On the Significance of the Collapse Operation

Paweł Parys
University of Warsaw

Higher order pushdown automata (H-O PDA) [Maslov 74, 76]

A 1-stack is an ordinary stack. A 2-stack
(resp. n + 1-stack) is a stack of 1-stacks (resp. n-stack).

Operations on 2-stacks: s_i are 1-stacks. Top of stack is on right.

$push_2$: $[s_1 \ldots s_{i-1} s_i]$ -> $[s_1 \ldots s_{i-1} s_i\, s_i]$

pop_2 : $[s_1 \ldots s_{i-1} s_i]$ -> $[s_1 \ldots s_{i-1}]$

$push_1 x$: $[s_1 \ldots s_{i-1} [a_1 \ldots a_{j-1} a_j]]$ -> $[s_1 \ldots s_{i-1} [a_1 \ldots a_{j-1} a_j\, x]]$

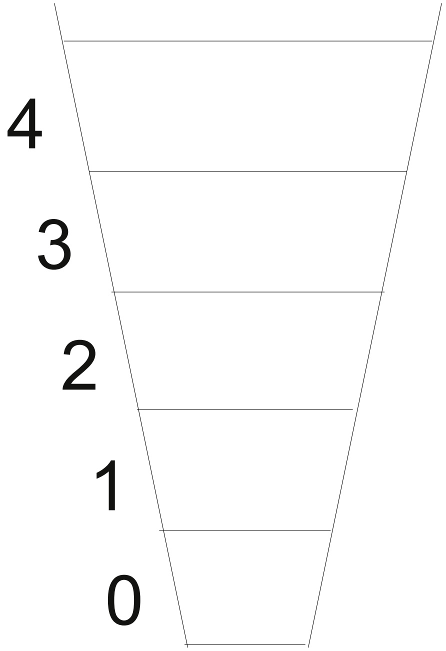
pop_1 : $[s_1 \ldots s_{i-1} [a_1 \ldots a_{j-1} a_j]]$ -> $[s_1 \ldots s_{i-1} [a_1 \ldots a_{j-1}]]$

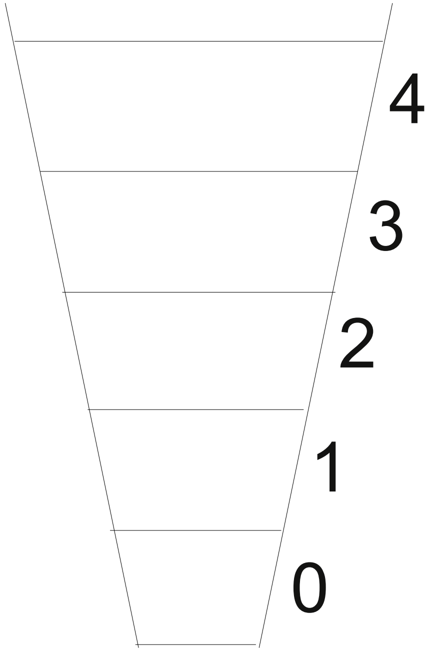
An **order-n PDA** has an order-n stack, and
has $push_i$ and pop_i for each $1 \leq i \leq n$.

Two hierarchies (of trees):

trees generated by
H-O pushdown systems

trees generated by
H-O recursion schemes





Are these two hierarchies equal?

- levels 0 and 1 – yes

Two hierarchies (of trees):

Are these two hierarchies equal?

- Knapik, Niwiński, Urzyczyn 2002

trees generated by
H-O pushdown systems

$\overset{?}{=}$

trees generated by
H-O schemes

$=$

trees generated by
safe H-O schemes

Two hierarchies (of trees):

Are these two hierarchies equal?

- Knapik, Niwiński, Urzyczyn 2002
- Caucal 2002

trees generated by
H-O pushdown systems $\overset{?}{=}$ trees generated by
H-O schemes

$=$

trees generated by
safe H-O schemes

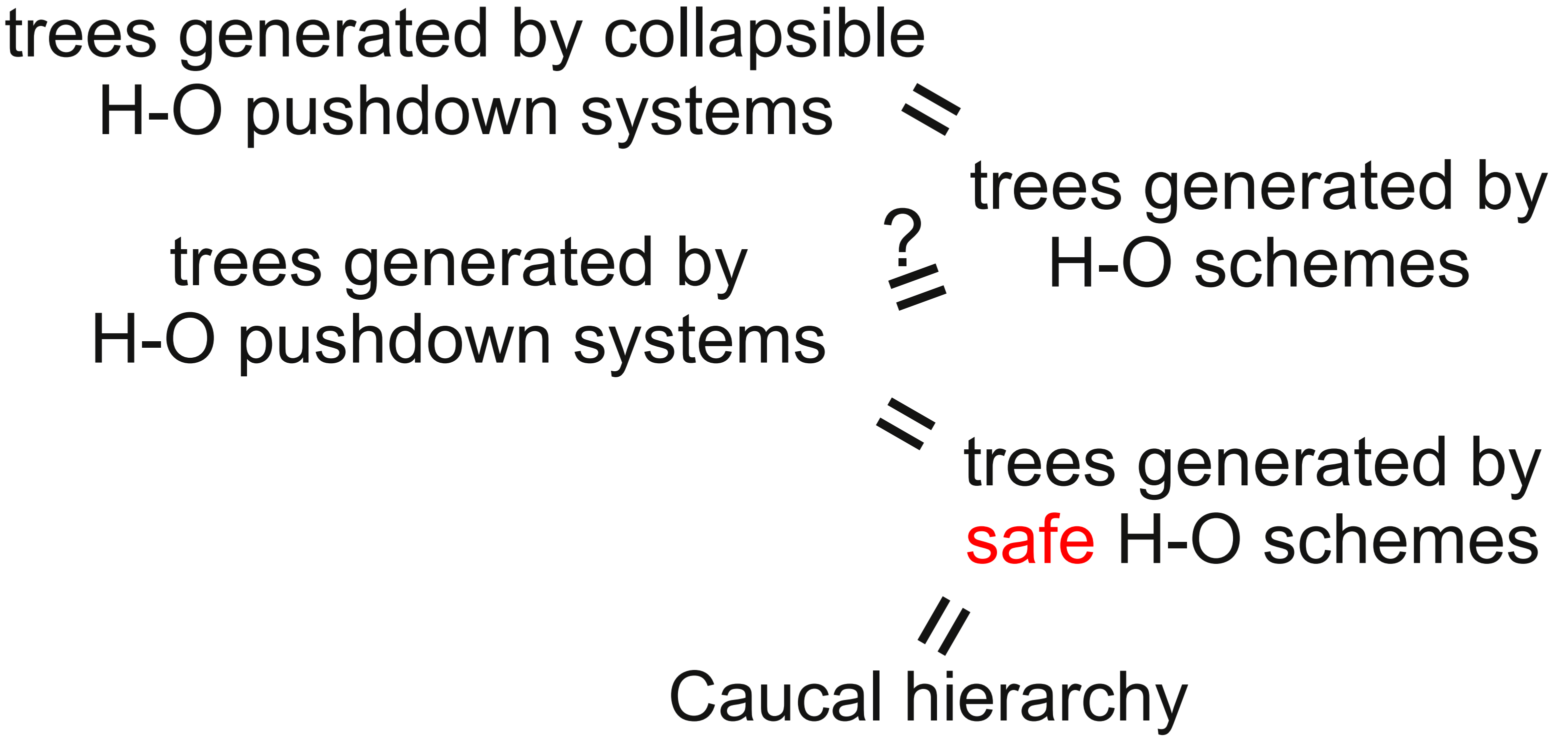
$=$

Caucal hierarchy

Two hierarchies (of trees):

Are these two hierarchies equal?

- Hague, Murawski, Ong, Serre 2008



trees generated by collapsible H-O pushdown systems

=

trees generated by H-O schemes

?=

trees generated by H-O pushdown systems

=

trees generated by safe H-O schemes

=

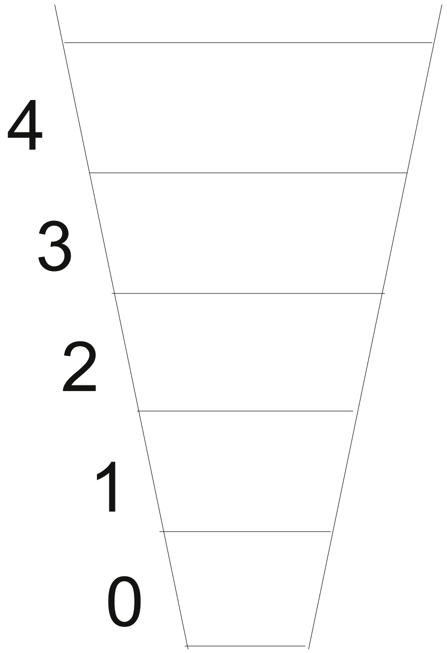
Caucal hierarchy

Two hierarchies (of trees):

H-O pushdown systems

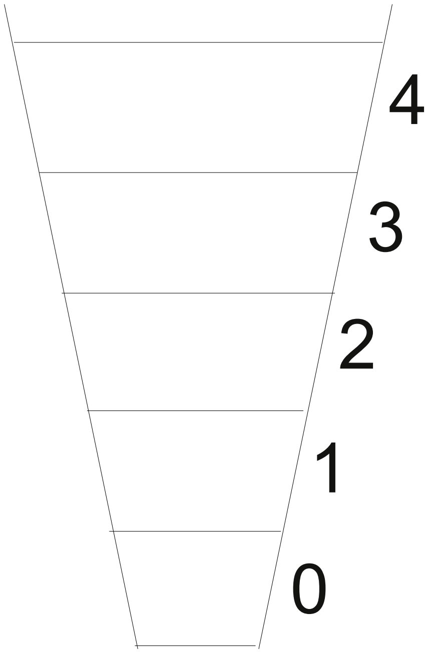
safe H-O schemes

Caucal hierarchy



collapsible H-O
pushdown systems

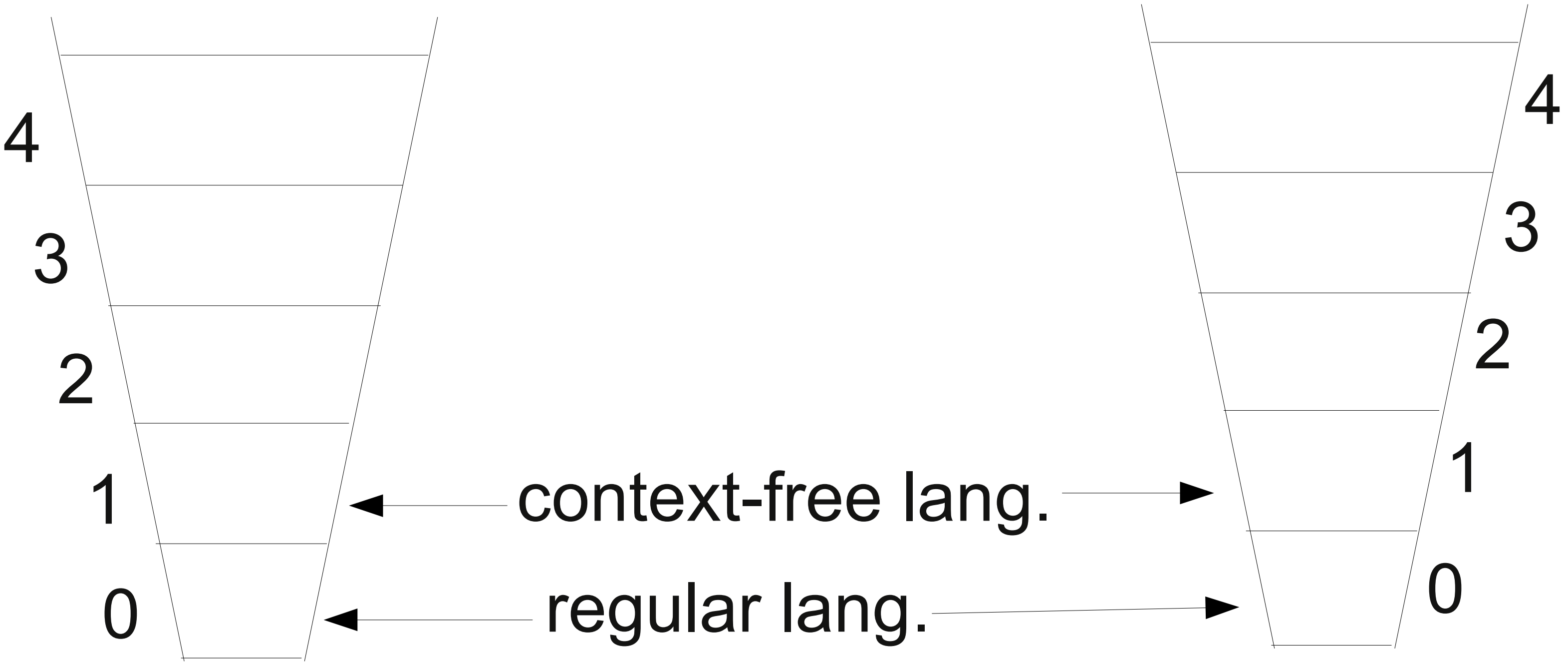
all H-O schemes



Equivalently: two hierarchies of word languages

deterministic H-O
pushdown automata

deterministic collapsible H-O
pushdown automata



context-free lang.

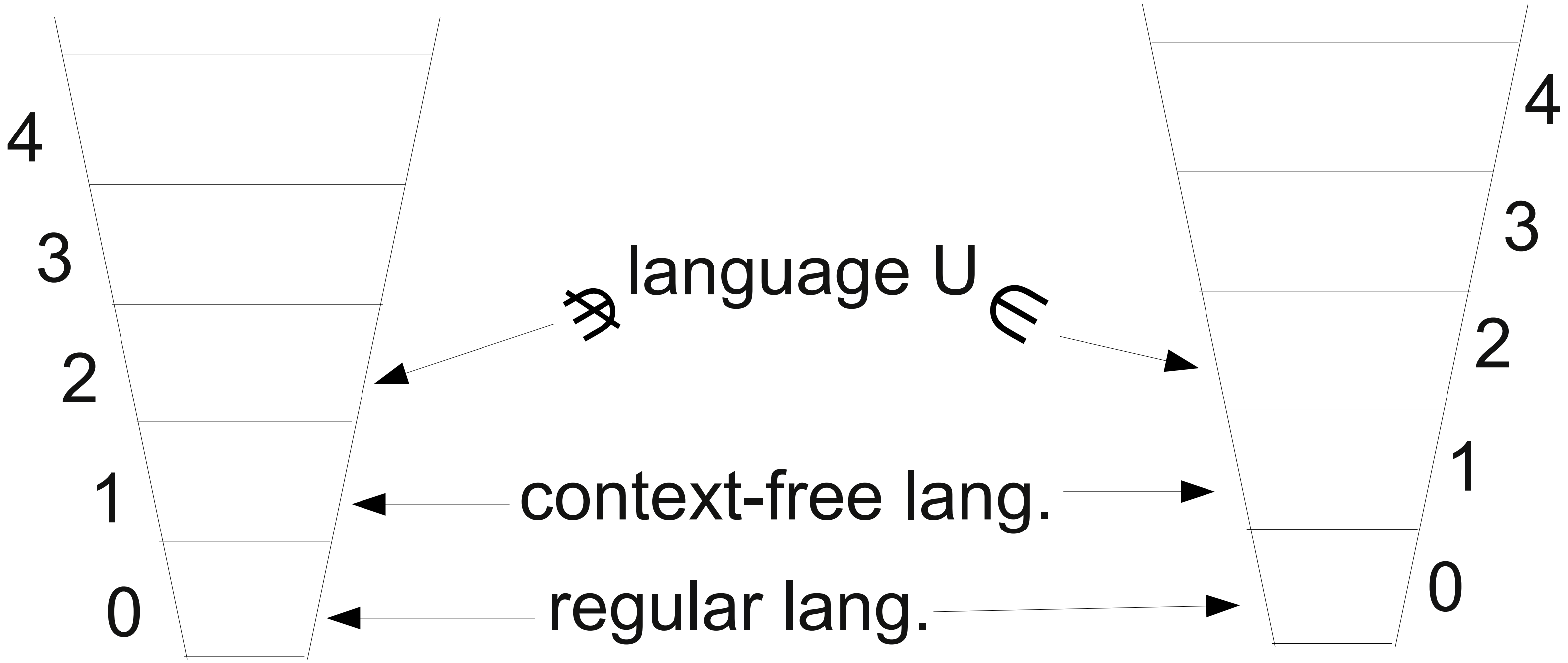
regular lang.

Previous result (STACS 2011):

- level 2 is different

deterministic H-O pushdown automata

deterministic collapsible H-O pushdown automata



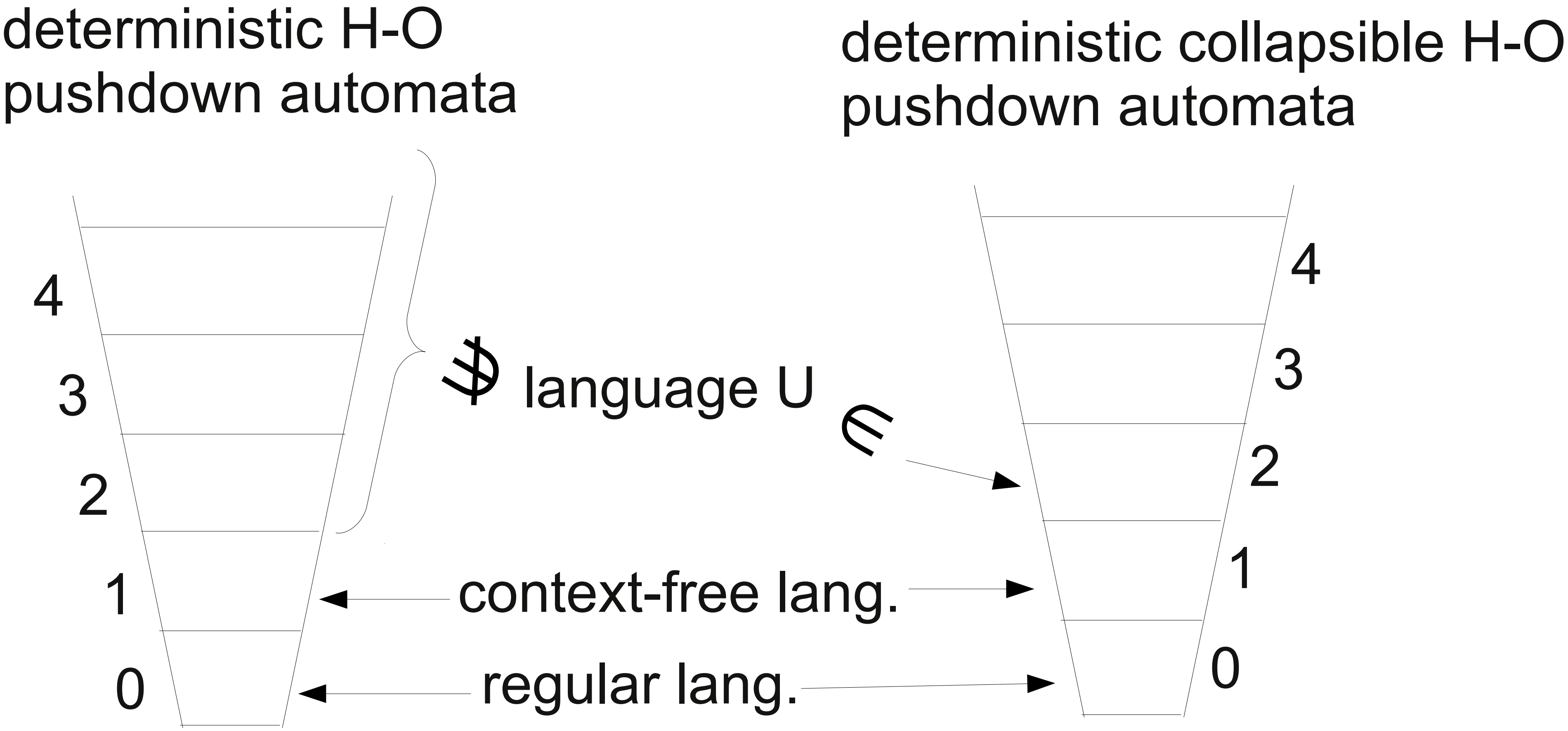
∉ language U ∈

context-free lang.

regular lang.

This result:

- the union of the hierarchies is different



deterministic H-O
pushdown automata

deterministic collapsible H-O
pushdown automata

∌ language U ∈

context-free lang.

regular lang.

Collapsible Pushdown Automata

Collapsible PDA are an extension of H-O PDA

Each 0-stack (stack symbol) is created with a fresh identifier.

For $2 \leq i \leq n$ we have a new operation collapse_i

It removes all (i-1)-stacks which contain the topmost symbol.

Notice: $\text{collapse}_1 = \text{pop}_1$

Example: Urzyczyn's language U

alphabet: $[$, $]$, $*$

U contains words of the form:

$$\underbrace{[\ldots[\ldots[}_{A}\ \underbrace{[\ldots]\ldots[\ldots]}_{B}\ \underbrace{*\ldots*}_{C}$$

- segment A is a prefix of a well-bracketed word that ends in [which is not matched in the entire word
- segment B is a well-bracketed word
- segments A and C have the same length

for example:

$$[\ [\ ]\ [\ [\ ]\ [\ [\ ]\ ]\ {*}{*}{*}{*} \ \in U$$

How to recognize U by an automaton with collapse?

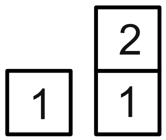
- one stack symbol
- first order stack counts the number of currently open brackets
- a copy (push$_2$) is done after each bracket

$\boxed{1}$

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

How to recognize U by an automaton with collapse?

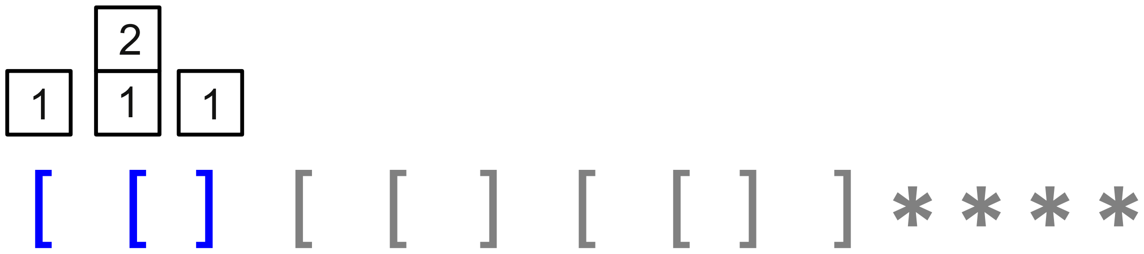
- one stack symbol
- first order stack counts the number of currently open brackets
- a copy ($push_2$) is done after each bracket



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

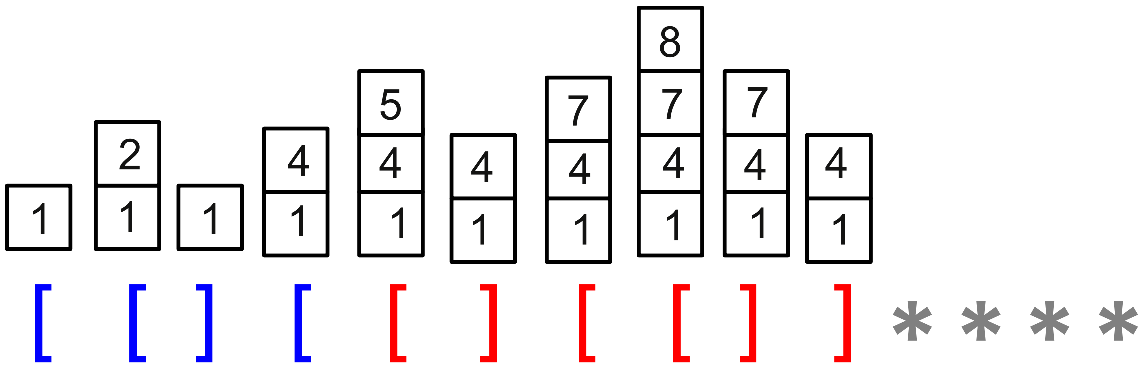
How to recognize U by an automaton with collapse?

- one stack symbol
- first order stack counts the number of currently open brackets
- a copy ($push_2$) is done after each bracket



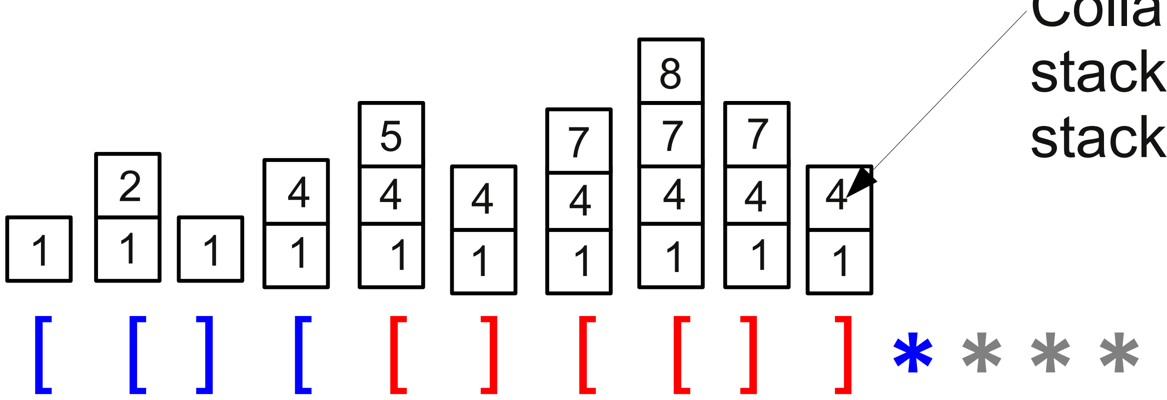
How to recognize U by an automaton with collapse?

- one stack symbol
- first order stack counts the number of currently open brackets
- a copy ($push_2$) is done after each bracket



How to recognize U by an automaton with collapse?

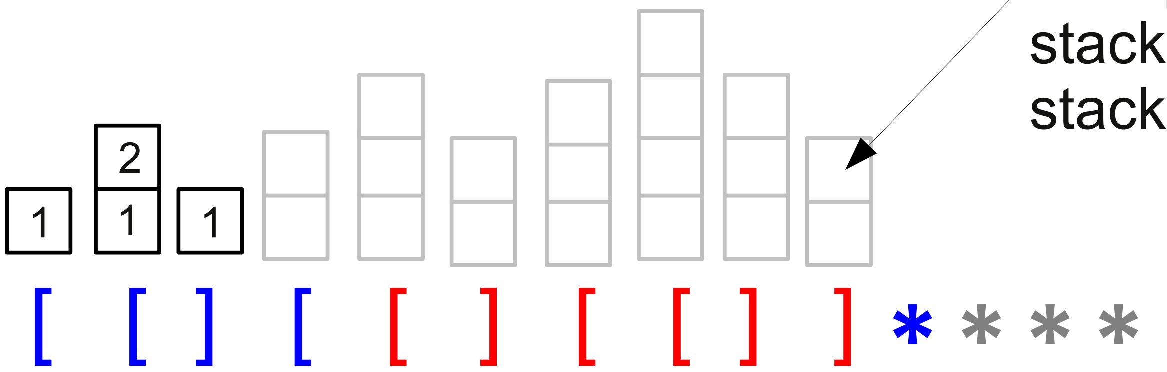
- one stack symbol
- first order stack counts the number of currently open brackets
- a copy ($push_2$) is done after each bracket
- on the first star we make the collapse
- we count the number of stacks



Collapse = remove all stacks on which this stack symbol is present

How to recognize U by an automaton with collapse?

- one stack symbol
- first order stack counts the number of currently open brackets
- a copy ($push_2$) is done after each bracket
- on the first star we make the collapse
- we count the number of stacks



Collapse = remove all stacks on which this stack symbol is present

Related open problem

The same question for nondeterministic (collapsible) H-O PDA:

Is there a language
- not recognized by any nondeterministic H-O PDA
- recognized by a nondeterministic Collapsible H-O PDA

(here the second levels are equal,
possibly there is a difference on level 3)