Collapse Operation Increases Expressive Power of Deterministic Higher Order Pushdown Automata

Paweł Parys

University of Warsaw

Two hierarchies:

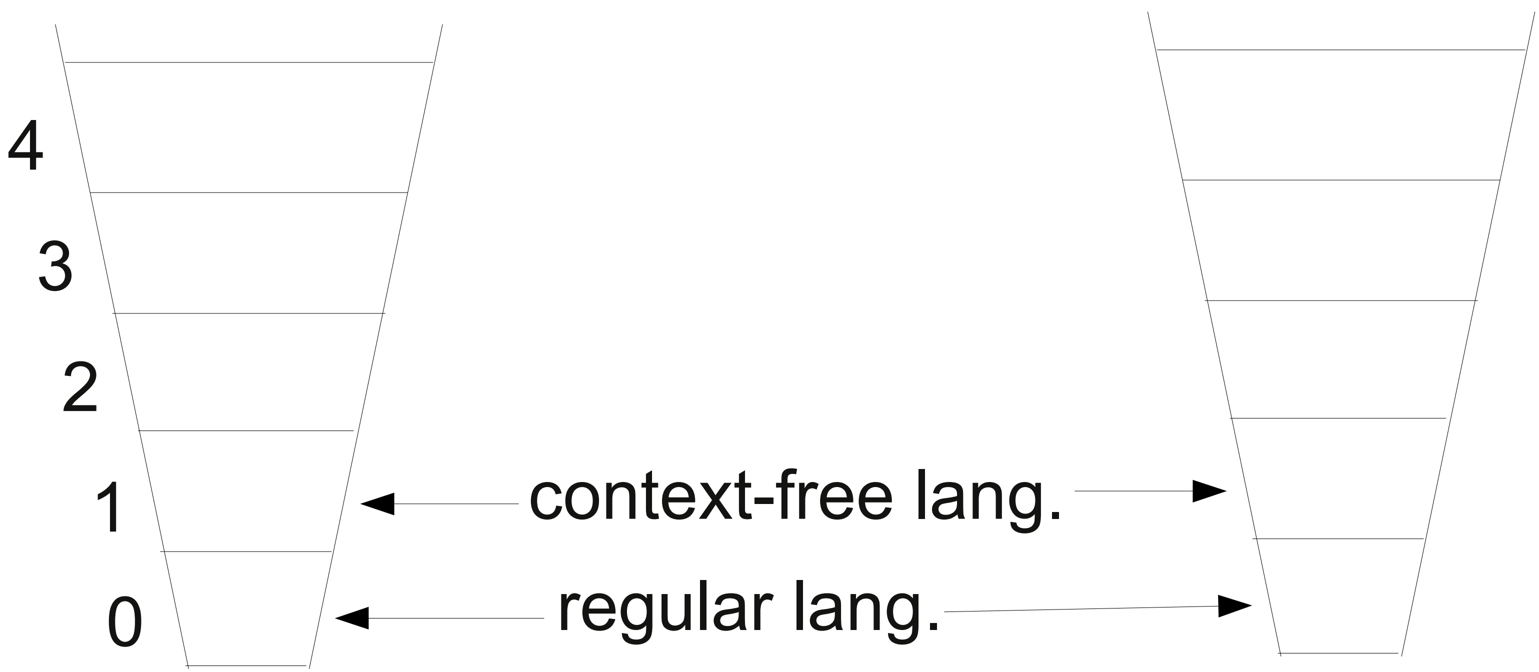
deterministic H-O pushdown automata

safe det. H-O grammars

Caucal hierarchy

deterministic H-O pushdown automata with collapse (panic) operation

all det. H-O grammars



Two hierarchies:

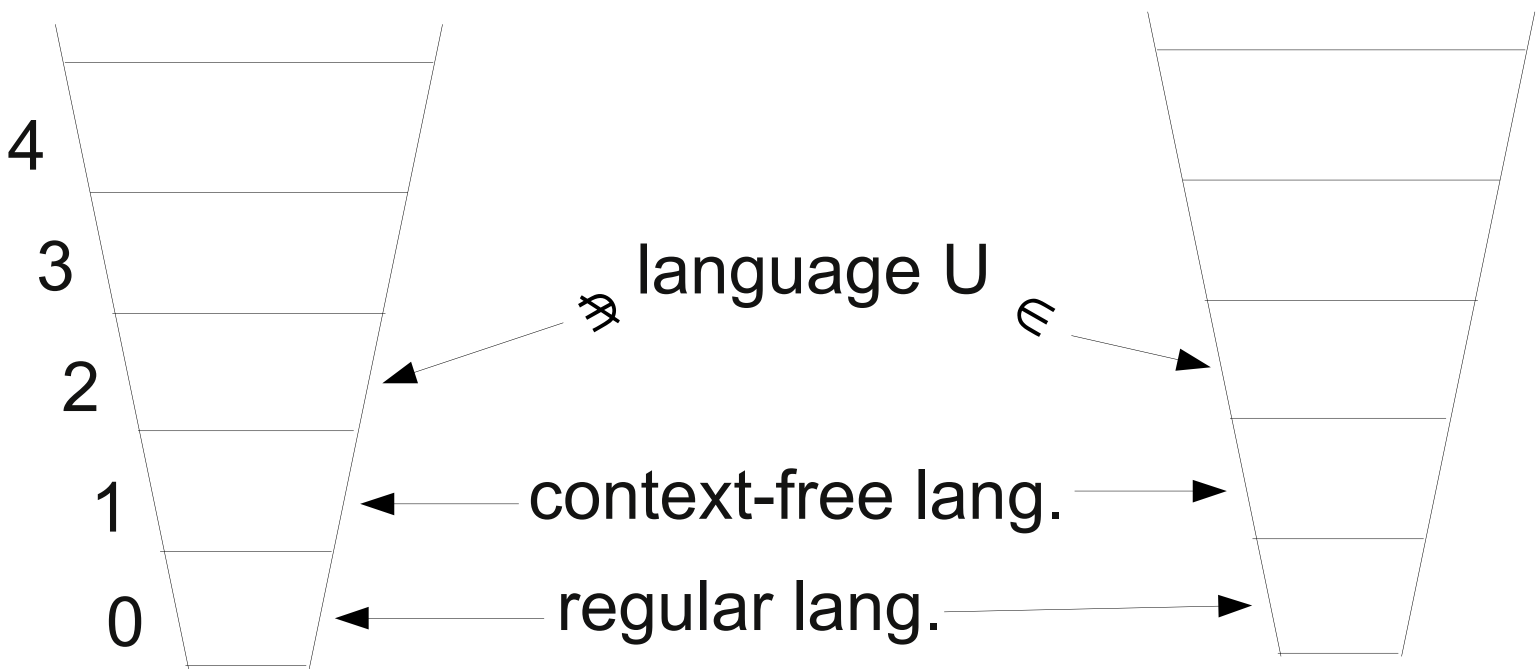
deterministic H-O pushdown automata

safe det. H-O grammars

Caucal hierarchy

deterministic H-O pushdown automata with collapse (panic) operation

all det. H-O grammars



The splitting language (proposed by P. Urzyczyn)

alphabet: [,], $*$

PBE = prefixes of bracket expressions, e.g. [[][

BE = (balanced) bracket expressions, e.g. [[][]]

U={u$*^n$: u∈PBE, v is the longest suffix of u which is BE,
n=|u|-|v|}

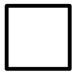
for example:

[[ ] [[ ] [[ ]] $* * * *$ ∈U

How to recognize U by an automaton with collapse?

U={u$\ast^n$: u∈PBE, v is the longest suffix of u which is BE, n=|u|-|v|}

- one stack symbol
- first order stack counts the number of currently open brackets
- a copy is done after each bracket

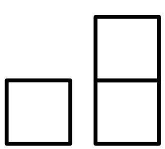


[[ ] [[ ] [[ ]] * * * *

How to recognize U by an automaton with collapse?

$U=\{u*^{n} : u \in PBE$, v is the longest suffix of u which is BE, $n=|u|-|v|\}$

- one stack symbol
- first order stack counts the number of currently open brackets
- a copy is done after each bracket

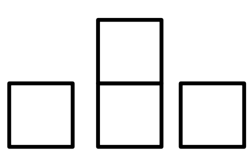


[[ ] [[ ] [[ ]] * * * *

How to recognize U by an automaton with collapse?

U={u$\ast^{n}$: u∈PBE, v is the longest suffix of u which is BE, n=|u|-|v|}

- one stack symbol
- first order stack counts the number of currently open brackets
- a copy is done after each bracket

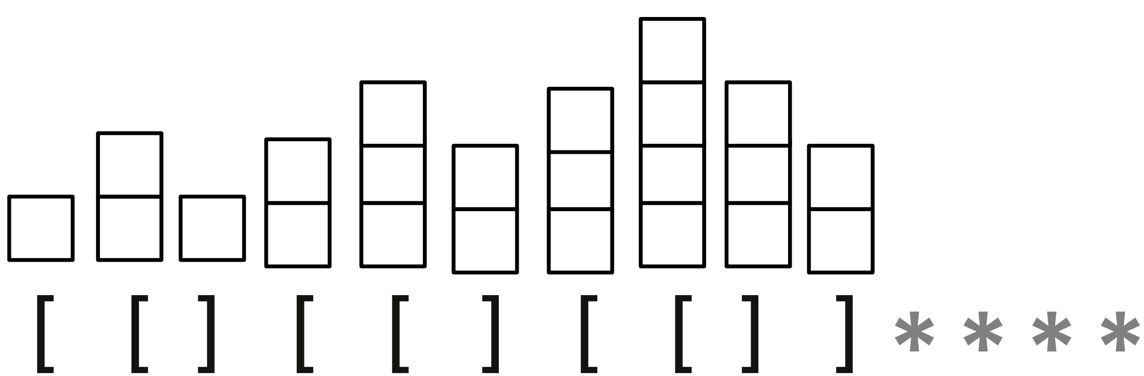


[[ ] [[ ] [[ ]] * * * *

How to recognize U by an automaton with collapse?

$U=\{u\ast^{n} : u\in PBE$, v is the longest suffix of u which is BE, $n=|u|-|v|\}$

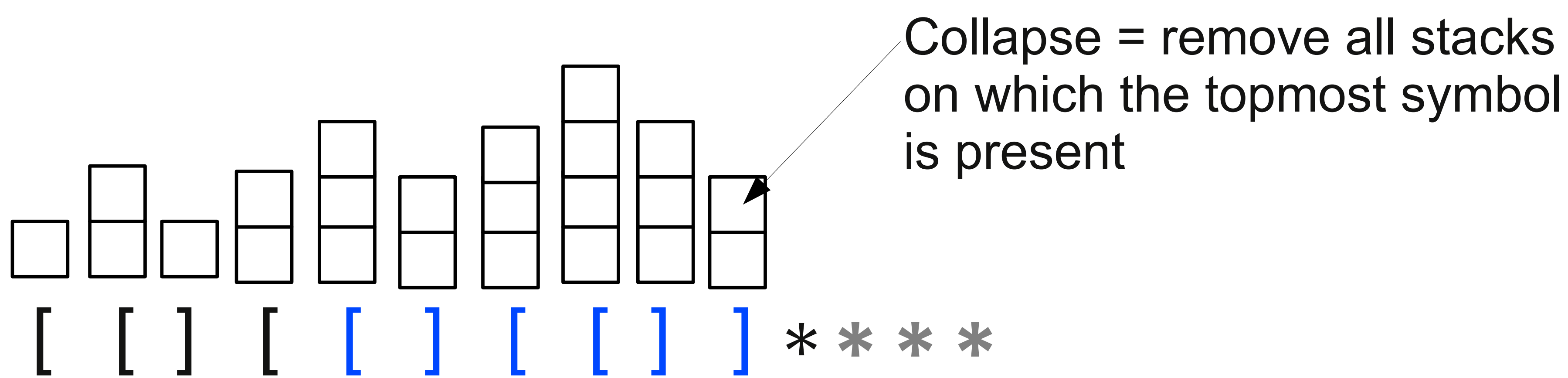
- one stack symbol
- first order stack counts the number of currently open brackets
- a copy is done after each bracket



How to recognize U by an automaton with collapse?

U={u$*^n$: u∈PBE, v is the longest suffix of u which is BE, n=|u|-|v|}

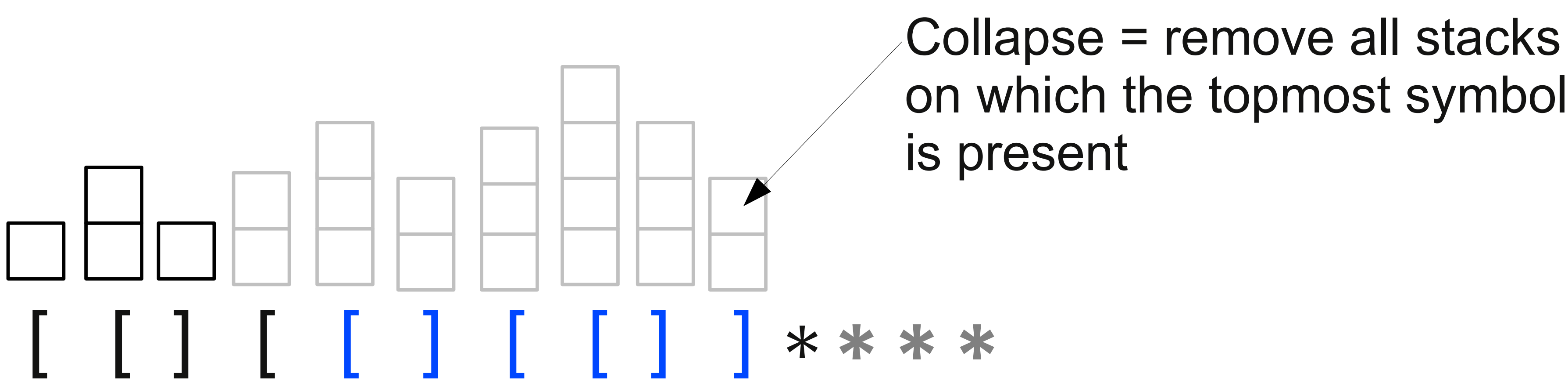
- one stack symbol
- first order stack counts the number of currently open brackets
- a copy is done after each bracket
- on the first star make the collapse
- count the number of stacks



How to recognize U by an automaton with collapse?

$U=\{u*^{n} : u \in PBE, v \text{ is the longest suffix of } u \text{ which is BE}, n=|u|-|v|\}$

- one stack symbol
- first order stack counts the number of currently open brackets
- a copy is done after each bracket
- on the first star make the collapse
- count the number of stacks



Ideas of the proof that collapse is necessary

Assume that A (automaton without collapse) recognizes U.

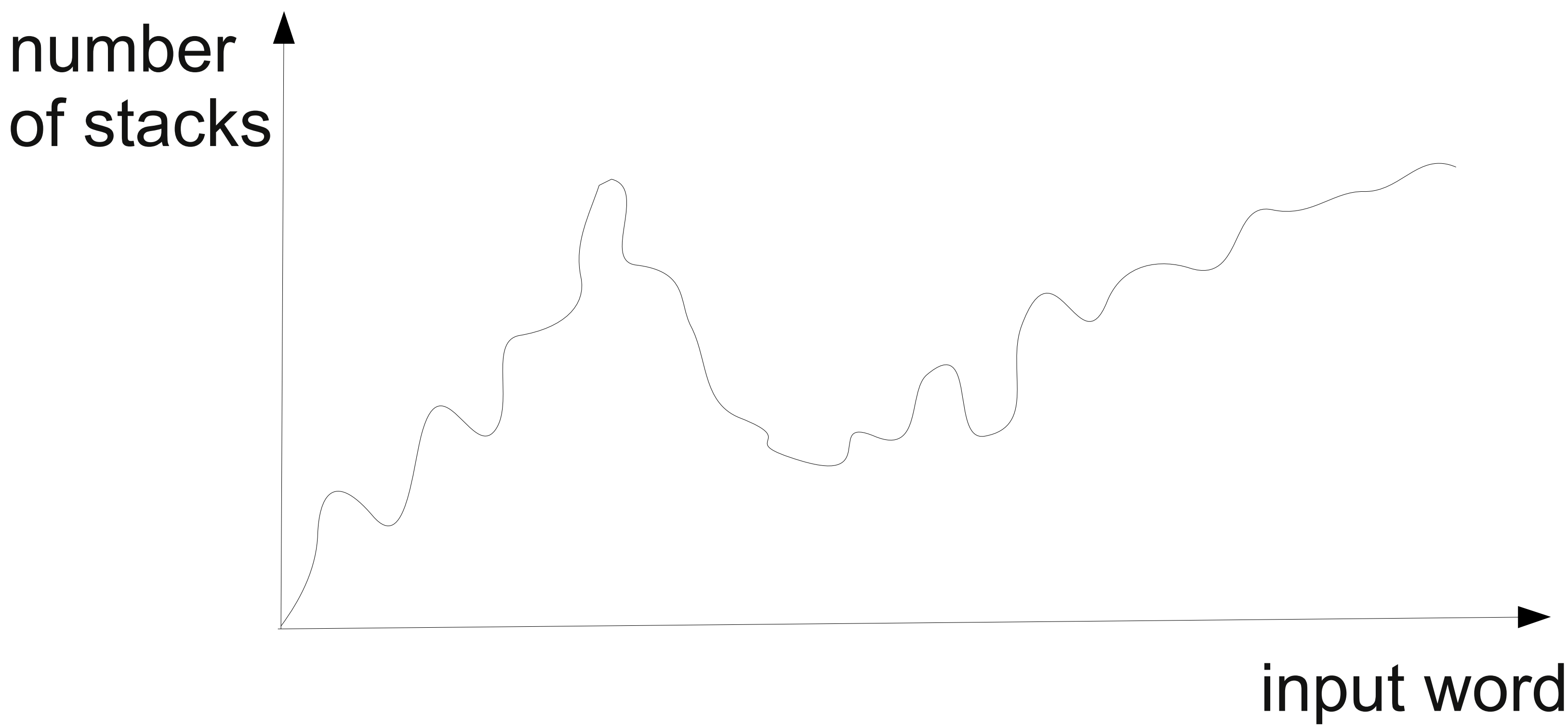
We first normalize A, then we show a contradiction.

Ideas of the proof that collapse is necessary

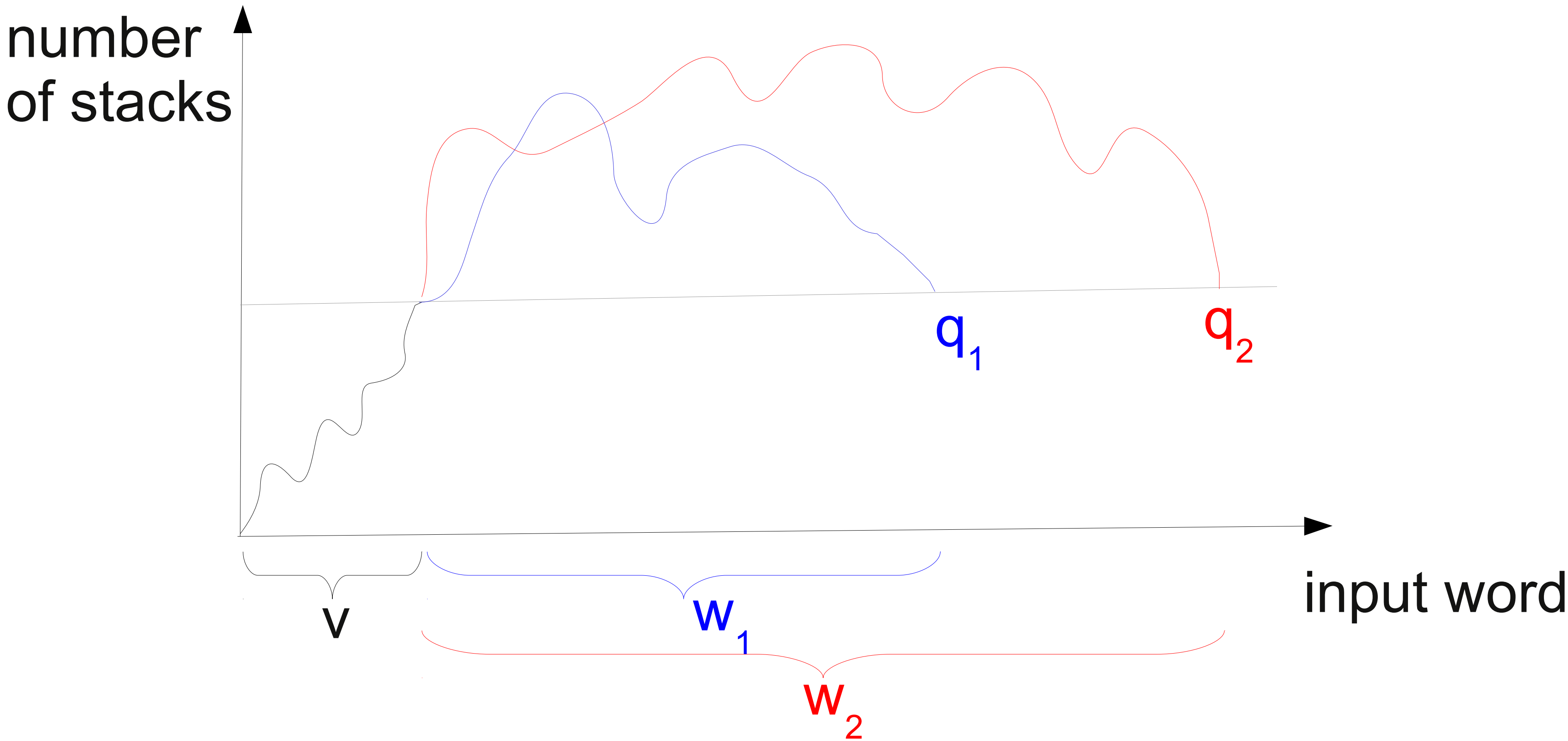
Assume that A (automaton without collapse) recognizes U.

We first normalize A, then we show a contradiction.

It is important to observe how the number of stacks changes (while A is reading a word).



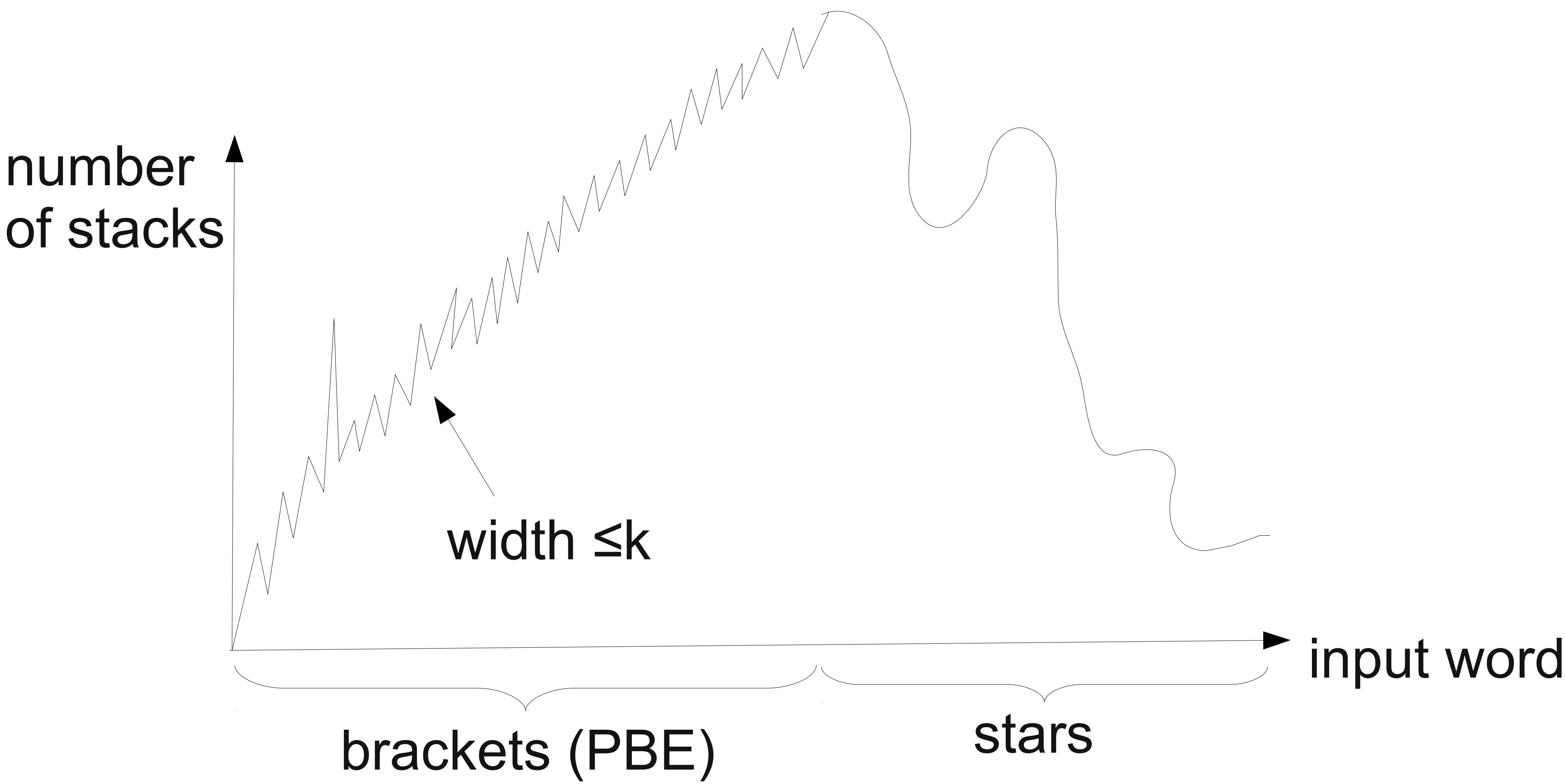
Collapse is necessary - observation 1



If $q_1=q_2$, then vw_1 and vw_2 are equivalent ($vw_1u\in U \Leftrightarrow vw_2u\in U$)
For fixed v, the number of stacks decrease below the level after v only for |Q| classes of vw.
But there are many classes of PBE → this situation is very rare.

Collapse is necessary - observation 1

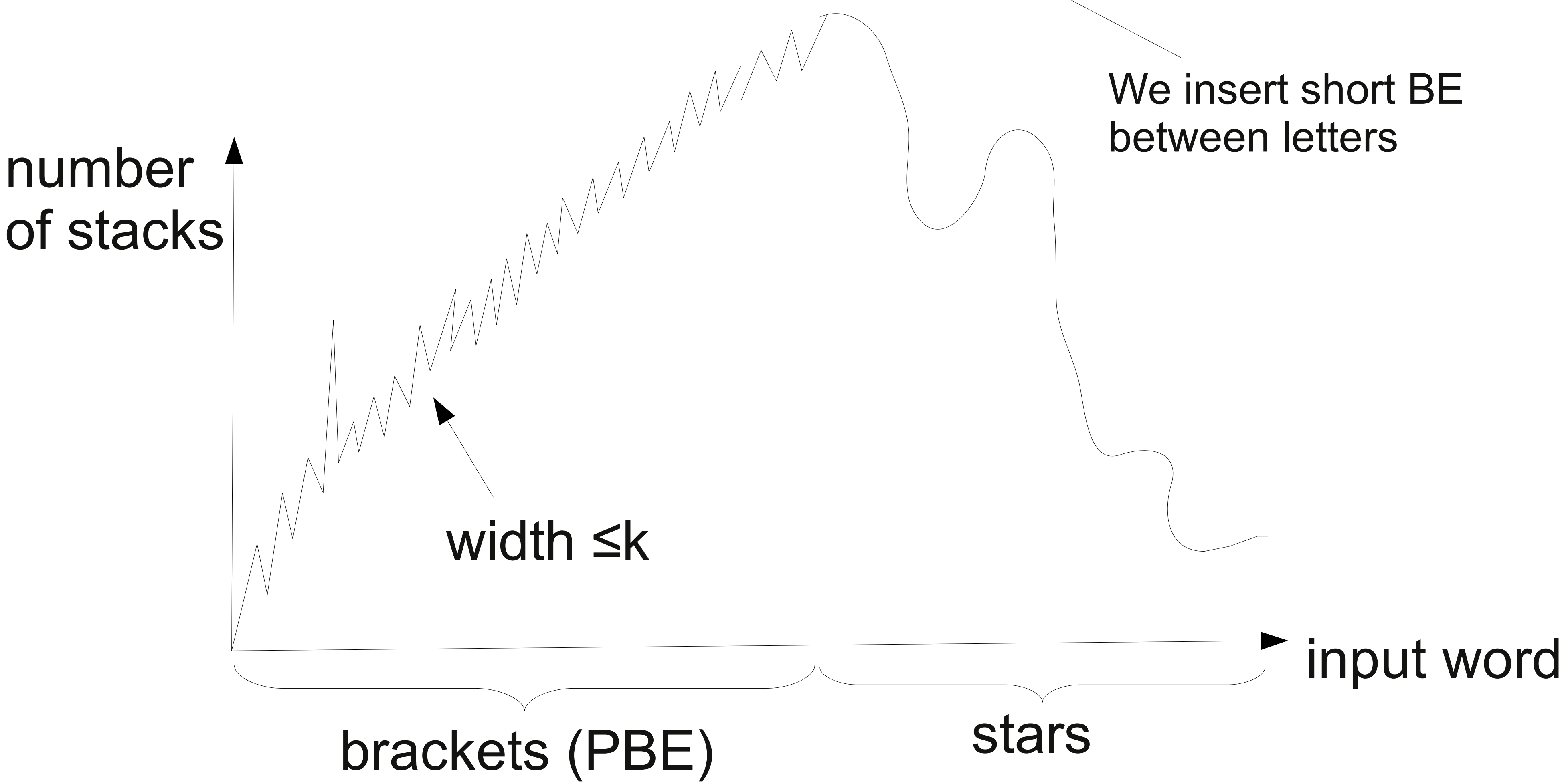
For "most" words A behaves like that:



Collapse is necessary - observation 1

For “most” words A behaves like that:

Each word can be “slightly modified” such that...

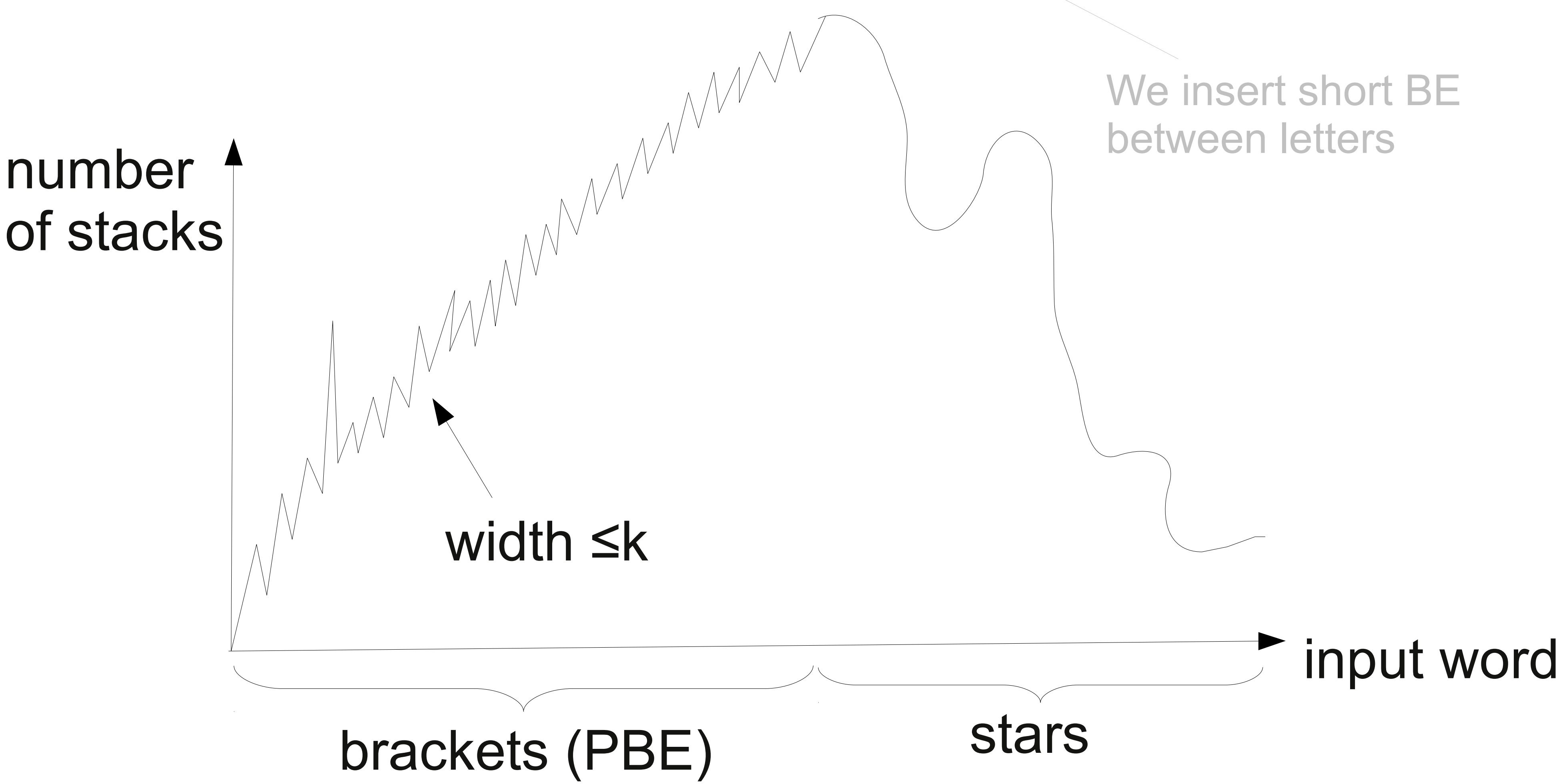


Collapse is necessary - observation 1

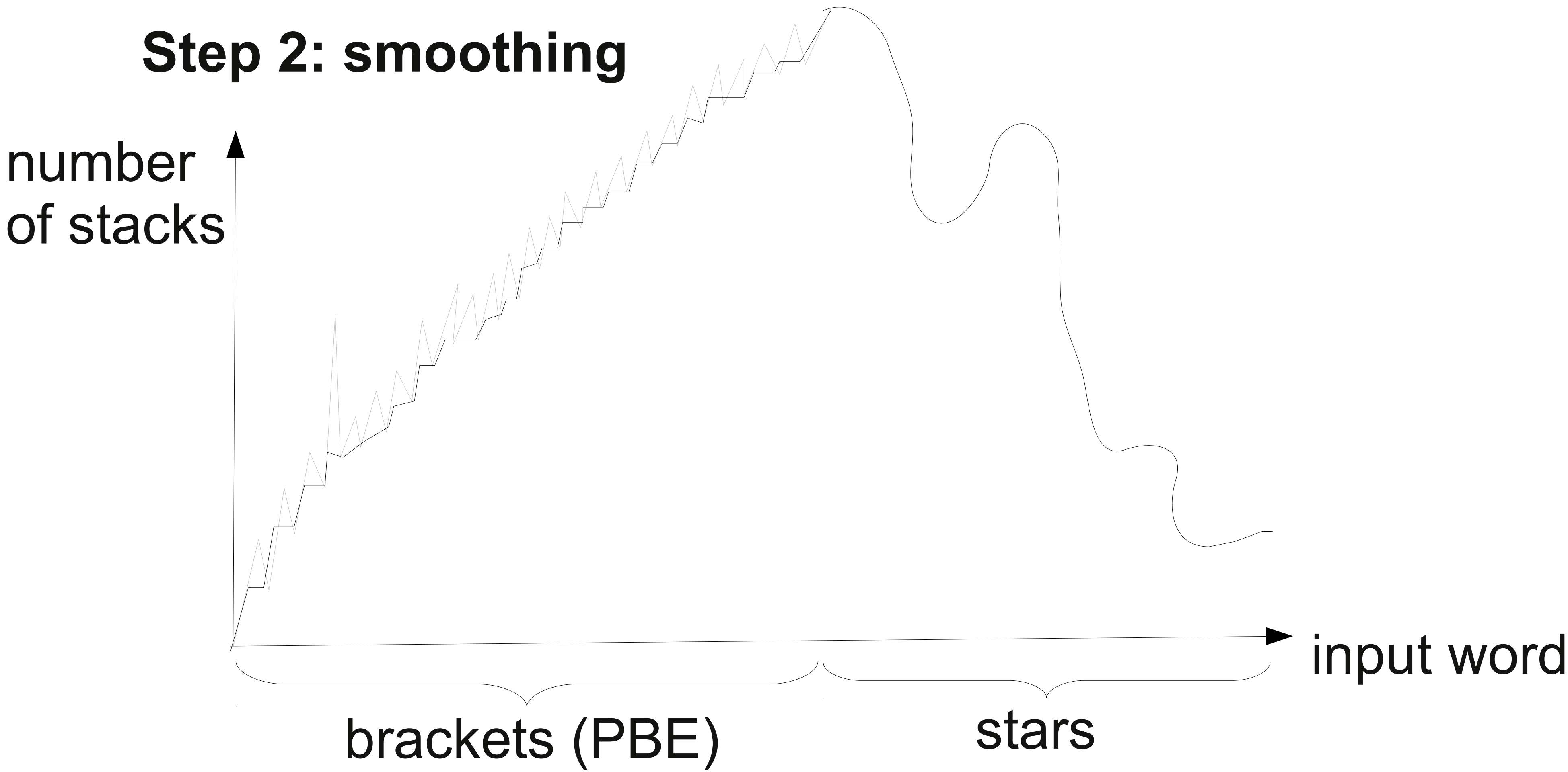
For “most” words A behaves like that:

Assume that for all words A behaves like that:

Each word can be “slightly modified” such that...



Step 2: smoothing



Construct a new automaton B (recognizing U) basing on A, such that the number of stacks never decreases while B is reading the brackets.
(the number of stacks of B = the minimal number of stacks of A during the last k letters)

Lemma 3

For any A there exists B such that:

- they do the same operations and accept the same words (but B may have more states and stack symbols), and
- after reading v, B “knows” if for some w there is $vw \in L(A)$.

(proof: construct B basing on A)

There is B recognizing U such that:

- the number of stacks never decreases while B is reading the brackets, and
- B knows if it has read a PBE or not.

Special words:

$$u_n = \underbrace{[^{n+1}]^n\,[^{n+1}]^n\,[^{n+1}]^n\,[^{n+1}]^n\,[^{n+1}]^n\,[^{n+1}]^n}_{|Q|+1 \text{ times}}$$

← number of open brackets is $|Q|+1$

Lemma 4.
If a (order 1) deterministic PDA recognizes PBE, after reading u_n it has at most C symbols on the stack (where C is a constant not depending on n).

⇓ $push_2$ is useless without pop_2

Automaton A (recognizing U) after reading u_n has at most C symbols on the last first level stack.

A final argument: problem with communication.

$$u_n = \underbrace{[^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n}_{|Q|+1 \text{ times}} \qquad u_{n,k} = u_n]^k * * * * *$$

Let s=the number of stacks after reading u_n

There are two parts of the computation:

1) Part reading u_n + part after the number of stacks becomes s-1.

2) Part after u_n using s or more stacks.

A final argument: problem with communication.

$$u_n = \underbrace{[^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n}_{|Q|+1 \text{ times}} \qquad u_{n,k} = u_n]^k*****$$

Let s=the number of stacks after reading u_n

There are two parts of the computation:

1) Part reading u_n + part after the number of stacks becomes s-1.
   This part knows n.
2) Part after u_n using s or more stacks.
   This part knows k.

A final argument: problem with communication.

$$u_n = \underbrace{[^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n}_{|Q|+1 \text{ times}} \qquad u_{n,k} = u_n]^k * * * * *$$

Let s=the number of stacks after reading u_n

There are two parts of the computation:

1) Part reading u_n + part after the number of stacks becomes s-1.
   This part knows n.

2) Part after u_n using s or more stacks.
   This part knows k.

Communication 1→2: the s-th stack is passed, which is of constant size, hence 2 does not know n.

Communication 2→1: only a state is passed, |Q| possibilities, hence 1 does not know k (which has |Q|+1 possible values).

A final argument: problem with communication.

$$u_n = \underbrace{[^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n [^{n+1}]^n}_{|Q|+1 \text{ times}} \qquad u_{n,k} = u_n]^k * * * * *$$

Let s=the number of stacks after reading u_n

There are two parts of the computation:

1) Part reading u_n + part after the number of stacks becomes s-1.
   This part knows n.

2) Part after u_n using s or more stacks.
   This part knows k.

Communication 1→2: the s-th stack is passed, which is of constant size, hence 2 does not know n.
Communication 2→1: only a state is passed, |Q| possibilities, hence 1 does not know k (which has |Q|+1 possible values).

The number of stars should be $(2n+1)\cdot(|Q|+1-k)$,
but it is the sum of stars accepted by 1 and by 2. → contradiction

Proof of Lemma 3

Lemma 2 (about smoothing) is proved similarly

Lemma 3: For any A there exists B such that:

- they do the same operations and accept the same words

  (but B may have more states and stack symbols), and
- after reading v, B “knows” if for some w there is vw∈L(A).

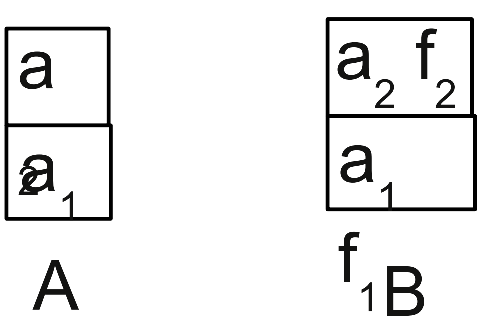
Assume that A (and B) is a first order PDA.

For each configuration (stack content) define f:Q→{acc,0}

To define f(q) start A in that configuration from a state q.

If it accepts (after reading some word), we take f(q)=acc, otherwise f(q)=0.

We product the stack alphabet with such functions.



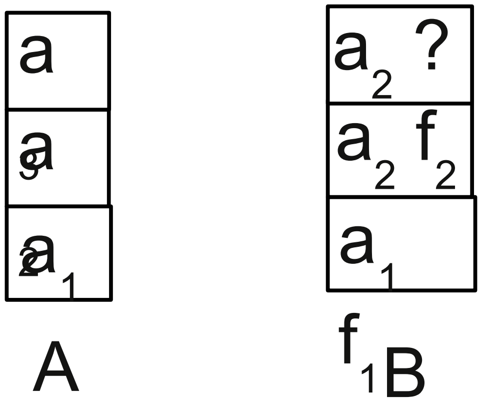
Proof of Lemma 3

Lemma 2 (about smoothing) is proved similarly

Lemma 3: For any A there exists B such that:

- they do the same operations and accept the same words (but B may have more states and stack symbols), and
- after reading v, B "knows" if for some w there is $vw \in L(A)$.

Assume that A (and B) is a first order PDA.
For each configuration (stack content) define $f: Q \rightarrow \{acc, 0\}$
To define f(q) start A in that configuration from a state q.
If it accepts (after reading some word), we take f(q)=acc, otherwise f(q)=0.
We product the stack alphabet with such functions.



B can calculate these functions:
f_k depends only on a_k and f_{k-1}

Proof of Lemma 3

Lemma 2 (about smoothing) is proved similarly

Lemma 3: For any A there exists B such that:

• they do the same operations and accept the same words
  (but B may have more states and stack symbols), and

• after reading v, B "knows" if for some w there is vw∈L(A).

Now let A (and B) be a second order PDA.

B can not compute functions f, because after copying a stack, they are no longer valid.

Proof of Lemma 3

Lemma 2 (about smoothing) is proved similarly

Lemma 3: For any A there exists B such that:

- they do the same operations and accept the same words
  (but B may have more states and stack symbols), and
- after reading v, B "knows" if for some w there is $vw \in L(A)$.

Now let A (and B) be a second order PDA.

Now, for each configuration (stacks content) we define

f_1:Q→{acc}∪P(Q), assigned to elements, and

f_2:Q→{acc,0}, assigned to first order stacks.

- To define $f_1(q)$ start A in that configuration from a state q.
  If it can accept without pop_2 we take $f_1(q)$=acc,
  otherwise $f_1(q)$=the set of states after pop_2.
- To define $f_2(q)$ make pop_2 and start A from a state q.
  If it accepts, we take $f_2(q)$=acc, otherwise $f_2(q)$=0.

B can calculate both these functions.

Summary

deterministic higher-order pushdown automata

	without collapse		with collapse
Solved:	level 2	$\neq$	level 2
Open problems:			
	level n	$\neq$	level n
	$\bigcup_n$ level n	$\neq$	$\bigcup_n$ level n