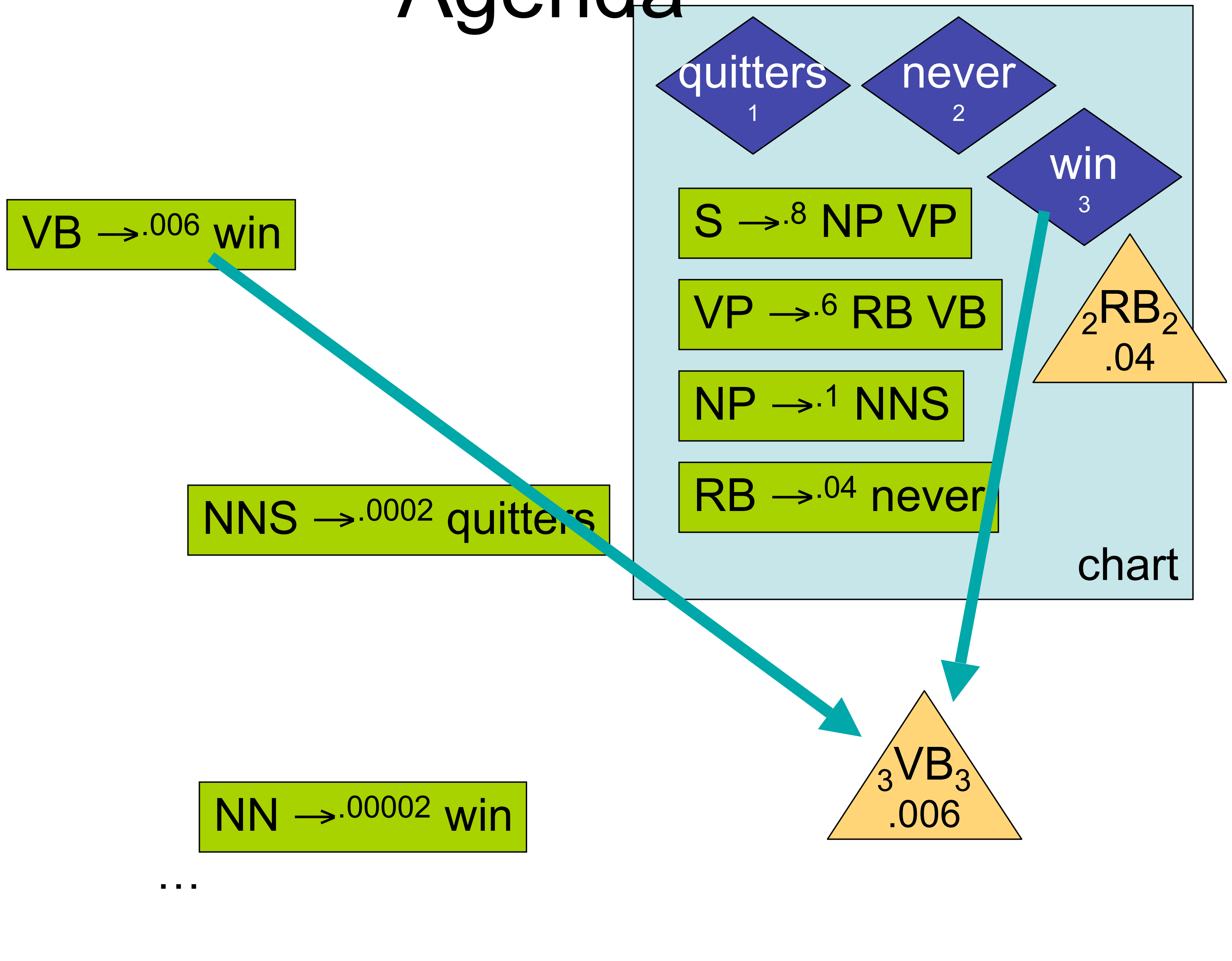

quitters 1
never 2
win 3
S →.8 NP VP
VP →.6 RB VB
NP →.1 NNS
RB →.04 never
2RB2 .04
chart
NNS →.0002 quitters
VB →.006 win
NN →.00002 win
…

Agenda



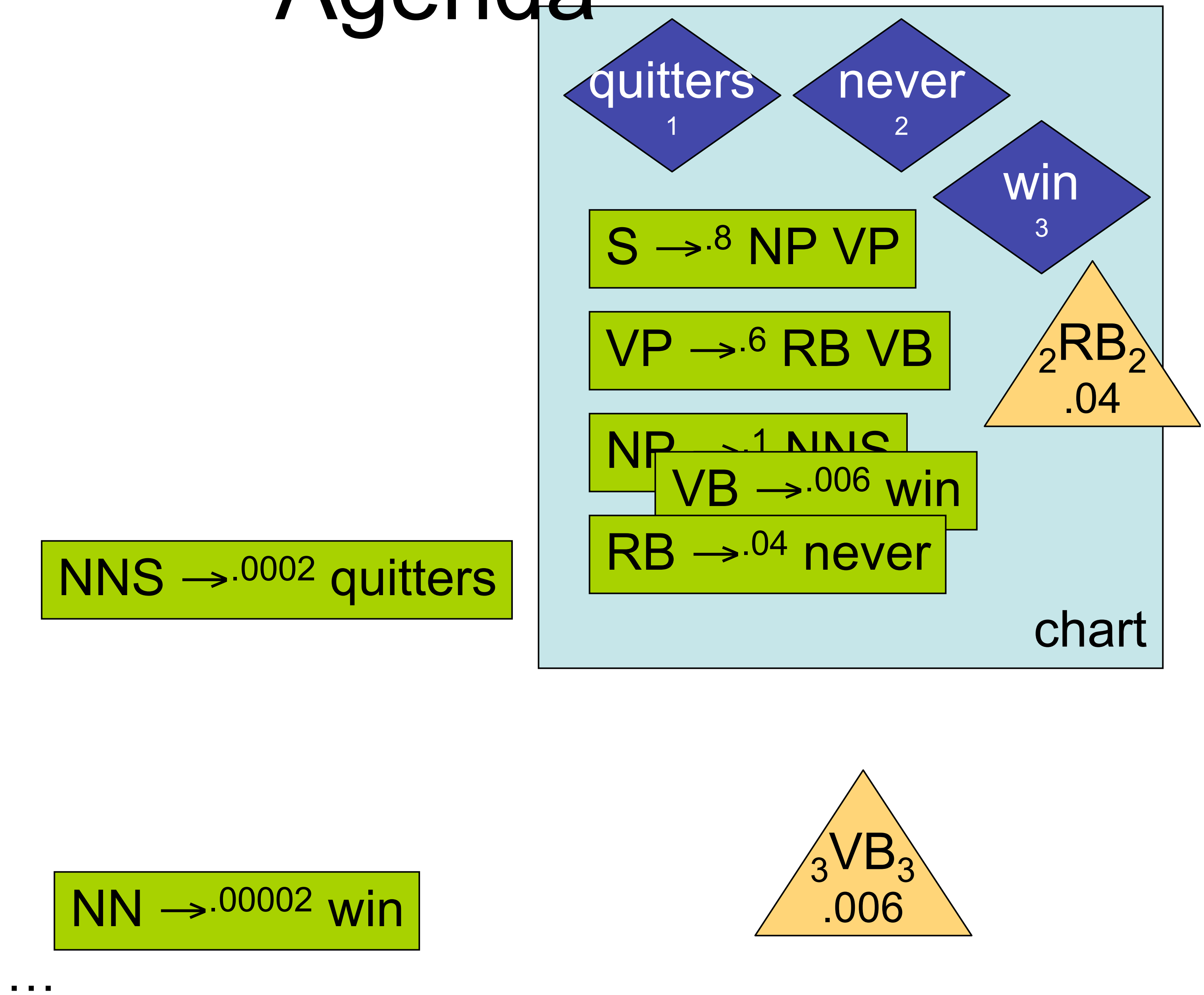
VB $\rightarrow^{.006}$ win
NNS $\rightarrow^{.0002}$ quitters
NN $\rightarrow^{.00002}$ win
…
quitters 1
never 2
win 3
S $\rightarrow^{.8}$ NP VP
VP $\rightarrow^{.6}$ RB VB
NP $\rightarrow^{.1}$ NNS
RB $\rightarrow^{.04}$ never
$_2RB_2$ .04
chart

Agenda



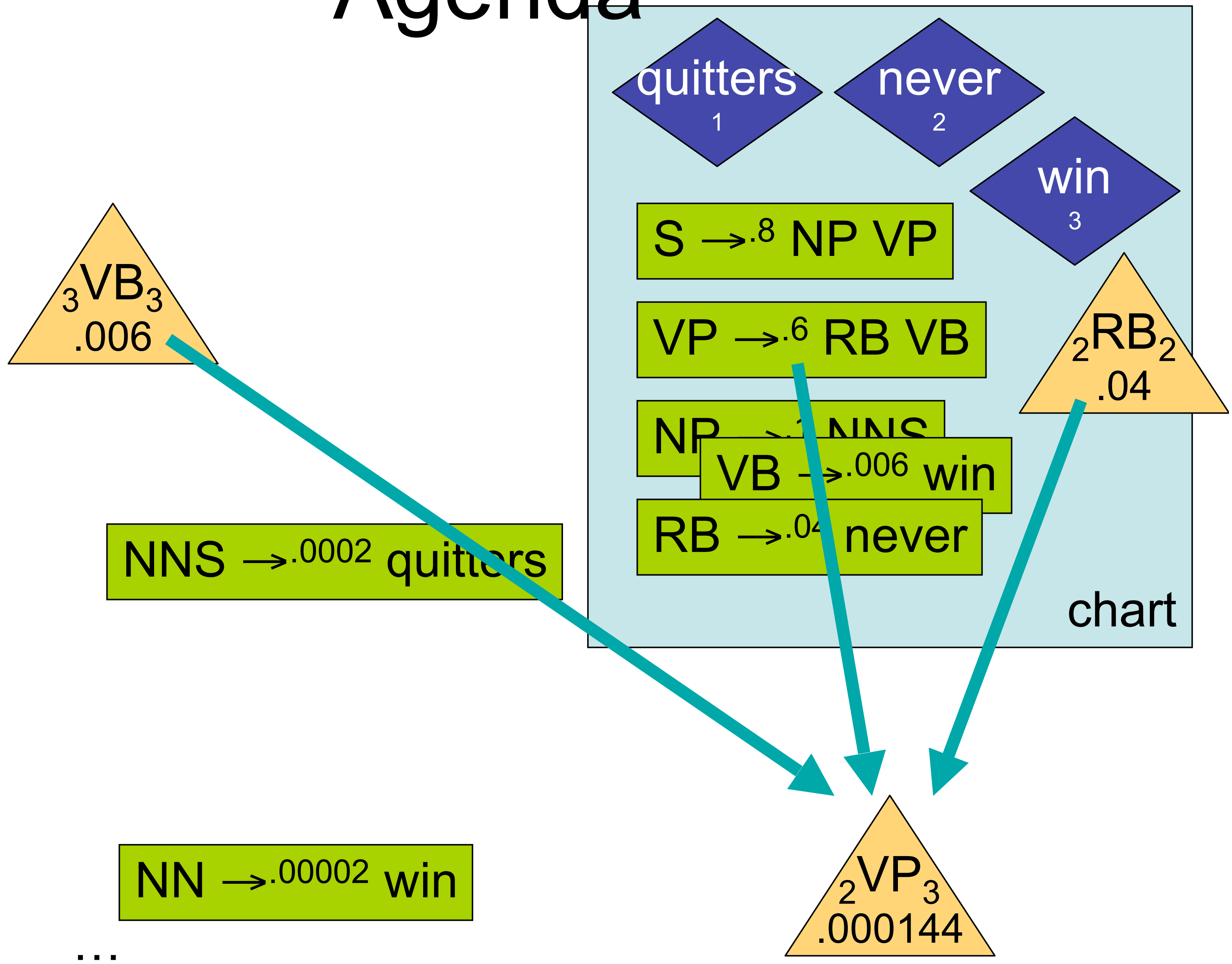
VB →.006 win
NNS →.0002 quitters
NN →.00002 win
…
quitters 1
never 2
win 3
S →.8 NP VP
VP →.6 RB VB
NP →.1 NNS
RB →.04 never
2RB2 .04
chart
3VB3 .006

Agenda



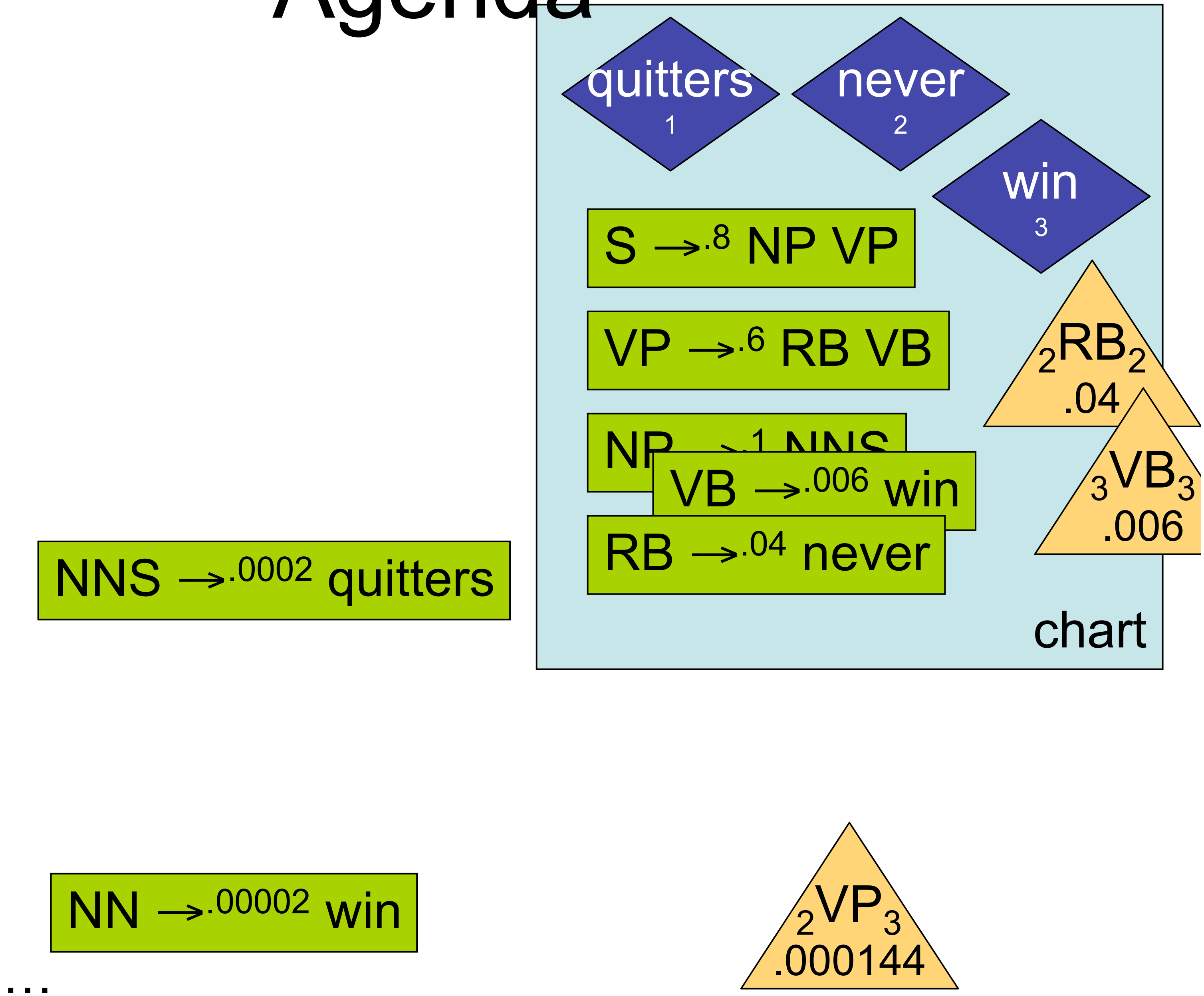
quitters
1
never
2
win
3
S →.8 NP VP
VP →.6 RB VB
VB →.006 win
RB →.04 never
2RB2
.04
chart
NNS →.0002 quitters
3VB3
.006
NN →.00002 win
…

Agenda

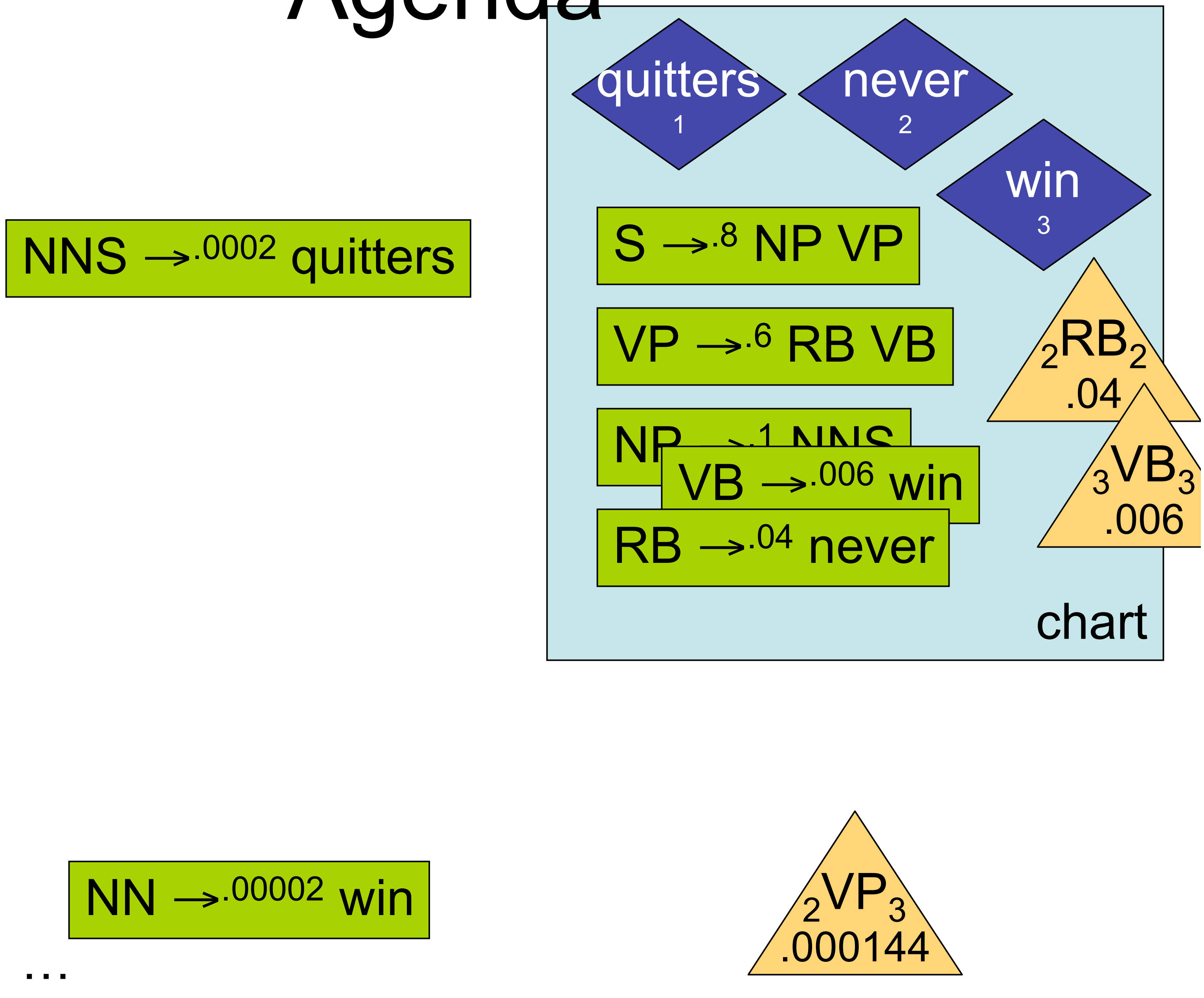


quitters
1
never
2
win
3
S →.8 NP VP
VP →.6 RB VB
VB →.006 win
RB →.04 never
chart
3VB3
.006
2RB2
.04
NNS →.0002 quitters
NN →.00002 win
…
2VP3
.000144

Agenda

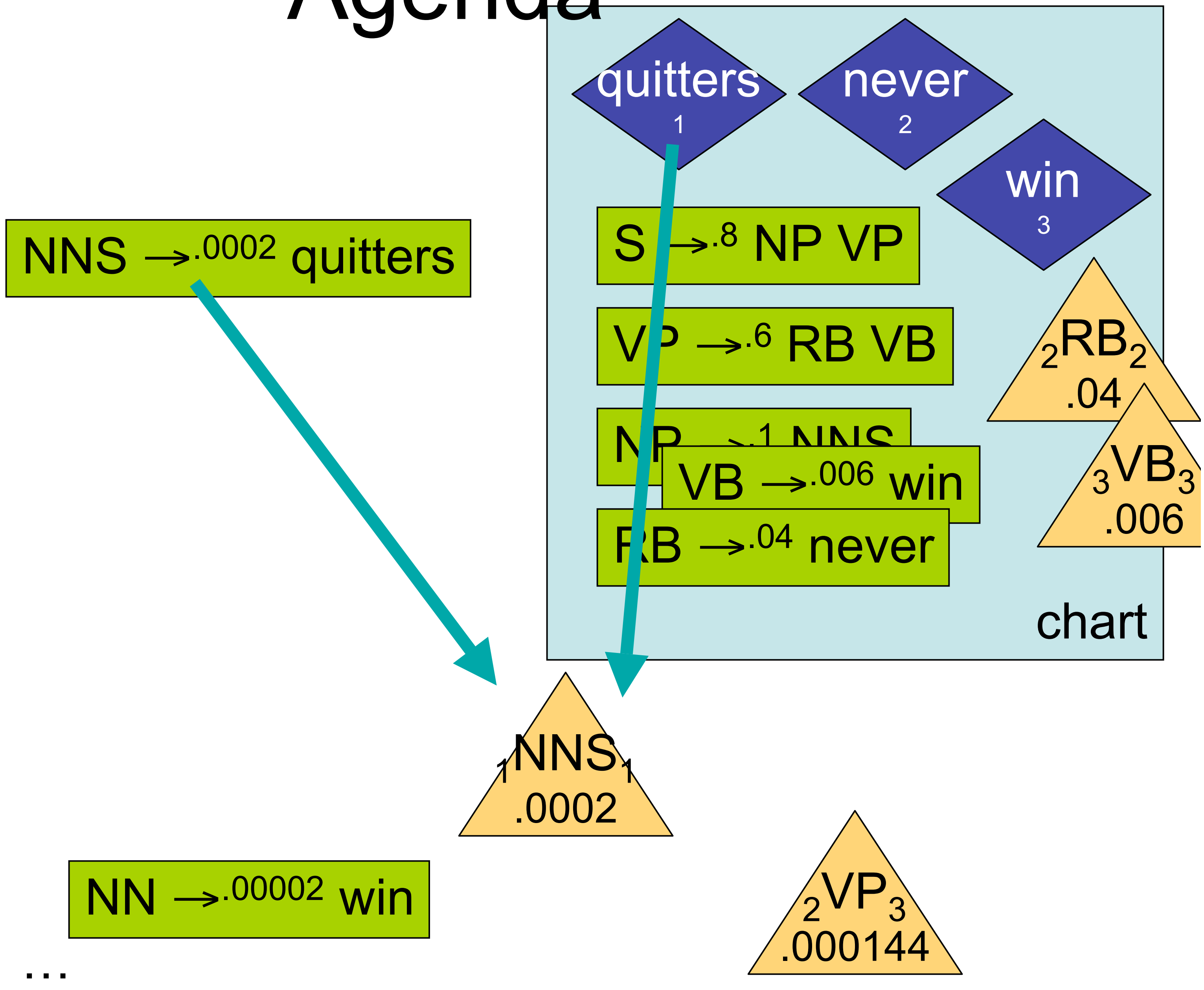

quitters
1
never
2
win
3
S →.8 NP VP
VP →.6 RB VB
NP
VB →.006 win
RB →.04 never
2RB2
.04
3VB3
.006
chart
NNS →.0002 quitters
NN →.00002 win
…
2VP3
.000144

Agenda



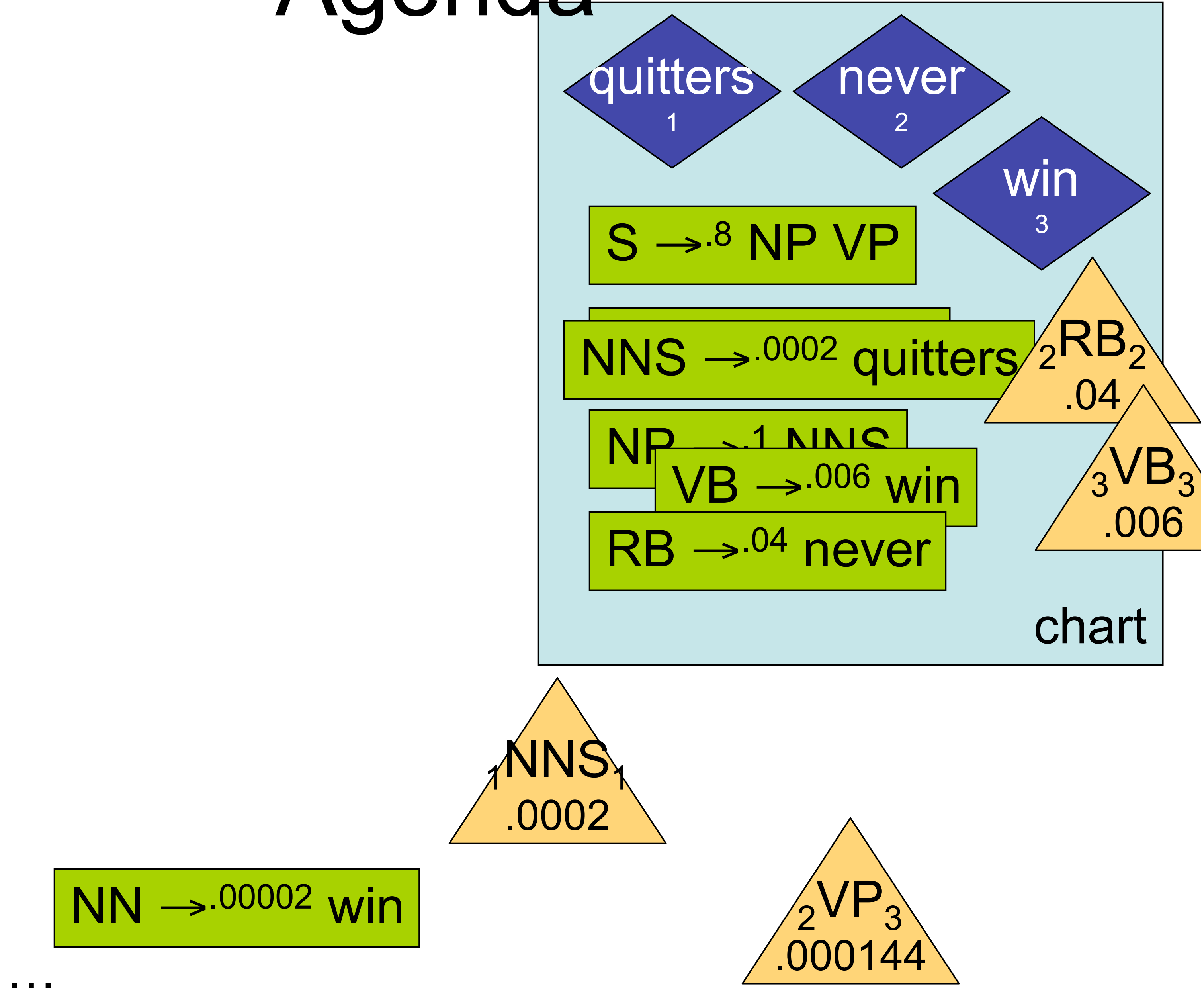
NNS →.0002 quitters
NN →.00002 win
…
quitters 1
never 2
win 3
S →.8 NP VP
VP →.6 RB VB
NP
VB →.006 win
RB →.04 never
2RB2 .04
3VB3 .006
chart
2VP3 .000144

Agenda

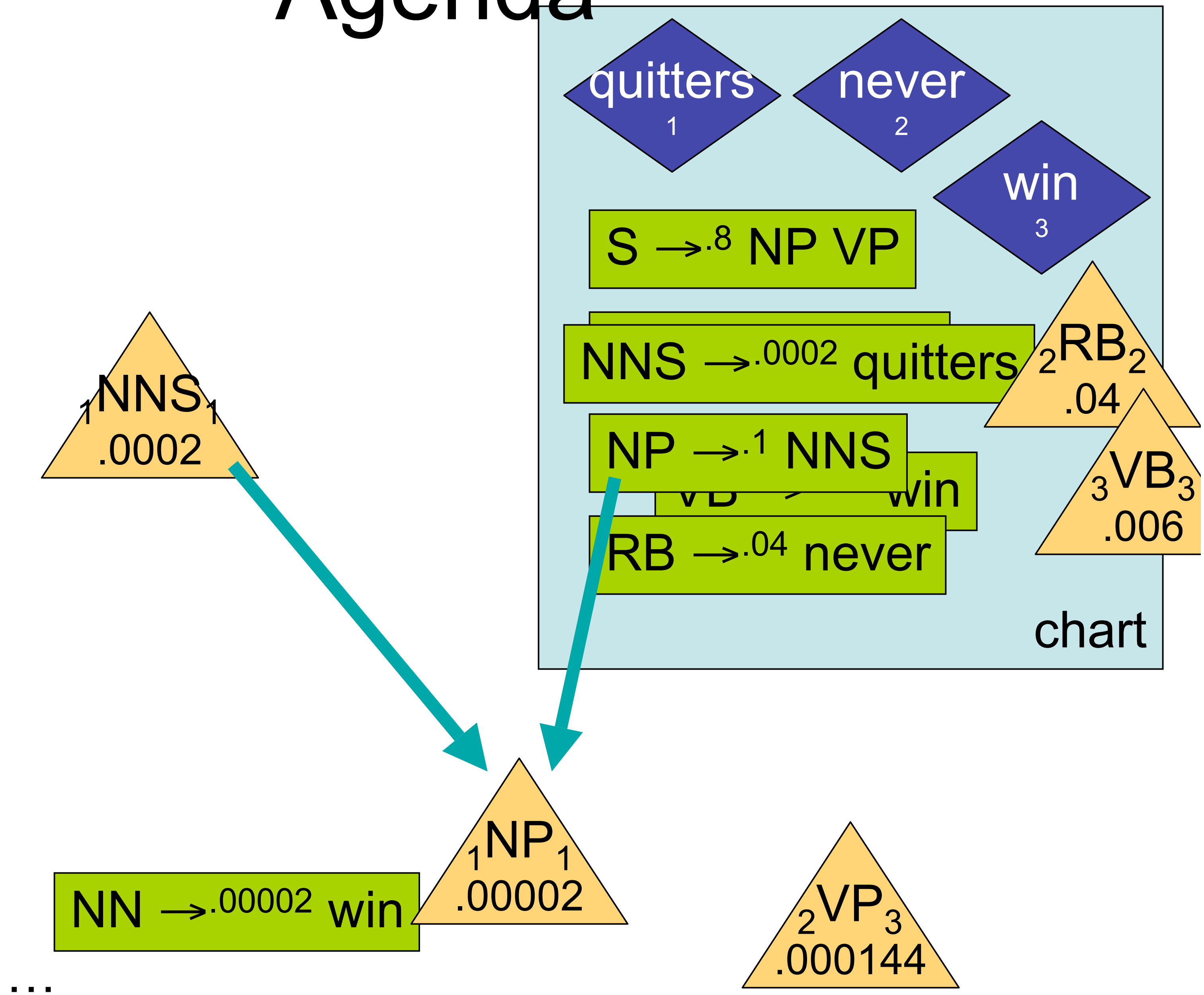


NNS →.0002 quitters
quitters 1
never 2
win 3
S →.8 NP VP
VP →.6 RB VB
NP →.1 NNS
VB →.006 win
RB →.04 never
2RB2 .04
3VB3 .006
chart
1NNS1 .0002
NN →.00002 win
2VP3 .000144
…

Agenda

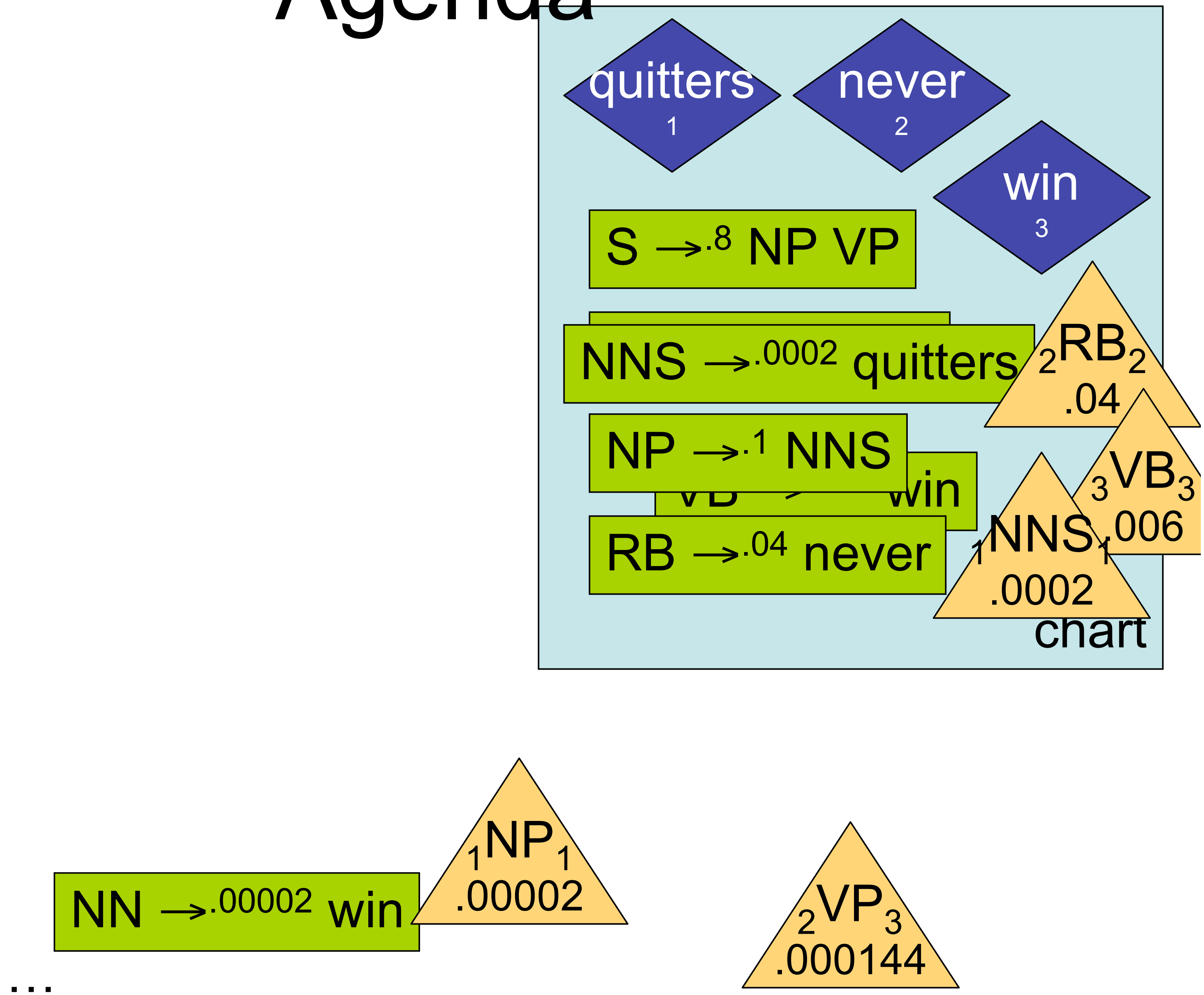

quitters
1
never
2
win
3
S →.8 NP VP
NNS →.0002 quitters
VB →.006 win
RB →.04 never
2RB2
.04
3VB3
.006
chart
1NNS1
.0002
NN →.00002 win
…
2VP3
.000144

Agenda



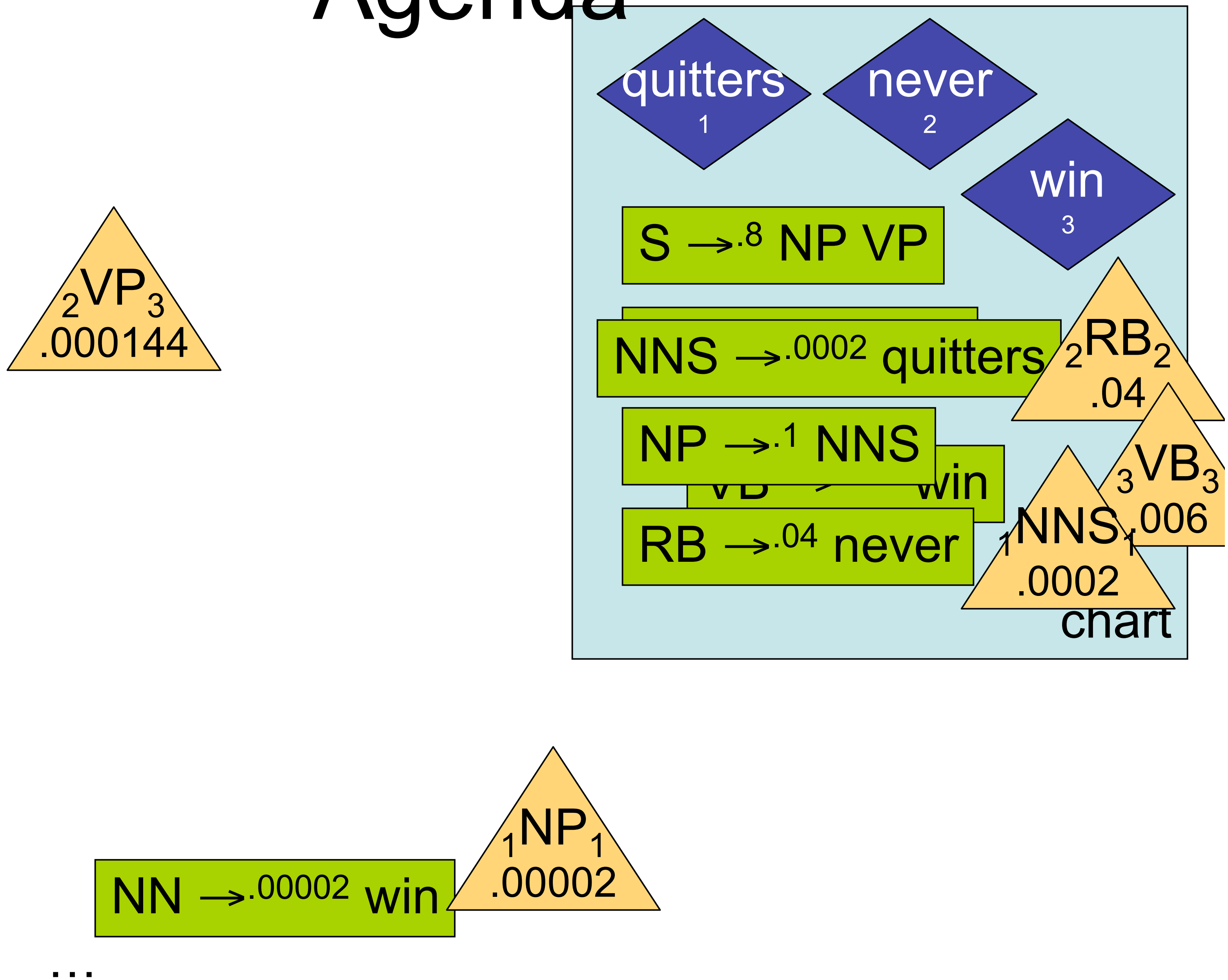
quitters
1
never
2
win
3
S →.8 NP VP
NNS →.0002 quitters
NP →.1 NNS
win
RB →.04 never
2RB2
.04
3VB3
.006
chart
1NNS1
.0002
1NP1
.00002
NN →.00002 win
2VP3
.000144
…

Agenda



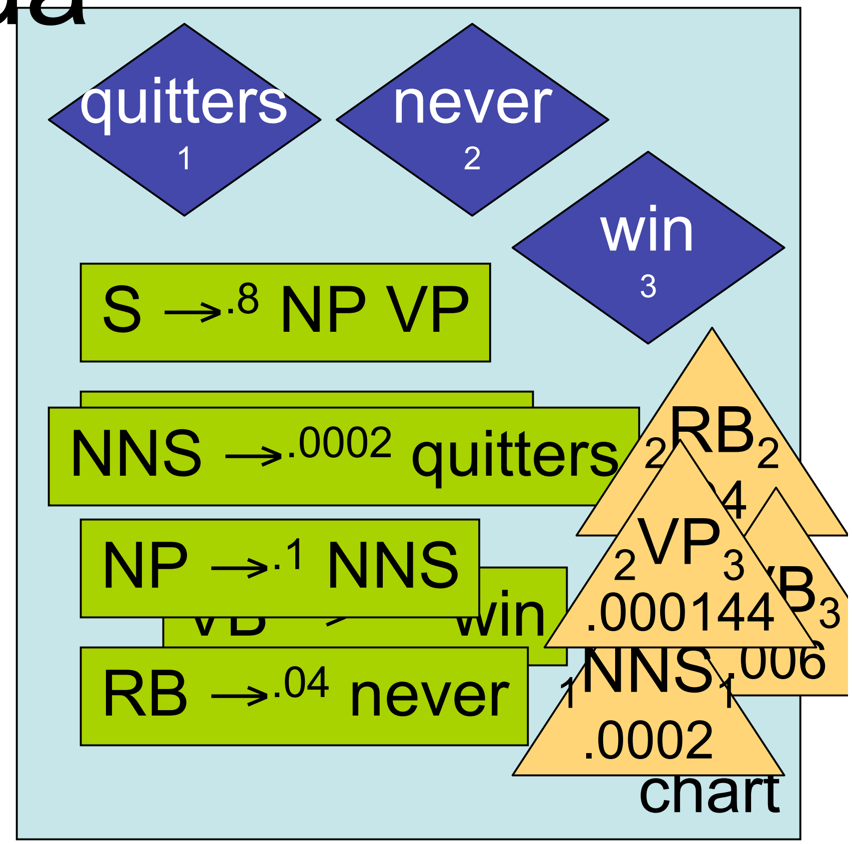
quitters
1
never
2
win
3
S →.8 NP VP
NNS →.0002 quitters
NP →.1 NNS
RB →.04 never
2RB2
.04
3VB3
.006
1NNS1
.0002
chart
NN →.00002 win
1NP1
.00002
2VP3
.000144
…

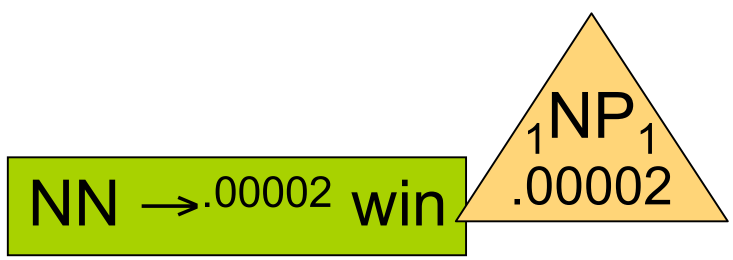
Agenda



$_2VP_3$
.000144
quitters
1
never
2
win
3
S →.8 NP VP
NNS →.0002 quitters
NP →.1 NNS
RB →.04 never
$_2RB_2$
.04
$_3VB_3$
.006
$_1NNS_1$
.0002
chart
NN →.00002 win
$_1NP_1$
.00002
…

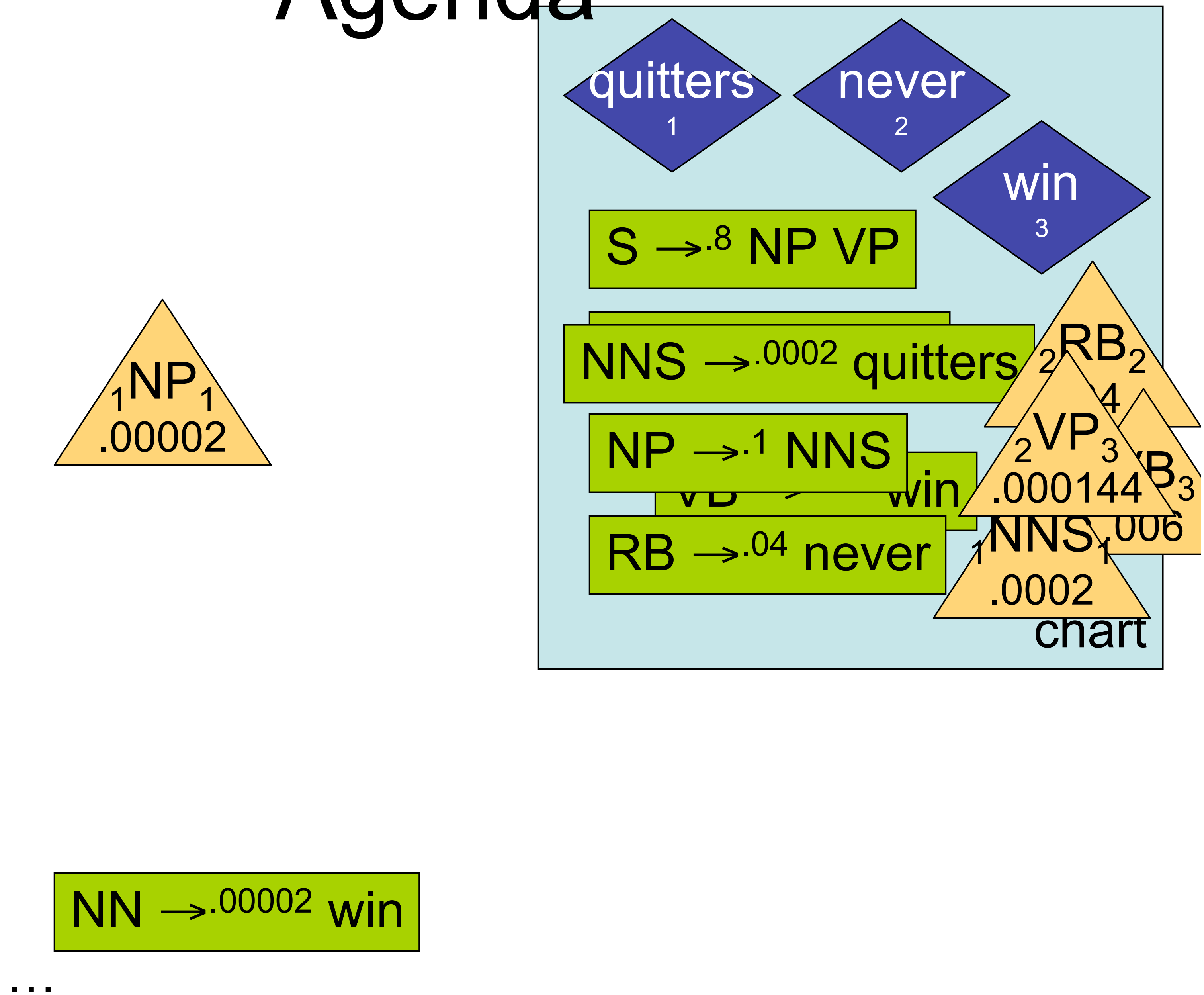
Agenda


quitters
1
never
2
win
3
S →.8 NP VP
NNS →.0002 quitters
NP →.1 NNS
win
RB →.04 never
2RB2
2VP3
.000144
.006
1NNS1
.0002
chart

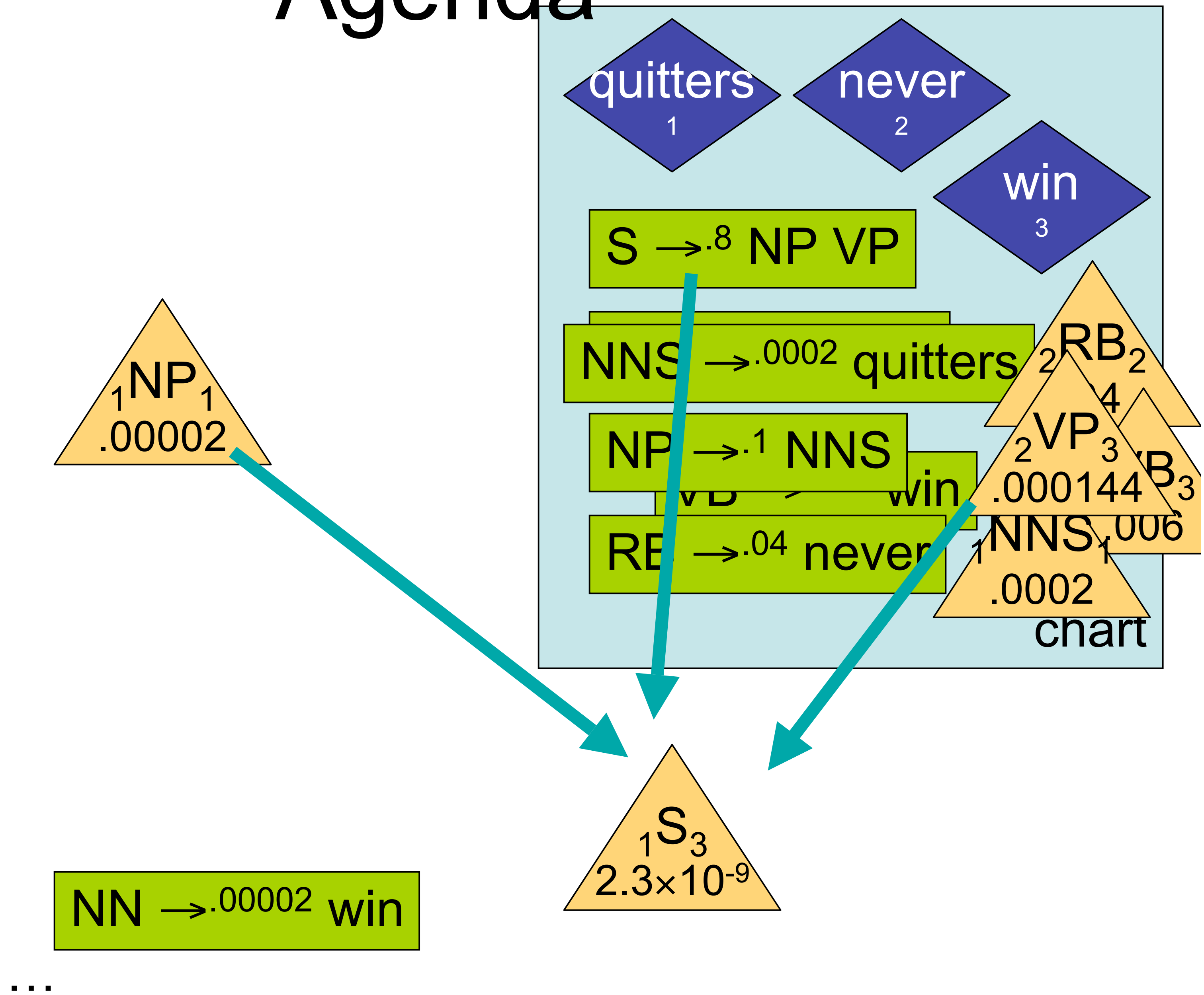

NN →.00002 win
1NP1
.00002

…

Agenda


quitters 1
never 2
win 3
S →.8 NP VP
NNS →.0002 quitters
NP →.1 NNS
RB →.04 never
2RB2
2VP3 .000144
1NNS1 .0002
.006
chart
1NP1 .00002
NN →.00002 win
…

Agenda



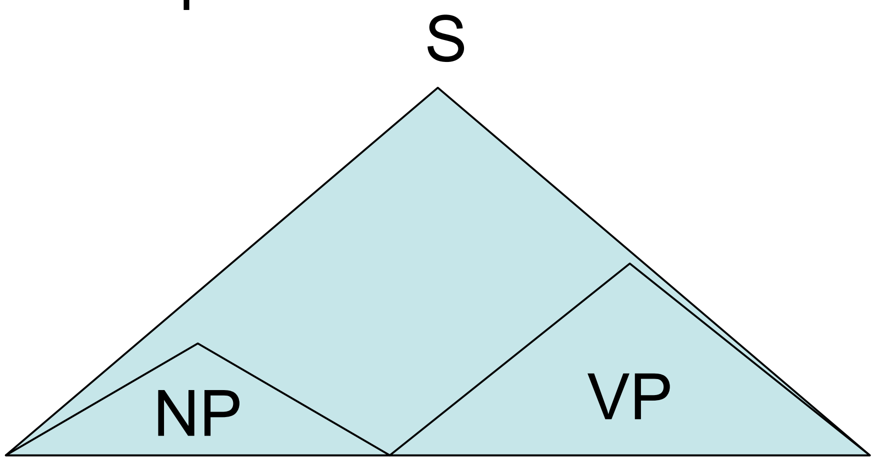
quitters 1
never 2
win 3
S →.8 NP VP
NNS →.0002 quitters
NP →.1 NNS
RB →.04 never
2RB2
2VP3
.000144
1NNS1
.0002
.006
chart
1NP1
.00002
1S3
2.3×10-9
NN →.00002 win
…

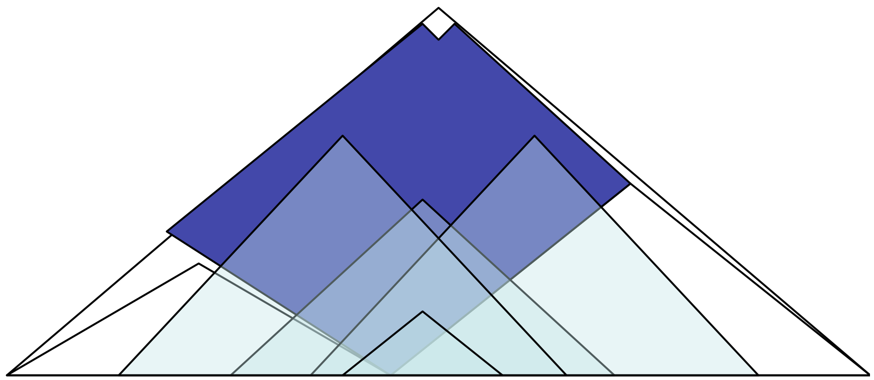
Unnecessary Work

- If you only want the best derivation, you don’t want to build items that aren’t in it!
- But you don’t know which items to build until you have the best parse.
- Key idea in the agenda:
  - Intelligently order **updates** to items’ weights.
  - Roughly analogous to trading depth and breadth in **search**.
- Note: for exact inside/outside, all of the work **is** necessary!

Unnecessary Work

best parse:

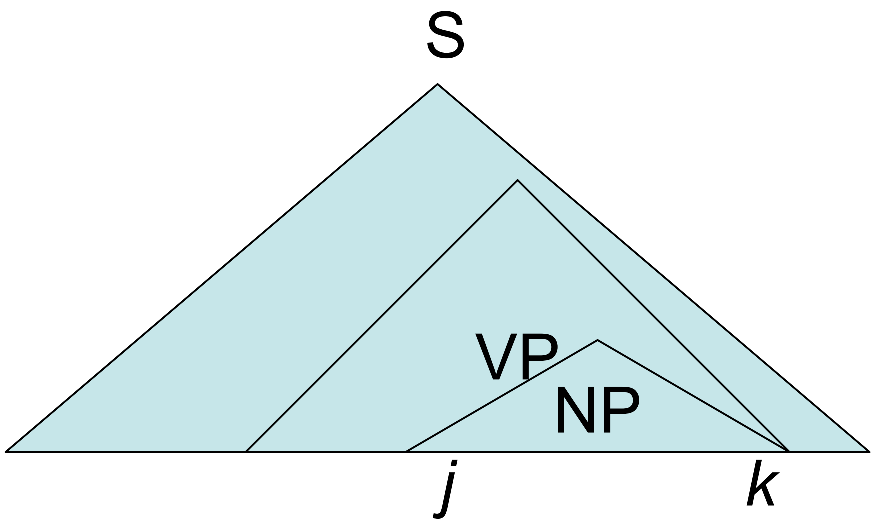


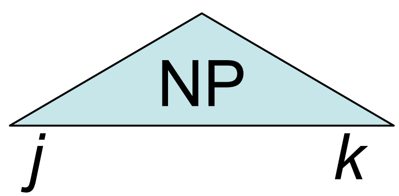


Repropagation

- Suppose (NP, 4, 7) currently has a weight of 0.3, constructed by (DT, 4, 5) $\otimes$ (NP, 5, 7).
- Now suppose we find that **a better** way to build (NP, 4, 7): (DT, 4, 5) $\otimes$ (NNP, 5, 6) $\otimes$ (NNP, 6, 7) with value 0.31.
- Maybe now we have a better way to build (VP, 3, 7)! (Or anything else that used (NP, 4, 7).
- Have to re-build all of those consequents, and compare again, and recursively repropagate to consequents of any item whose value changes.
- May not be $O(n^3)$ anymore!

Repropagation





’s value increased!

So any **consequent** of (NP, *j*, *k*) might also increase.

Best-First Parsing

- **Viterbi** semiring (find the best parse): see Nederhof (2003) and Knuth (1977).
  - Cf. Goodman, build the chart and fill in weights at the same time.
- Many parsers in practice: prune, prune, prune.
- Alternative: order items by their weights.
  - "**Uniform cost search**"
  - Guarantee: the first time **goal** is popped from the agenda, you have the optimal parse.
- Charniak et al., 1998: heuristics to speed this up. "Figures of Merit" (big speed payoff).

Priorities

$$\text{priority}(I) = \text{ weight of the best parse that uses } I$$

$$= \text{inside}(I) \otimes \text{outside}(I)$$

$$\leq \text{inside}(I) \otimes \underbrace{\text{estimate}(\text{outside}(I))}_{\text{uniform cost search: } \mathbf{1}}$$

•Klein and Manning (2003): Blocked more than 90% of edges!
•Generalization of A* search (for hypergraphs instead of graphs).
•Heuristics? Computed by simpler, cheaper dynamic programs!
•Caveat emptor: only for Viterbi (max) semirings!

Dyna (Eisner et al., 2005)

- **Dyna** is a high-level programming language (like Prolog) for weighted deduction.
- Source code looks like Prolog.
- Compiles into C++.
- Core algorithms:
  - Generalized weighted, **prioritized** agenda.
    - Allows the use of heuristics, including A*
    - Handles repropagation if required
  - Efficient "tape" mechanism for **reverse** computation.
    - Very similar to backpropagation.

Dyna Programs

```
constit(X,I,J)     +=  word(W,I,J) * rewrite(X,W).
constit(X,I,J)     +=  constit(Y,I,Mid) * constit(Z,Mid,J) * rewrite(X,Y,Z).
goal               +=  constit("s",0,N) whenever length(N).
```

```
constit(X,I,J)     max=  word(W,I,J) * rewrite(X,W).
constit(X,I,J)     max=  constit(Y,I,Mid) * constit(Z,Mid,J) * rewrite(X,Y,Z).
goal               max=  constit("s",0,N) whenever length(N).
```

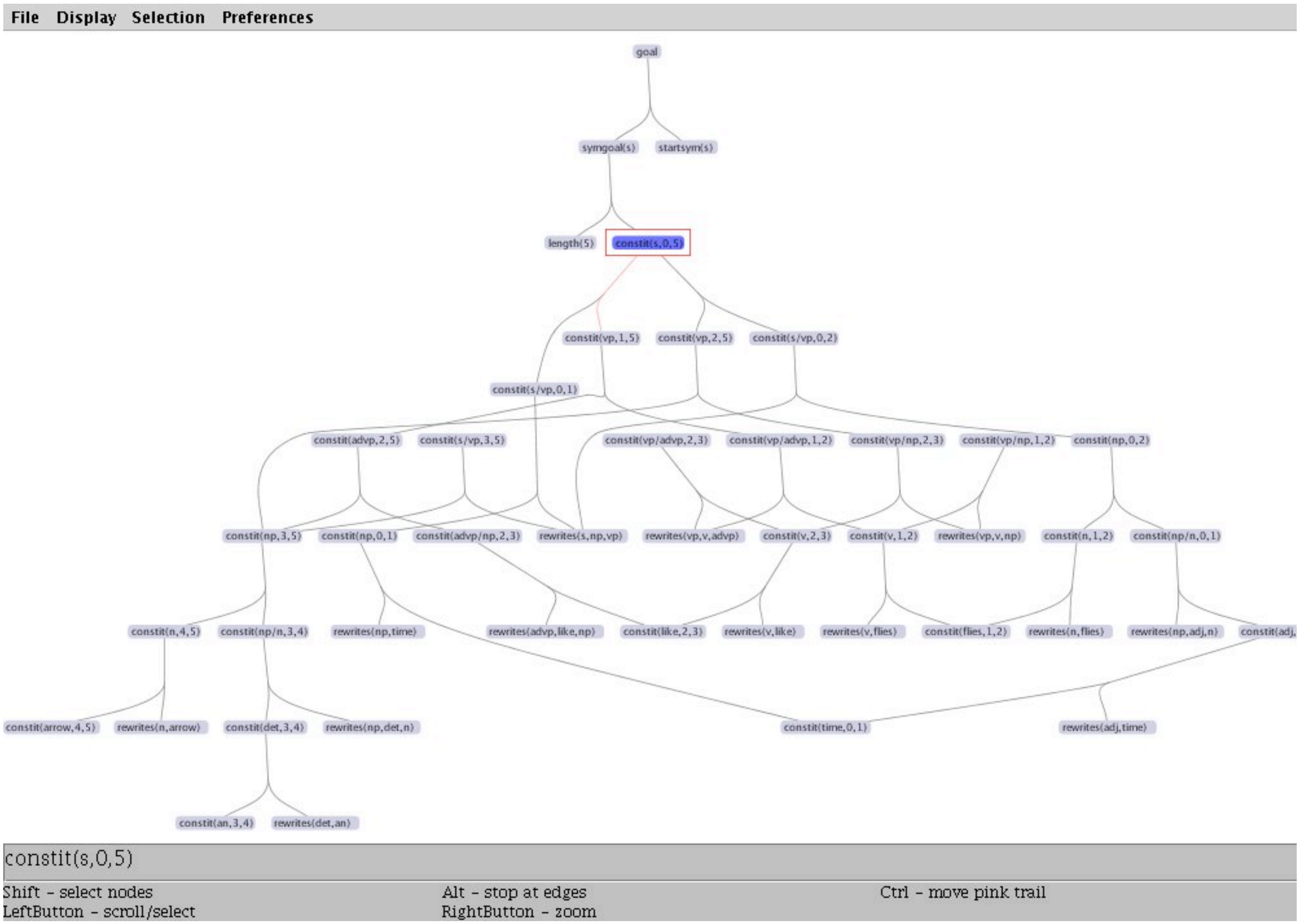
```
constit(X,I,J)     max=  word(W,I,J) * rewrite(X,W).
constit(X,I,J)     max=  constit(Y,I,Mid) * inter(X, Z, Mid, J).
inter(X, Z, Mid, J) max= constit(Z,Mid,J) * rewrite(X,Y,Z).
goal               max=  constit("s",0,N) whenever length(N).
```

Dyna Programs

```
constit(X,I,J)     += word(W,I,J) * rewrite(X,W).
constit(X,I,J)     += constit(Y,I,Mid) * constit(Z,Mid,J) * rewrite(X,Y,Z).
goal               += constit("s",0,N) whenever length(N).
```

```
reverseconstit(Y,I,Mid)  += reverseconstit(X,I,J) * constit(Z,Mid,J) * rewrite(X,Y,Z).
reverseconstit(Z,Mid,J) += constit(Y,I,Mid) * reverseconstit(X,I,J) * rewrite(X,Y,Z).
reverseconstit("s",0,N)  += 1.
```

Dyna Debugger



Parting Shots

- Weighted deduction as a convenient way to

design,

improve,

understand,

analyze,

unify,

transform,

and implement

otherwise tricky dynamic programming algorithms.