Collapse Operation Increases Expressive Power of Deterministic Higher Order Pushdown Automata

Paweł Parys

University of Warsaw

Motivation: from program verification to higher order pushdowns

Example

```
open(x, “foo”)
a := 0
while a<100 do
     read(x)
     a := a+1
close(x)
```

is the file “foo” accessed according to open,read*,close?

Motivation: from program verification to higher order pushdowns

Example

Step 1: information about infinite data domains is approximated.

```
open(x, "foo")
a := 0
while a<100 do
    read(x)
    a := a+1
close(x)
```

→

```
open(x, "foo")

while * do
    read(x)

close(x)
```

is the file "foo" accessed according to open,read*,close?

→

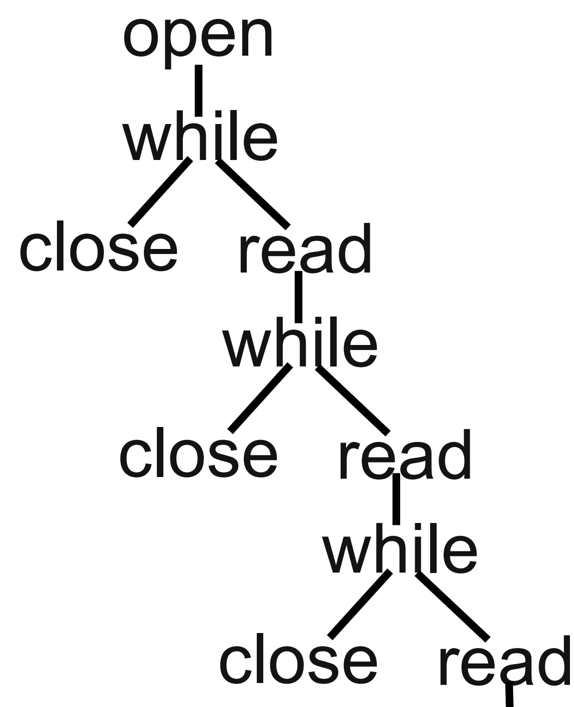
is the file "foo" accessed according to open,read*,close?

Motivation: from program verification to higher order pushdowns

Example

Step 2: consider the tree of possible control flows.

```
open(x, “foo”)
while * do
    read(x)
close(x)
```



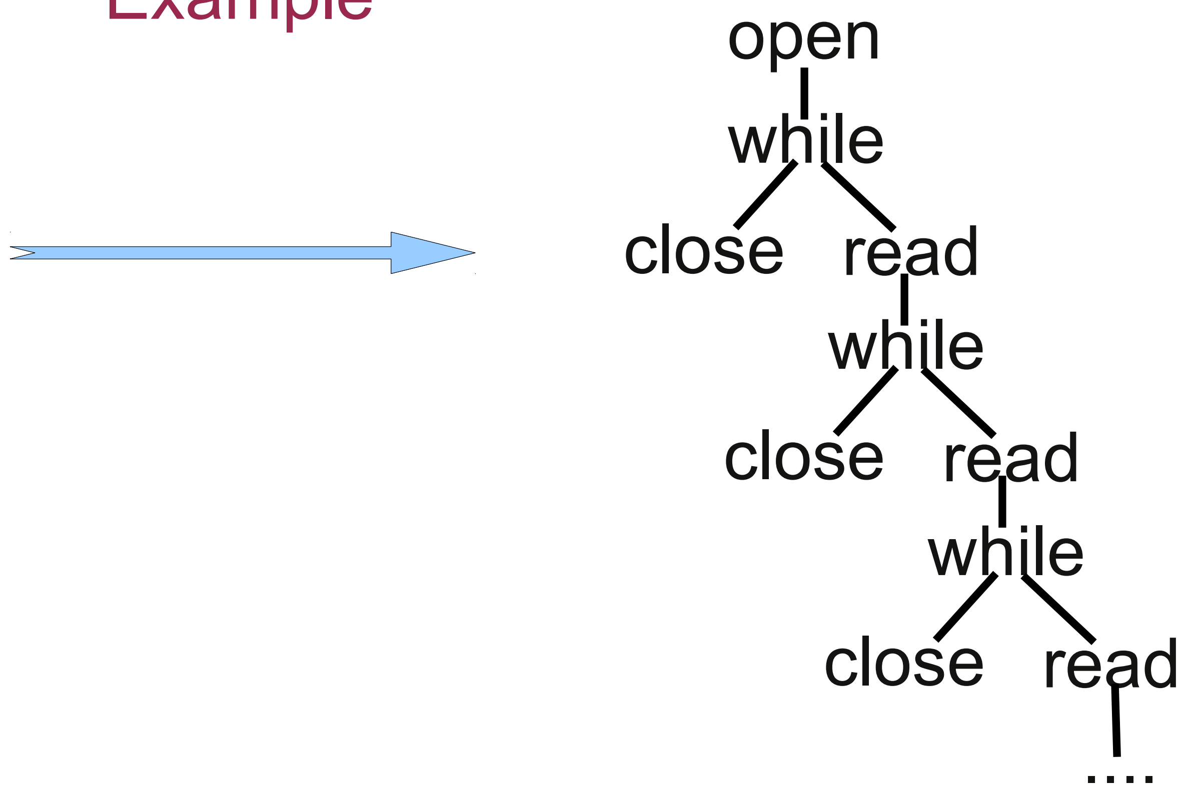
is the file “foo” accessed according to open,read*,close?

is each path labelled by open,read*,close?

Motivation: from program verification to higher order pushdowns

Example

```
open(x, “foo”)
while * do
    read(x)
close(x)
```



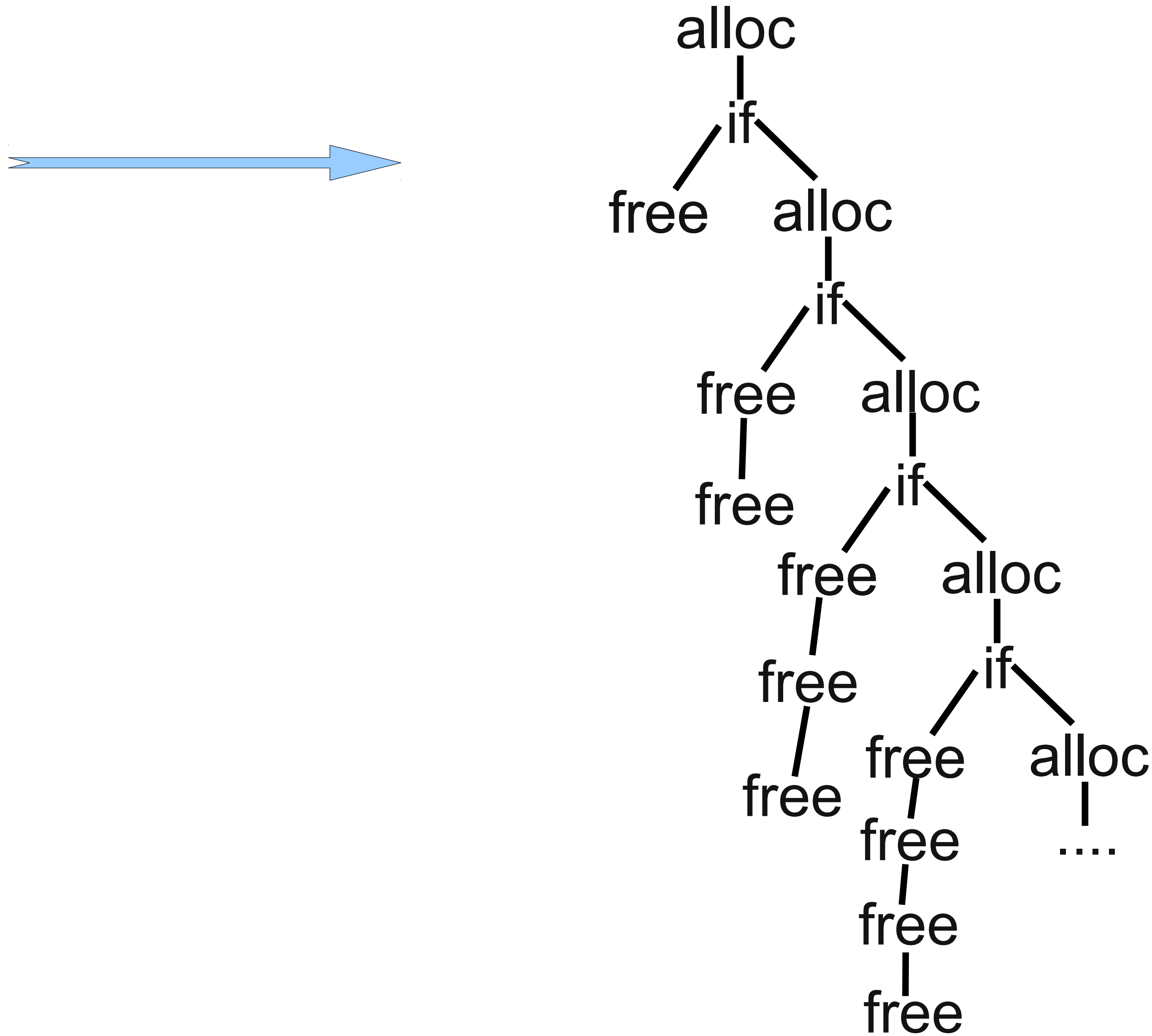
Observation: for programs without recursion, each path of the tree is a regular language.
(the program is a deterministic finite automaton)

Rabin 1969: Regular trees have decidable MSO theory.

Motivation: from program verification to higher order pushdowns

Example 2 - program with recursion

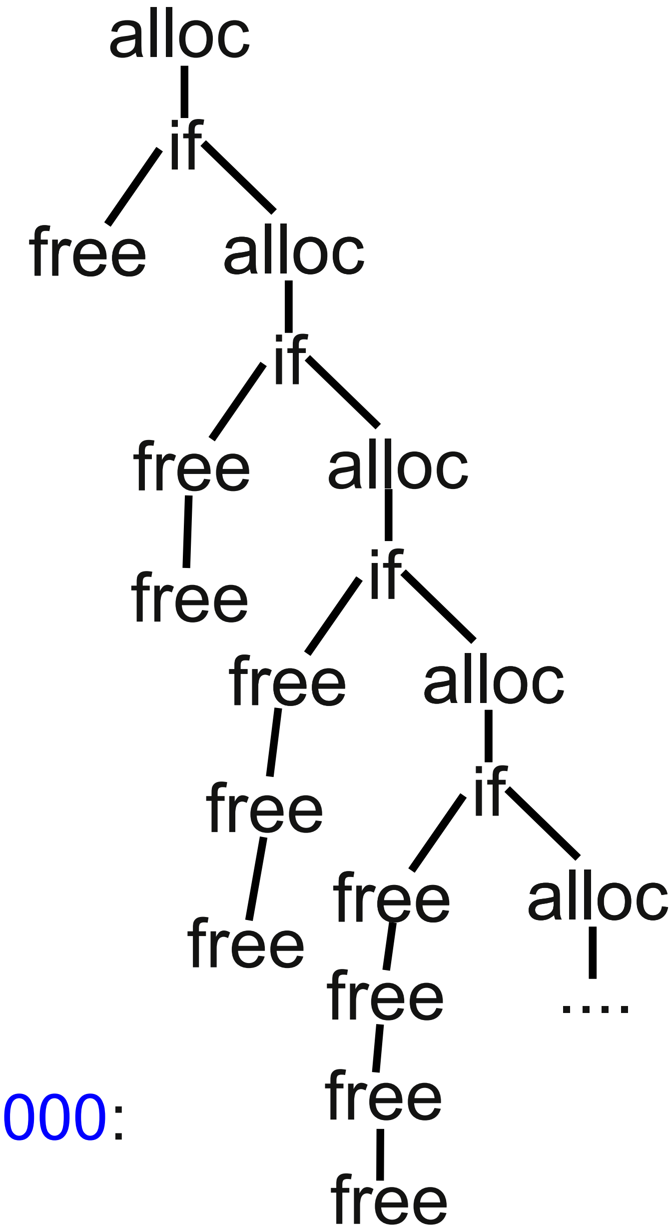
```
let f(x) =
    alloc(x)
    if * then f(x)
    free(x)
f(x)
```



Motivation: from program verification to higher order pushdowns

Example 2 - program with recursion

```
let f(x) =
    alloc(x)
    if * then f(x)
    free(x)
f(x)
```



Now the tree is not regular!!

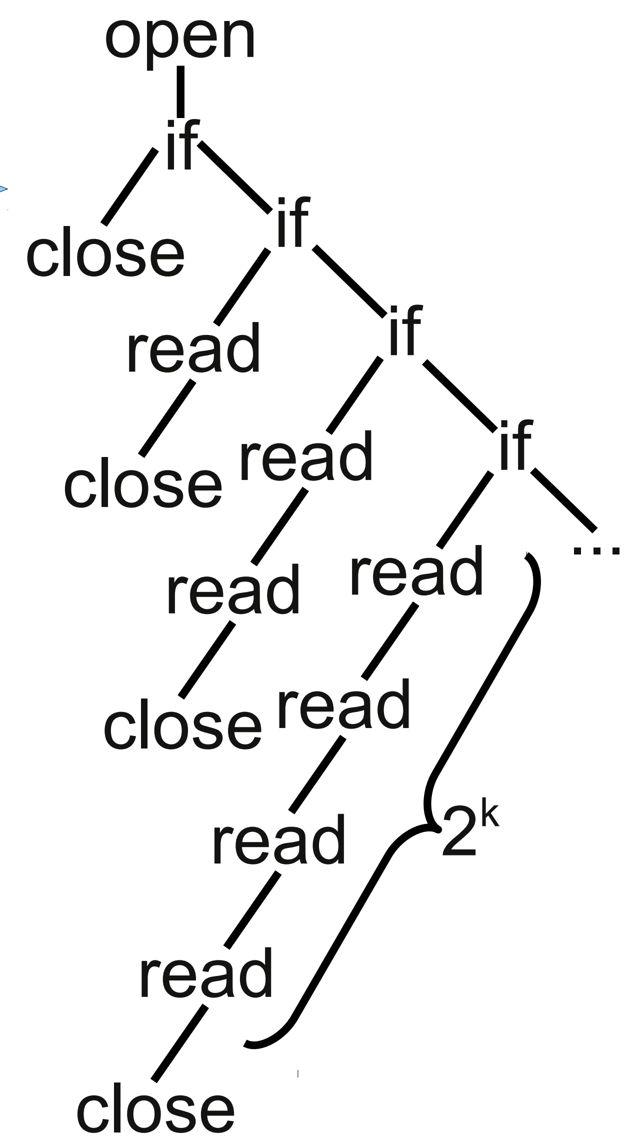
But each path is recognized by
a **deterministic** pushdown automaton.

Muller, Schupp 1985 / Caucal 1986 / Stirling 2000:
such trees have decidable MSO theory.

Motivation: from program verification to higher order pushdowns

What about higher order programs?

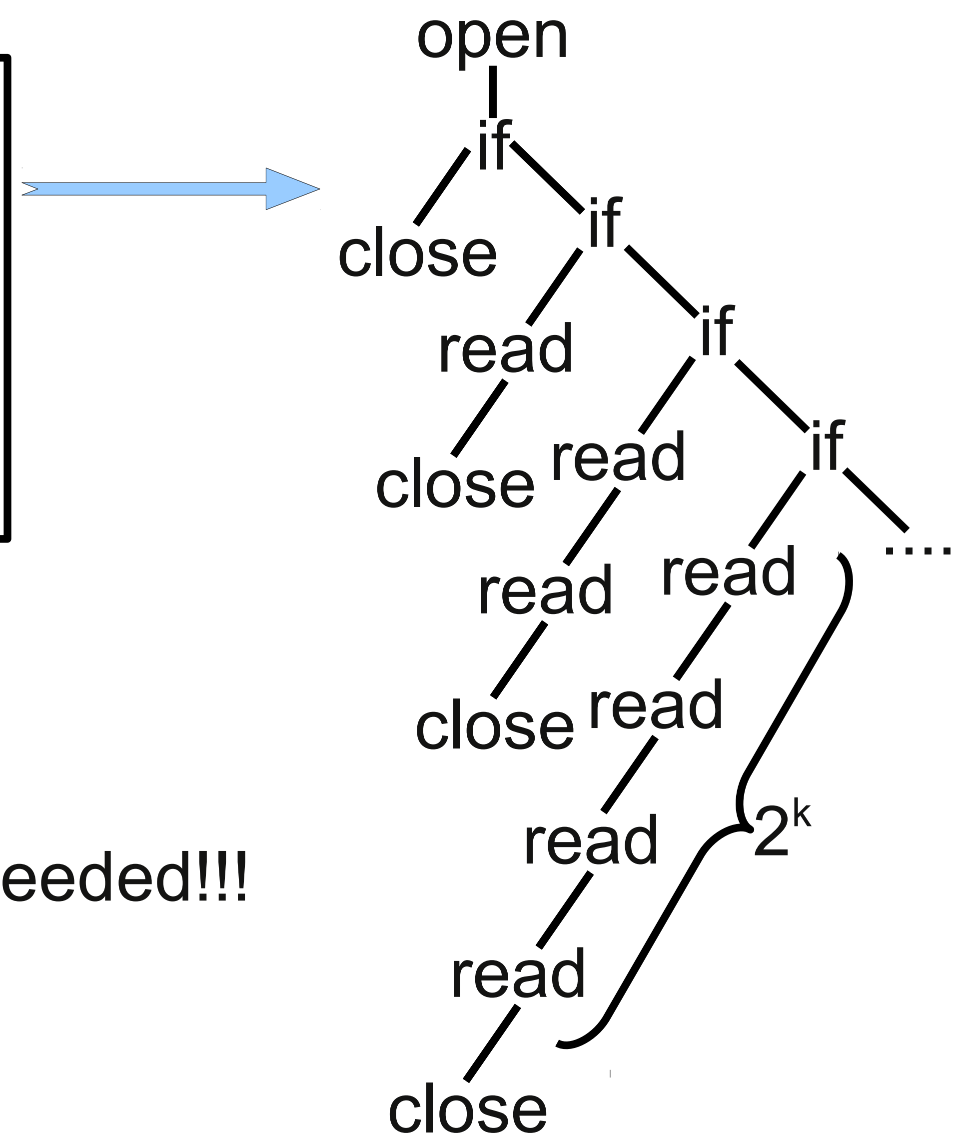
```
let f(x, g) =
    if * then g(x)
    else f(x, fun h x -> h(x); h(x))
open(x)
f(x, read)
close(x)
```



Motivation: from program verification to higher order pushdowns

What about higher order programs?

```
let f(x, g) =
    if * then g(x)
    else f(x, fun h x -> h(x); h(x))
open(x)
f(x, read)
close(x)
```



Better automata class is needed!!!

Higher order pushdown automata (HOPDA) [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 $1 \le i \le n$.

Relation between HOPDA and programs

We skip the formal definition

For each level we have introduced two classes of trees:

$\mathbf{PushdownTree}_n \Sigma$ = trees generated by order-n deterministic HOPDA

$\mathbf{RecSchTree}_n \Sigma$ = trees generated by order-n recursion scheme (program)

Are these classes equal?

For levels 0 and 1: yes

For levels >1: in some sense...

Relation between HOPDA and programs

$\mathbf{PushdownTree}_n \Sigma$ = trees generated by order-n deterministic HOPDA

$\mathbf{SafeRecSchTree}_n \Sigma$ = trees generated by order-n safe recursion scheme

Knapik, Niwiński, Urzyczyn 2002:

For each n, $\mathbf{PushdownTree}_n \Sigma = \mathbf{SafeRecSchTree}_n \Sigma$

and these trees have decidable MSO theory.

what is safety?

It is some syntactic constraint on the recursion schemes.

(the result of passing order-k parameters to a function has to be of order lower than k)

Safety restriction disappears at level 1.

Another characterization of these trees - the Caucal hierarchy (Caucal 2002)

$\mathbf{PushdownTree}_n \Sigma = \mathbf{SafeRecSchTree}_n \Sigma = \mathbf{CaucalTree}_n \Sigma$

Relation between HOPDA and programs

- Is the safety restriction essential for MSO decidability?

Ong 2006:
Trees from $\mathbf{RecSchTree}_n \Sigma$ have decidable MSO theory.

- What is the corresponding automata class?

Hague, Murawski, Ong, Serre 2008:
$\mathbf{RecSchTree}_n \Sigma$ contains exactly trees generated by collapsible deterministic HOPDA.

- Is safety really a restriction?

this paper:
$\mathbf{RecSchTree}_2 \Sigma \neq \mathbf{SafeRecSchTree}_2 \Sigma$

Collapsible HOPDA

Collapsible HOPDA is an extension of a HOPDA

Elements of 1-stack are tuples $(a,n_1,...,n_k)$, where $a \in \Sigma$, $n_i \in \mathbb{N}$.

$push_1 a$ - push $(a,n_1,...,n_k)$ on the top of the topmost order 1 stack,
where n_i is the size of the topmost order i stack

$collapse_i$ - if the topmost stack symbol is $(a,n_1,...,n_k)$
leave only first n_i-1 elements of the topmost order i stack

Notice: $collapse_1 = pop_1$

Example: Urzyczyn's language U

alphabet: [,], *

U contains words of the form:

[...[...[[...]...[...] * ... *

A B C

- segment A is a prefix of a well-bracketed word that ends in [which 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?

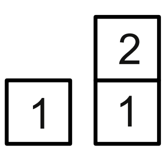
- one stack symbol
- first order stack counts the number of currently open brackets
- a copy ($push_2$) is done after each bracket

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



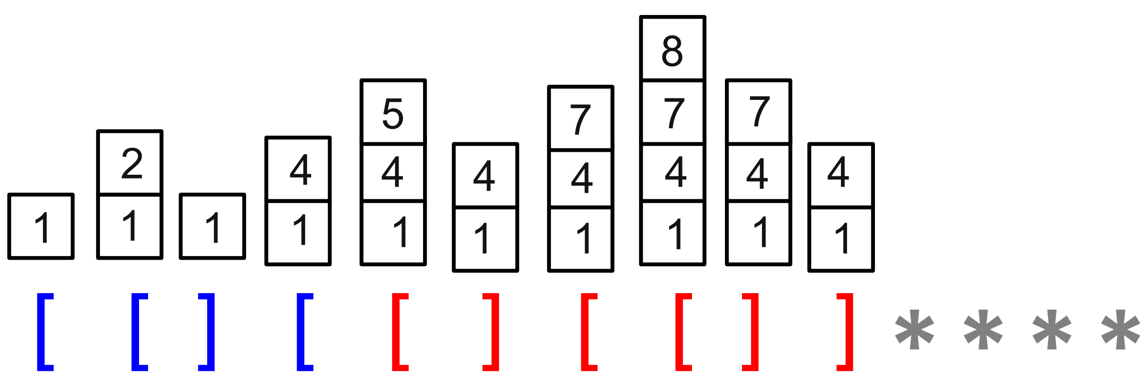
[[ ] [[ ] [[ ]] * * * *

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

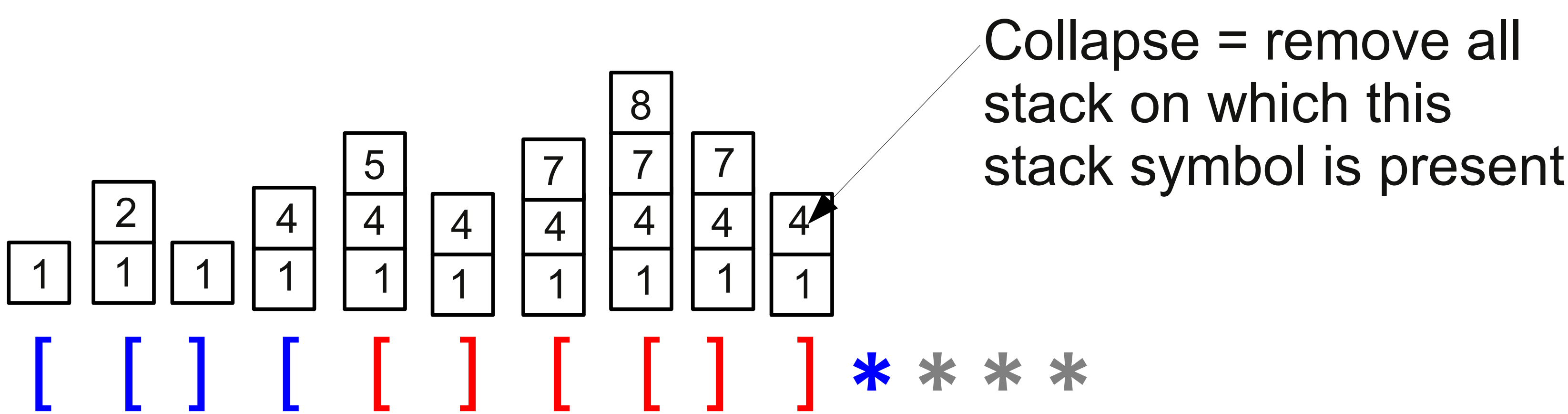
How to recognize U by an automaton with collapse?

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



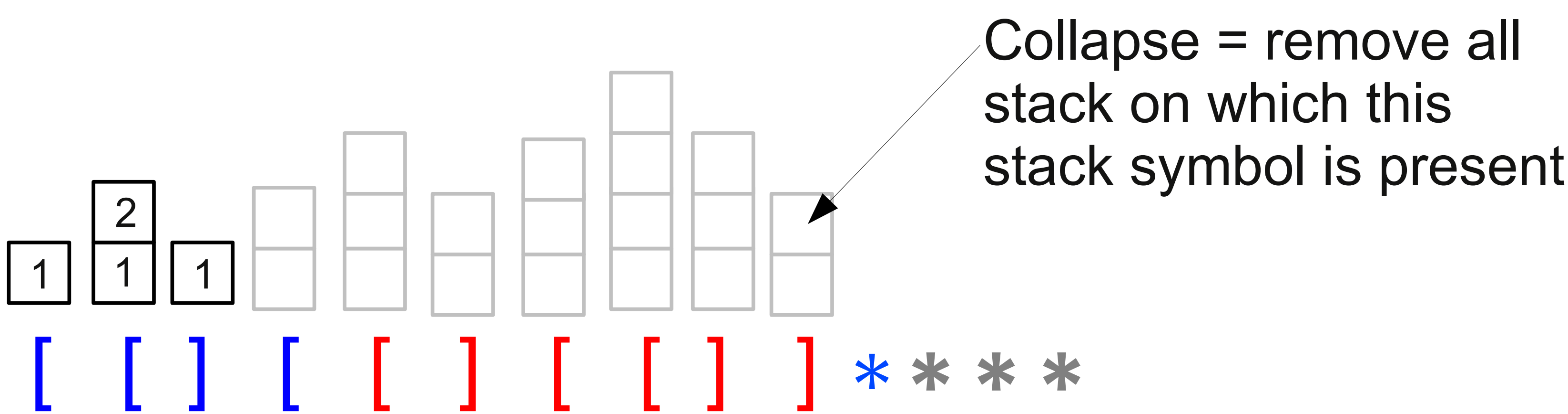
How to recognize U by an automaton with collapse?

- 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



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



Two hierarchies (of trees / of word languages):

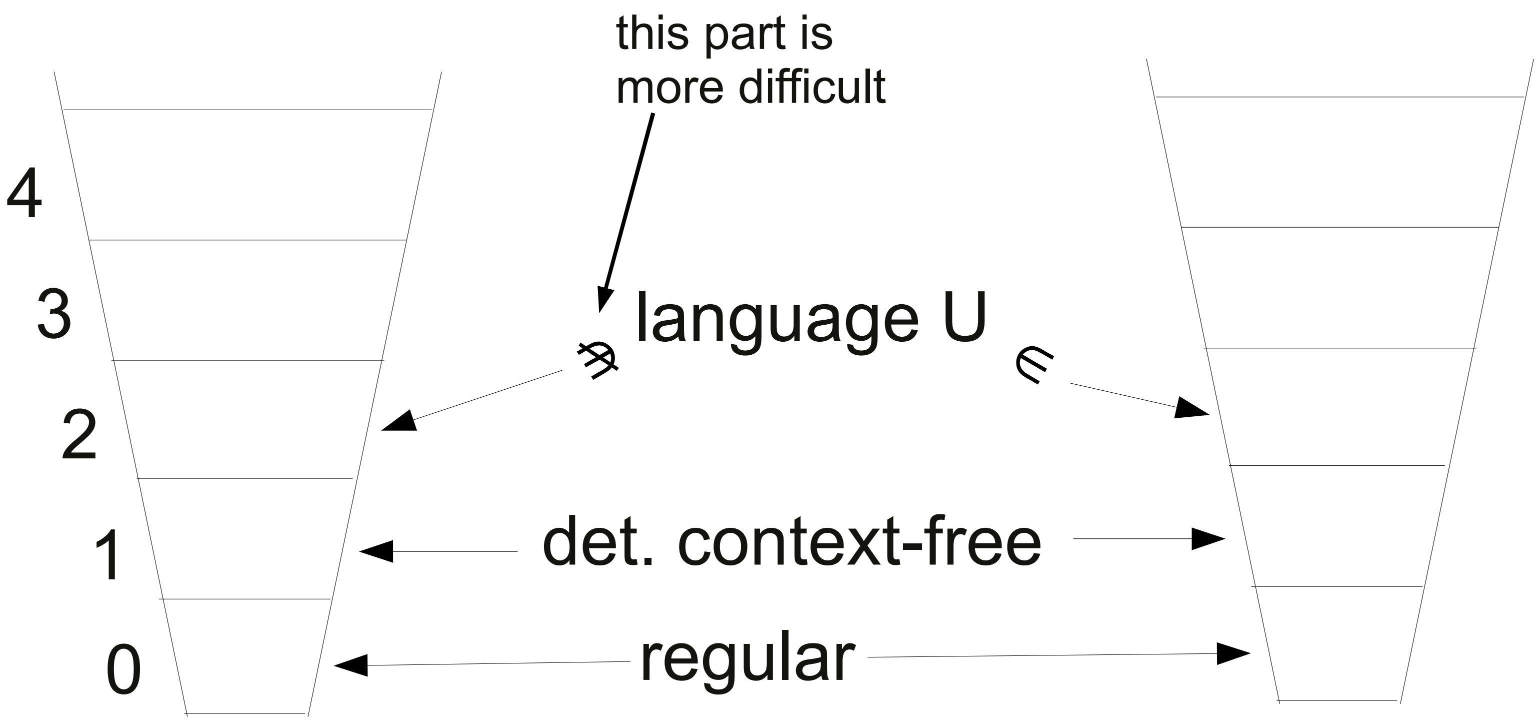
deterministic H-O pushdown automata

deterministic collapsible H-O pushdown automata

safe H-O schemas

H-O schemas

Caucal hierarchy



Open problems

1) Show that U (or some other language) is not accepted by a deterministic HOPDA (without collapse) of an arbitrary order, i.e. that the union of the whole hierarchies are different.

Open problems

1) Show that U (or some other language) is not accepted by a deterministic HOPDA (without collapse) of an arbitrary order, i.e. that the union of the whole hierarchies are different.

2) Does collapse increase recognizing power of **nondeterministic** HOPDA?

Aehlig, Miranda, Ong 2005: for level 2 – NO (collapse can be simulated by nondeterminism)

but:
- nondeterministic automata does not have a natural connection with verification
- most problems are undecidable, even universality for level-1 PDA (but emptiness is decidable)

Why U cannot be recognized without collapse?

Assume there is an order-2 HOPDA A recognizing U.

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

Lemma 1. We may assume that A does not use pop_2 before first star.

Lemma 2. Automaton A after reading u_n has at most C symbols on the last 1-stack.

Why U cannot be recognized without collapse?

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

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.

Lemma 1. We may assume that A does not use pop_2 before first star.

Lemma 2. Automaton A after reading u_n has at most C symbols on the last 1-stack.

Why U cannot be recognized without collapse?

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

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.

Lemma 1. We may assume that A does not use pop_2 before first star.

Lemma 2. Automaton A after reading u_n has at most C symbols on the last 1-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.
   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).

Lemma 2. Automaton A after reading u_n has at most C symbols on the last 1-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.

   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