Boundedness and Coverability for Pushdown Vector Addition Systems

Grégoire Sutre

LaBRI, CNRS & University of Bordeaux, France

ACTS, CMI, Chennai — February 2017

Based on joint works with J. Leroux, M. Praveen and P. Totzke.

Table of Contents

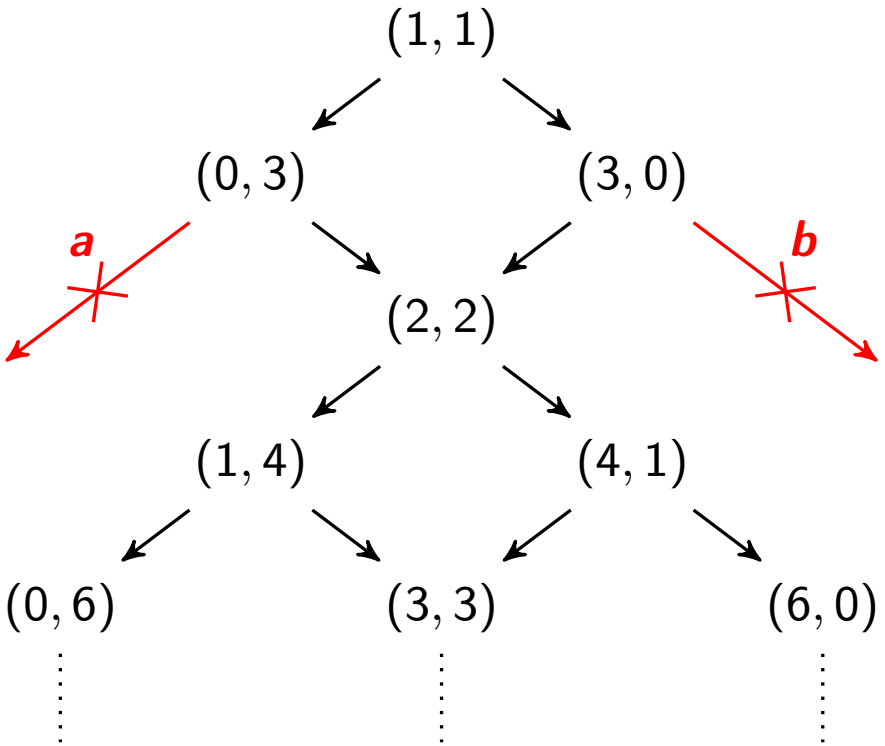
Table of Contents

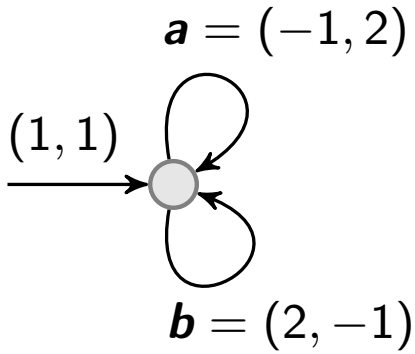
Vector Addition Systems

Definition

A VAS is a finite set of vectors $\boldsymbol{a} \in \mathbb{Z}^d$. For $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^d$ it has a step

$$\boldsymbol{u} \xrightarrow{\boldsymbol{a}} \boldsymbol{v} \quad \text{if} \quad \boldsymbol{v} = \boldsymbol{u} + \boldsymbol{a}.$$





Vector Addition Systems

Definition

A VAS is a finite set of vectors $\boldsymbol{a} \in \mathbb{Z}^d$. For $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^d$ it has a step

$$\boldsymbol{u} \xrightarrow{\boldsymbol{a}} \boldsymbol{v} \quad \text{if} \quad \boldsymbol{v} = \boldsymbol{u} + \boldsymbol{a}.$$

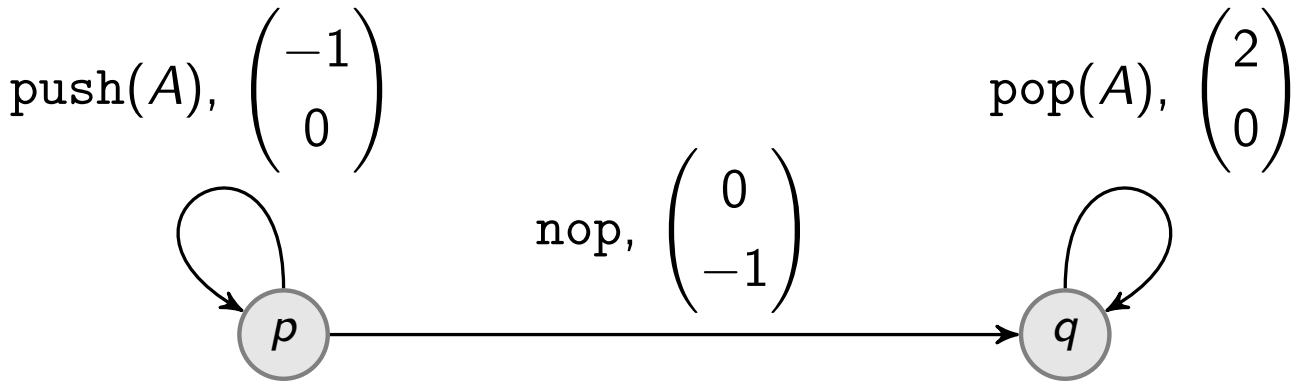
Equivalent to Petri nets

Many decidable verification questions

- Reachability: does $\boldsymbol{u} \xrightarrow{*} \boldsymbol{v}$?
- Coverability: does there exist $\boldsymbol{v}' \geq \boldsymbol{v}$ such that $\boldsymbol{u} \xrightarrow{*} \boldsymbol{v}'$?
- Boundedness: is $\{\boldsymbol{v} \mid \boldsymbol{u} \xrightarrow{*} \boldsymbol{v}\}$ finite ?
- ...

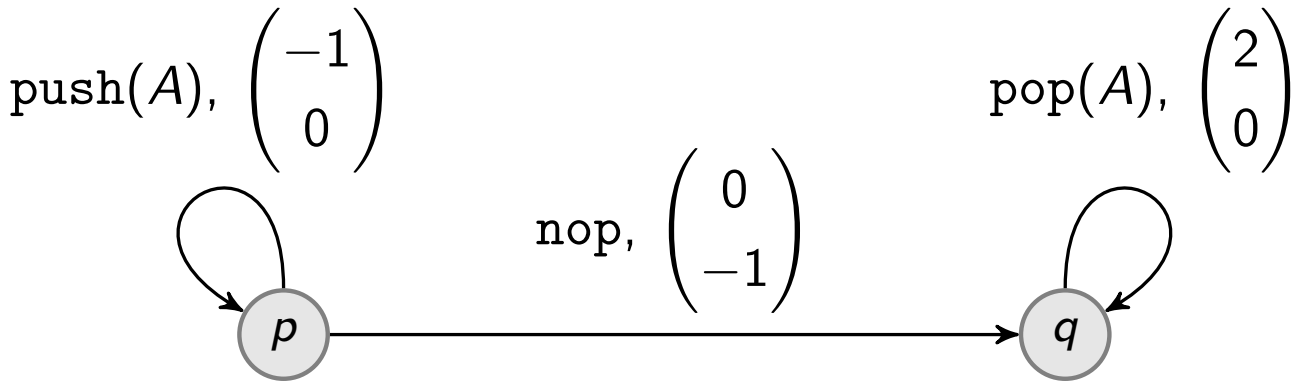
Pushdown Vector Addition Systems

. . . are products of VAS with pushdown automata.



Pushdown Vector Addition Systems

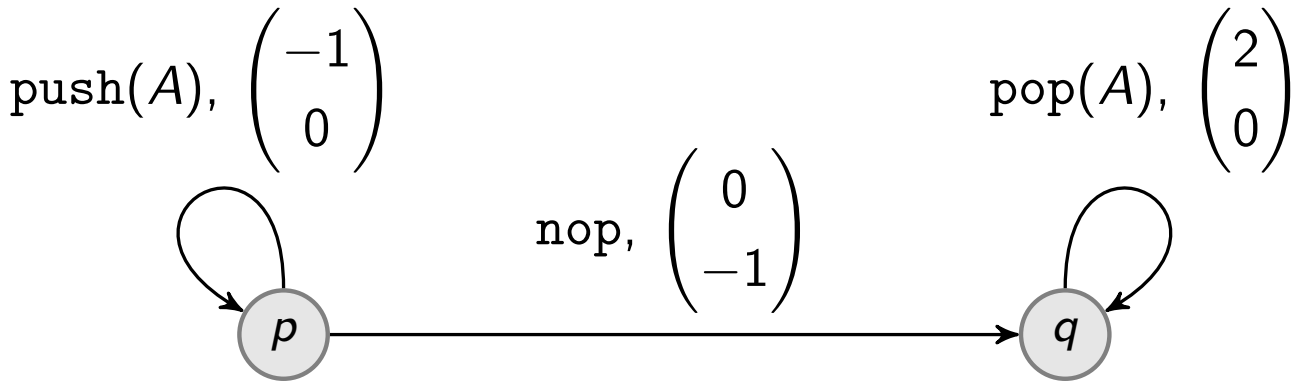
. . . are products of VAS with pushdown automata.



$p, \bot, \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

Pushdown Vector Addition Systems

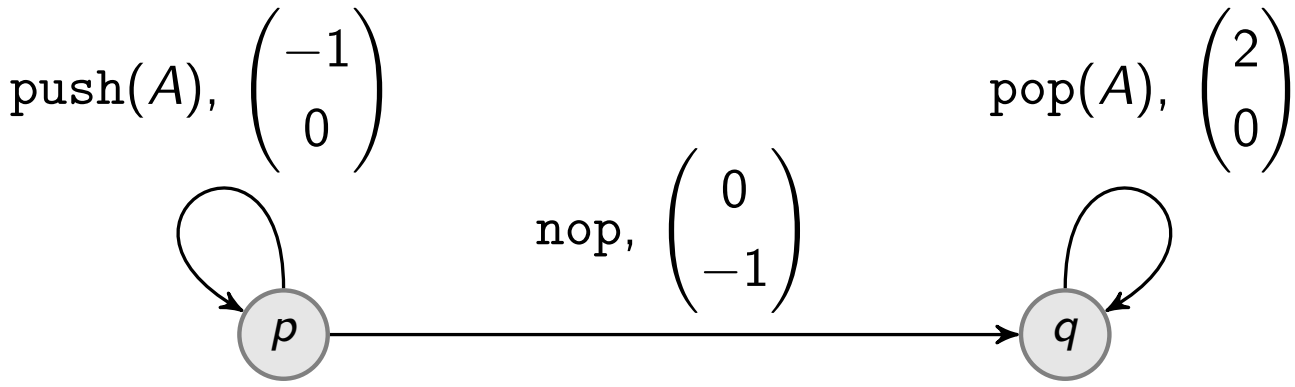
. . . are products of VAS with pushdown automata.



$$p, \bot, \begin{pmatrix}2\\1\end{pmatrix} \longrightarrow\longrightarrow p, AA\bot, \begin{pmatrix}0\\1\end{pmatrix}$$

Pushdown Vector Addition Systems

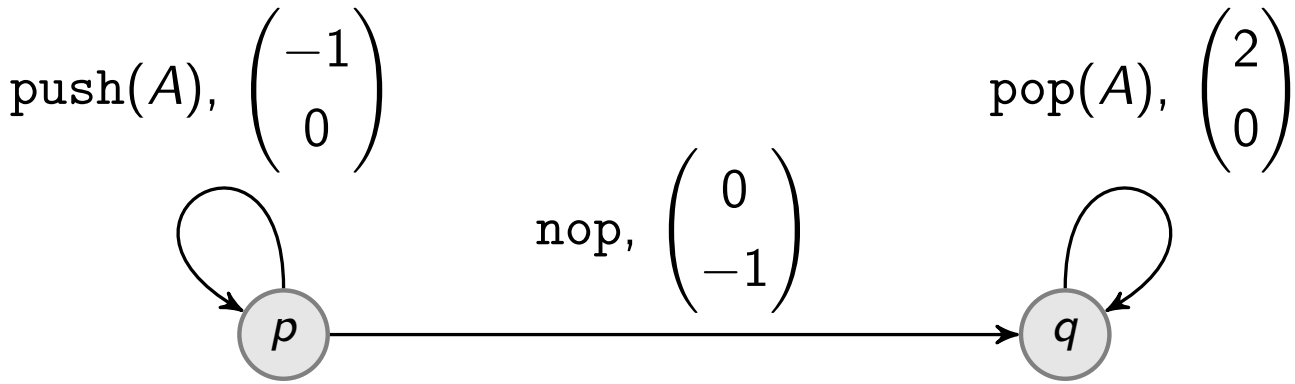
. . . are products of VAS with pushdown automata.



$$p, \bot, \begin{pmatrix}2\\1\end{pmatrix} \longrightarrow\longrightarrow p, AA\bot, \begin{pmatrix}0\\1\end{pmatrix} \longrightarrow q, AA\bot, \begin{pmatrix}0\\0\end{pmatrix}$$

Pushdown Vector Addition Systems

. . . are products of VAS with pushdown automata.



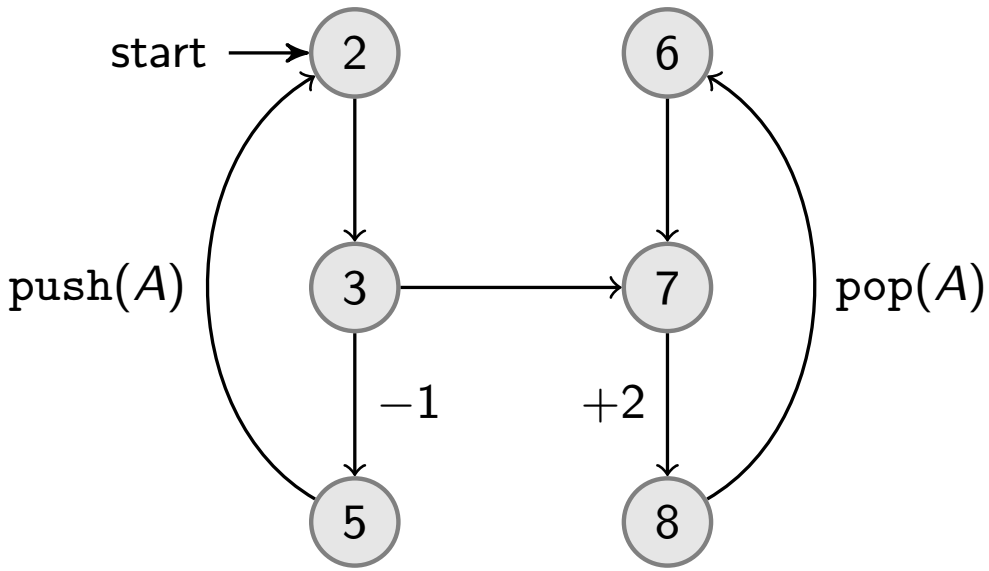
$$p, \bot, \begin{pmatrix}2\\1\end{pmatrix} \longrightarrow\longrightarrow p, AA\bot, \begin{pmatrix}0\\1\end{pmatrix} \longrightarrow q, AA\bot, \begin{pmatrix}0\\0\end{pmatrix} \longrightarrow\longrightarrow q, \bot, \begin{pmatrix}4\\0\end{pmatrix}$$

Pushdown Vector Addition Systems

. . . are products of VAS with pushdown automata.

They can for example model recursive programs with variables over $\mathbb{N}$.

```
1: x ← n
2: procedure DoubleX
3:     if (⋆ ∧ x > 0) then
4:         x ← (x − 1)
5:         DoubleX
6:     end if
7:     x ← (x + 2)
8: end procedure
```



Pushdown Vector Addition Systems — Definition

Definition

A pushdown VAS is a triple $\langle Q, \Gamma, \Delta \rangle$ where

- Q : finite set of states
- Γ : finite stack alphabet
- $\Delta \subseteq Q \times (\mathtt{Op} \times \mathbb{Z}^d) \times Q$: finite set of transitions, with

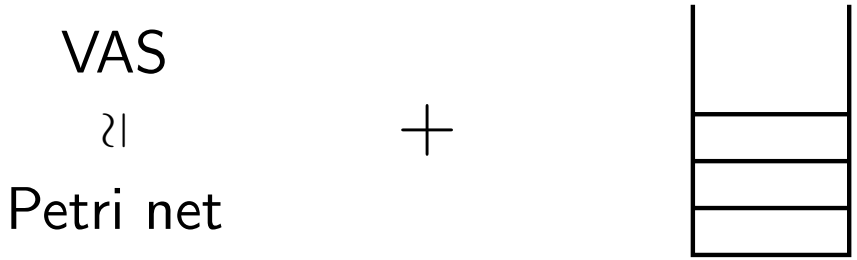
$$\mathtt{Op} = \{\mathtt{nop}\} \cup \{\mathtt{push}(\gamma), \mathtt{pop}(\gamma) \mid \gamma \in \Gamma\}$$

Configurations: $(q, \sigma, \boldsymbol{v})$ with $q \in Q$, $\sigma \in \Gamma^*$ and $\boldsymbol{v} \in \mathbb{N}^d$

Steps: as expected

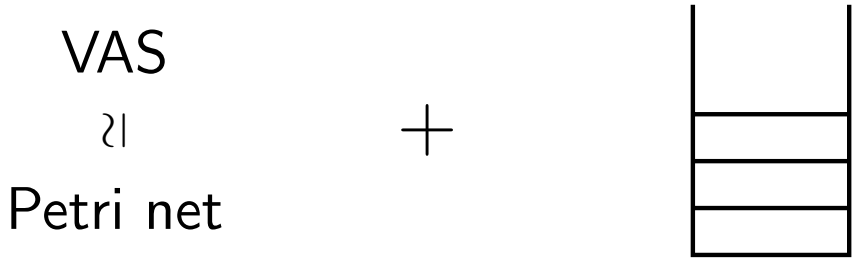
- Reachability: does $(p, \varepsilon, \boldsymbol{u}) \xrightarrow{*} (q, \varepsilon, \boldsymbol{v})$?
- Coverability: does there exist $\boldsymbol{v}' \geq \boldsymbol{v}$ with $(p, \varepsilon, \boldsymbol{u}) \xrightarrow{*} (q, \varepsilon, \boldsymbol{v}')$?
- Boundedness: is $\{(q, \sigma, \boldsymbol{v}) \mid (p, \varepsilon, \boldsymbol{u}) \xrightarrow{*} (q, \sigma, \boldsymbol{v})\}$ finite ?

Pushdown Vector Addition Systems — Motivations



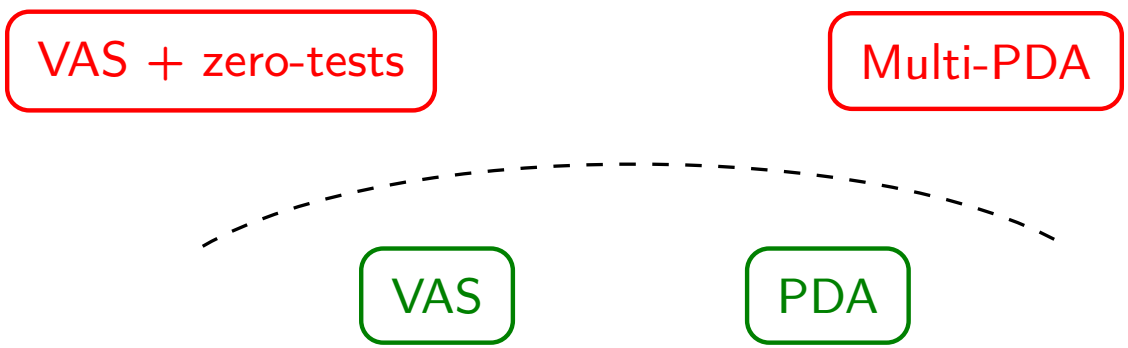
- Richer model for the verification of concurrent systems
  - Multi-threaded recursive programs
  - One recursive server + unboundedly many finite-state clients

Pushdown Vector Addition Systems — Motivations



- Richer model for the verification of concurrent systems
  - Multi-threaded recursive programs
  - One recursive server + unboundedly many finite-state clients

- Is the model too powerful?



Brief State of the Art

	Boundedness	Coverability	Reachability
VAS	ExpSpace-c[1,2]	ExpSpace-c[1,2]	Decidable[3,4,5]
+ full counter	Decidable[7]	Decidable[6]	
+ stack	Decidable[9]	Tower-h[8]	
1-VAS + stack	ExpTime-e[11]	Decidable[10]	?

[1] Lipton 1976
[2] Rackoff 1978
[3] Mayr 1981
[4] Kosaraju 1982
[5] Leroux, Schmitz 2015
[6] Reinhardt 2008
[7] Finkel, Sangnier 2010
[8] Lazić 2012
[9] Leroux, Praveen, S. 2014
[10] Leroux, S., Totzke 2015
[11] Leroux, S., Totzke 2015

Brief State of the Art

	Boundedness	Coverability	Reachability
VAS	ExpSpace-c[1,2]	ExpSpace-c[1,2]	Decidable[3,4,5]
+ full counter	Decidable[7]	Decidable[6]	
+ stack	Decidable[9]	Tower-h[8]	
1-VAS + stack	ExpTime-e[11]	Decidable[10]	?

Subclasses of pushdown VAS with decidable reachability

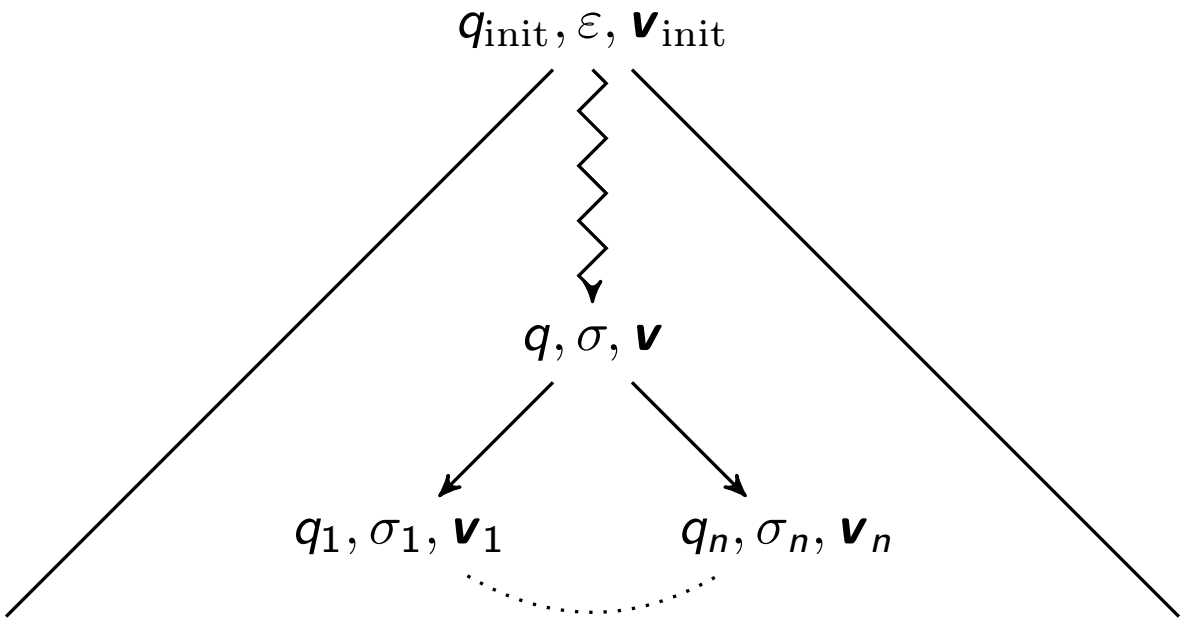
- Multiset pushdown systems [Sen, Viswanathan 2006]
- VAS $\cap$ CFL of finite index [Atig, Ganty 2011]

Related decidable models with counters and recursion

- BPA($\mathbb{Z}$) [Bouajjani, Habermehl, Mayr 2003]

Table of Contents

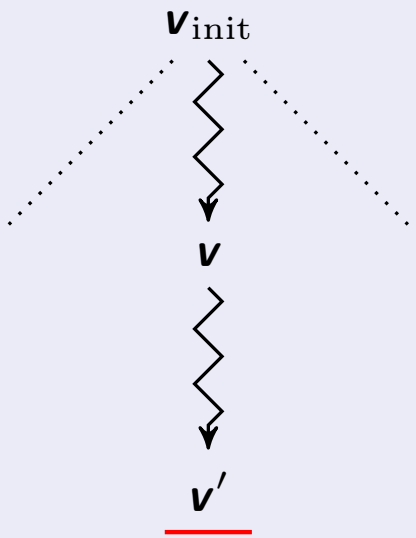
Reachability Tree of a Pushdown VAS



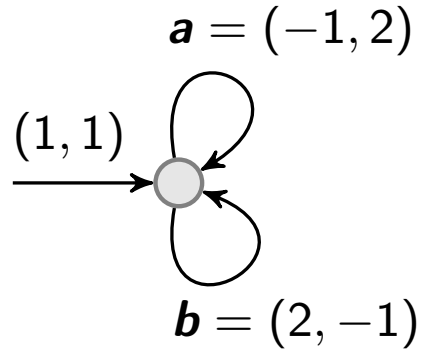
- Exhaustive and enumerative forward exploration from $(q_{\text{init}}, \varepsilon, \boldsymbol{v}_{\text{init}})$
- Potentially infinite, need to truncate

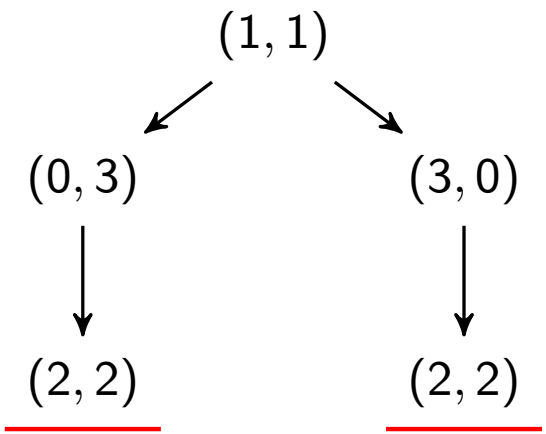
Reduced Reachability Tree for VAS [Karp, Miller 1969]

Truncation Rule



if $v \leq v'$

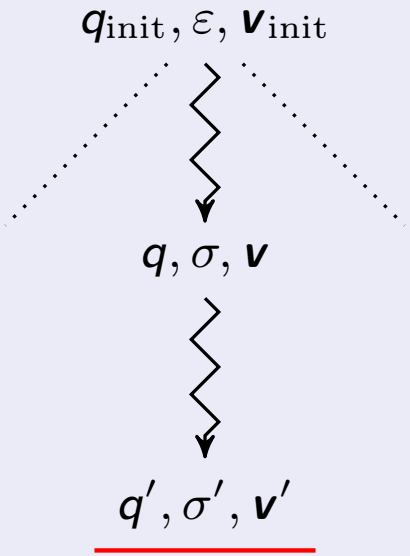




- The reduced reachability tree is finite
- It contains enough information to decide boundedness
- Crucial ingredient: the strict order $<$ is a simulation relation

Tentative Simulation-Based Truncation for Pushdown VAS

Truncation Rule



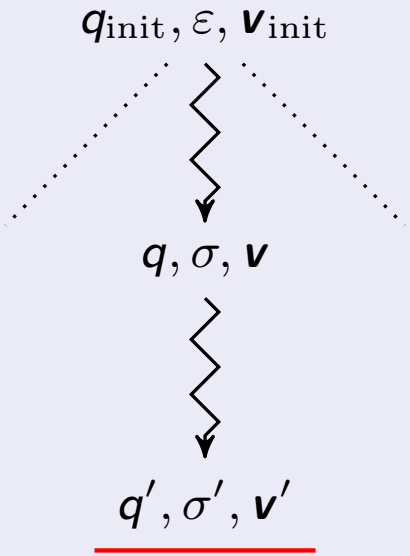
if $q = q'$, $\boldsymbol{v} \leq \boldsymbol{v}'$ and $\sigma \leq_{\text{prefix}} \sigma'$

➪ No loss of information to decide boundedness

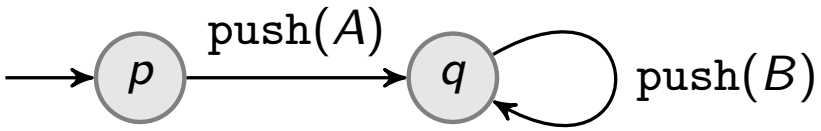
But...

Tentative Simulation-Based Truncation for Pushdown VAS

Truncation Rule



if $q = q'$, $\boldsymbol{v} \leq \boldsymbol{v}'$ and $\sigma \leq_{\text{prefix}} \sigma'$



$(p, \bot)$

$(q, A\bot)$

$(q, BA\bot)$

$(q, BBA\bot)$

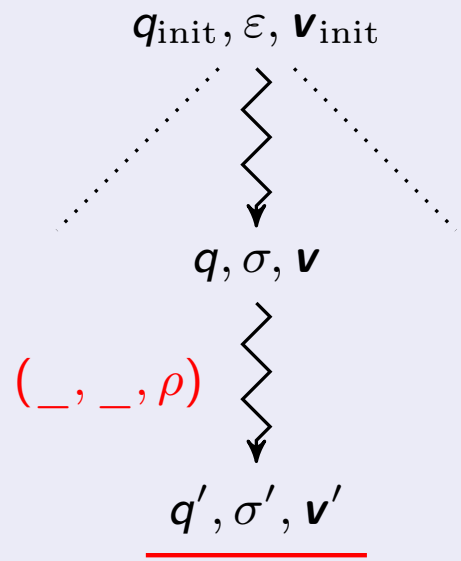
⇨ No loss of information to decide boundedness

But...

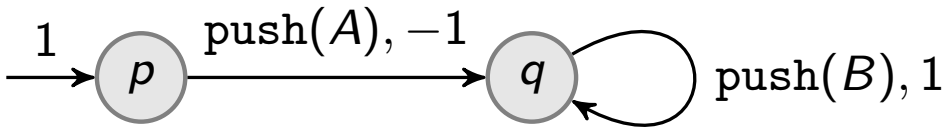
The reduced reachability tree may be infinite!

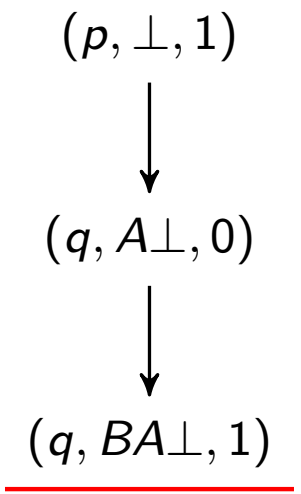
Reduced Reachability Tree for Pushdown VAS

Truncation Rule



if $\begin{cases} q = q' \text{ and } \boldsymbol{v} \leq \boldsymbol{v}' \\ \sigma \leq_{\text{suffix}} \rho \text{ for all } \rho \end{cases}$





- The reduced reachability tree is finite
- It contains enough information to decide boundedness

Finiteness of the Reduced Reachability Tree

Proposition

The reduced reachability tree of a pushdown VAS is finite.

Proof. By contradiction, assume that it is infinite.

The tree is finitely branching. So, by König's Lemma, there is an infinite branch

$$(q_{\text{init}}, \varepsilon, \boldsymbol{v}_{\text{init}}) \to (q_1, \sigma_1, \boldsymbol{v}_1) \to (q_2, \sigma_2, \boldsymbol{v}_2) \cdots$$

Finiteness of the Reduced Reachability Tree

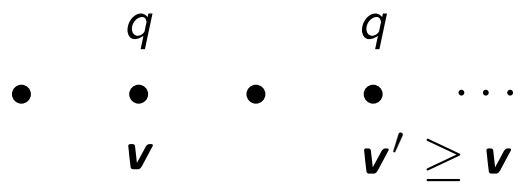
Proposition

The reduced reachability tree of a pushdown VAS is finite.

Proof. By contradiction, assume that it is infinite.

The tree is finitely branching. So, by König's Lemma, there is an infinite branch

$$(q_{\text{init}}, \varepsilon, \boldsymbol{v}_{\text{init}}) \to (q_1, \sigma_1, \boldsymbol{v}_1) \to (q_2, \sigma_2, \boldsymbol{v}_2) \cdots$$



Finiteness of the Reduced Reachability Tree

Proposition

The reduced reachability tree of a pushdown VAS is finite.

Proof. By contradiction, assume that it is infinite.

The tree is finitely branching. So, by König's Lemma, there is an infinite branch

$$(q_{\text{init}}, \varepsilon, \boldsymbol{v}_{\text{init}}) \to (q_1, \sigma_1, \boldsymbol{v}_1) \to (q_2, \sigma_2, \boldsymbol{v}_2) \cdots$$

Finiteness of the Reduced Reachability Tree

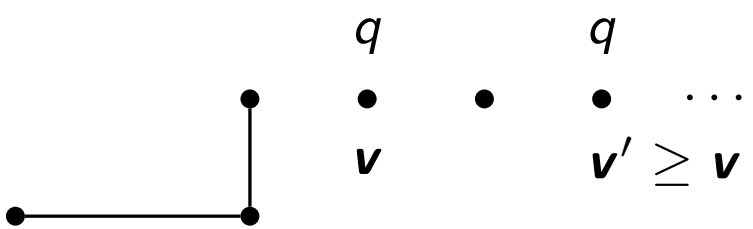
Proposition

The reduced reachability tree of a pushdown VAS is finite.

Proof. By contradiction, assume that it is infinite.

The tree is finitely branching. So, by König's Lemma, there is an infinite branch

$$(q_{\text{init}}, \varepsilon, \boldsymbol{v}_{\text{init}}) \to (q_1, \sigma_1, \boldsymbol{v}_1) \to (q_2, \sigma_2, \boldsymbol{v}_2) \cdots$$



Finiteness of the Reduced Reachability Tree

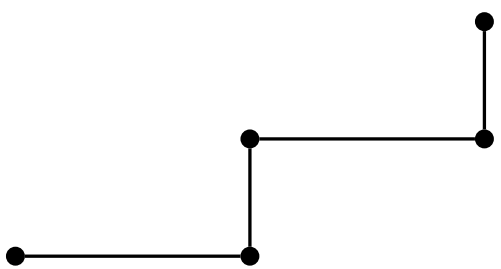
Proposition

The reduced reachability tree of a pushdown VAS is finite.

Proof. By contradiction, assume that it is infinite.

The tree is finitely branching. So, by König's Lemma, there is an infinite branch

$$(q_{\text{init}}, \varepsilon, \boldsymbol{v}_{\text{init}}) \to (q_1, \sigma_1, \boldsymbol{v}_1) \to (q_2, \sigma_2, \boldsymbol{v}_2) \cdots$$



Finiteness of the Reduced Reachability Tree

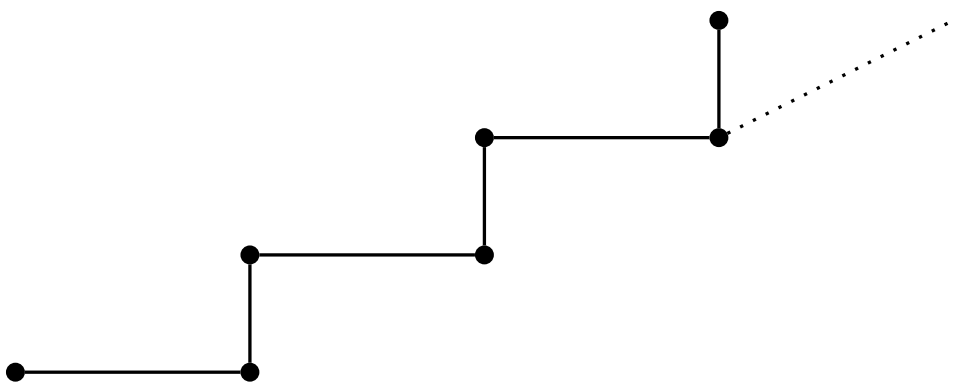
Proposition

The reduced reachability tree of a pushdown VAS is finite.

Proof. By contradiction, assume that it is infinite.

The tree is finitely branching. So, by König's Lemma, there is an infinite branch

$$(q_{\text{init}}, \varepsilon, \boldsymbol{v}_{\text{init}}) \to (q_1, \sigma_1, \boldsymbol{v}_1) \to (q_2, \sigma_2, \boldsymbol{v}_2) \cdots$$



Finiteness of the Reduced Reachability Tree

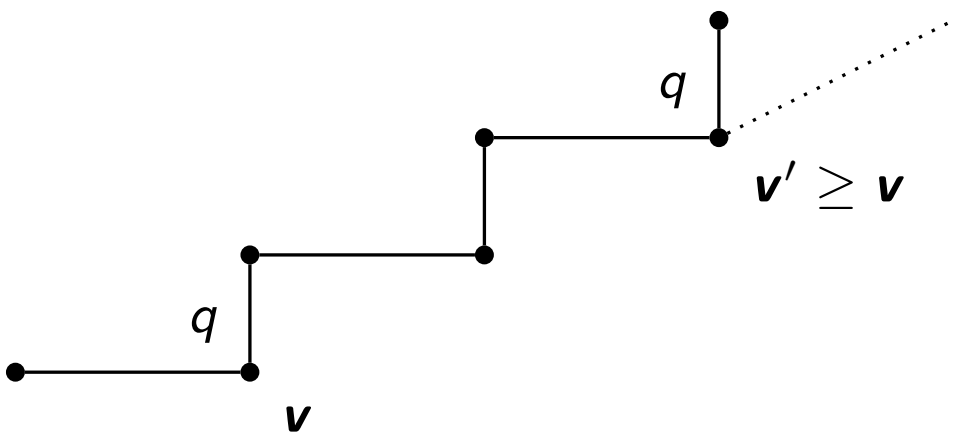
Proposition

The reduced reachability tree of a pushdown VAS is finite.

Proof. By contradiction, assume that it is infinite.

The tree is finitely branching. So, by König's Lemma, there is an infinite branch

$$(q_{\text{init}}, \varepsilon, \boldsymbol{v}_{\text{init}}) \to (q_1, \sigma_1, \boldsymbol{v}_1) \to (q_2, \sigma_2, \boldsymbol{v}_2) \cdots$$



□

RRT-based Algorithm for Pushdown VAS Boundedness

Proposition

A pushdown VAS is unbounded iff its reduced reachability tree contains

$$\underbrace{(q, \sigma, \boldsymbol{v}) \rightsquigarrow (q, \sigma', \boldsymbol{v}')}_{\sigma \text{ remains a suffix}}$$

such that $\boldsymbol{v} \leq \boldsymbol{v}'$ *and* $\sigma \leq_{\text{suffix}} \sigma'$*, and at least one of these inequalities is strict.*

Theorem ([Leroux, Praveen, S. 2014])

Boundedness is decidable for pushdown VAS.

Worst-Case Complexity of the Algorithm

How big can the reduced reachability tree be?

Worst-Case Complexity of the Algorithm

How big can the reduced reachability tree be?

Theorem ([Leroux, Praveen, S. 2014])

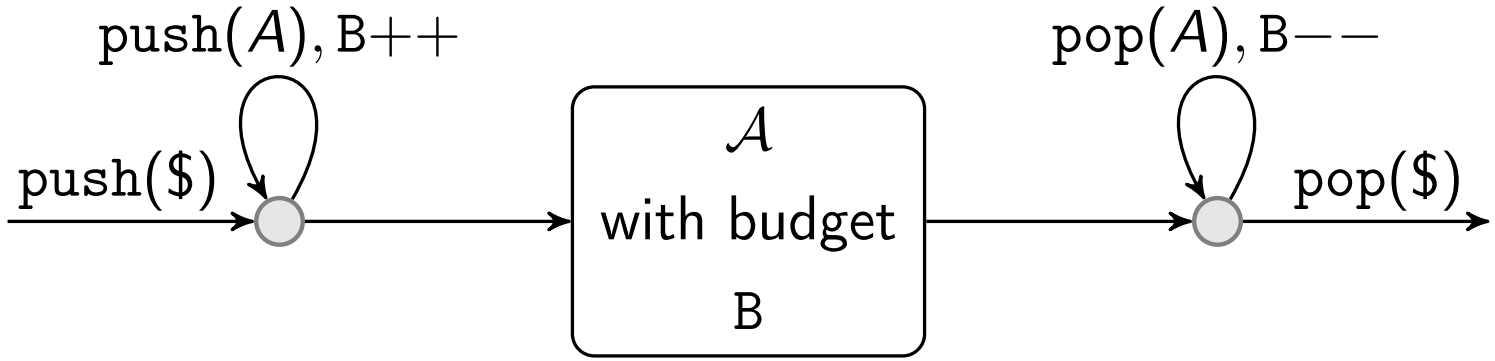
The reduced reachability tree of a pushdown VAS has at most an hyper-Ackermannian number of nodes, and this bound is tight.

Table of Contents

Observation ([Lazić 2012])

Reachability in dimension d reduces to Coverability in dimension $d+1$.

Proof. Budget construction. Use the stack to test the budget for zero. Add a new counter $\mathtt{B}$ and two new stack symbols $A, \$$.



$(q^{\mathcal{A}}_{\text{init}}, \varepsilon, \mathbf{0}) \xrightarrow{*} (q^{\mathcal{A}}_{\text{final}}, \varepsilon, \mathbf{0})$ iff $(q^{\mathcal{A}'}_{\text{init}}, \varepsilon, \mathbf{0}, 0) \xrightarrow{*} (q^{\mathcal{A}'}_{\text{final}}, \varepsilon, \_, \_)$ □

Coverability versus Reachability in Pushdown VAS

Observation ([Lazić 2012])

Reachability in dimension d reduces to Coverability in dimension $d + 1$.

$$\mathit{Reach}(0) \sqsubseteq \mathit{Cover}(1) \sqsubseteq \mathit{Reach}(1) \sqsubseteq \mathit{Cover}(2) \sqsubseteq \cdots$$

Coverability versus Reachability in Pushdown VAS

Observation ([Lazić 2012])

Reachability in dimension d reduces to Coverability in dimension $d + 1$.

$$\mathit{Reach}(0) \sqsubseteq \mathit{Cover}(1) \sqsupseteq \mathit{Reach}(1) \sqsubseteq \mathit{Cover}(2) \sqsubseteq \cdots$$

Theorem ([Leroux, S., Totzke 2015])

Coverability for 1-dimensional pushdown VAS is decidable.

Another Perspective

The coverability problem for d-dimensional pushdown VAS can be rephrased as follows.

Input:

- a VAS $\boldsymbol{A} \subseteq \mathbb{Z}^d$
- a context-free language $L \in \boldsymbol{A}^*$
- vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^d$

Output: whether there exist $\boldsymbol{a_1 a_2 \ldots a_k} \in L$ and $\boldsymbol{v'} \in \mathbb{N}^d$ such that

$$\boldsymbol{u} \xrightarrow{\boldsymbol{a_1}} \xrightarrow{\boldsymbol{a_2}} \cdots \xrightarrow{\boldsymbol{a_k}} \boldsymbol{v'} \quad \text{and} \quad \boldsymbol{v'} \geq \boldsymbol{v}$$

Grammar-Controlled Vector Addition Systems

A context-free grammar is a triple $G = (V, A, R)$ where

- V : nonterminal symbols (variables)
- A : terminal symbols
- R : production rules $X \vdash \alpha$ where $X \in V$ and $\alpha \in (V \cup A)^*$

Definition (1-dimensional GVAS)

A GVAS is a context-free grammar $G = (V, A, R)$ such that $A \subseteq \mathbb{Z}$.

Every GVAS can be transformed into an equivalent one where

- all variables $X \in V$ are productive
- $A = \{-1, 0, 1\}$

Summaries for Coverability

A GVAS is a context-free grammar $G = (V, A, R)$ such that $A \subseteq \mathbb{Z}$.

Notations:

$$\begin{aligned} L_X &= \{a_1 \cdots a_k \in A^* \mid X \stackrel{*}{\Longrightarrow} a_1 \cdots a_k\} \\ c \xrightarrow{X} d &\Leftrightarrow c \xrightarrow{a_1} \cdots \xrightarrow{a_k} d \text{ for some } a_1 \cdots a_k \in L_X \end{aligned}$$

Definition (Summary of a Variable)

$$\mathit{Summary}_X(c) = \sup\{d \mid c \xrightarrow{X} d\}$$

Coverability: $\mathit{Summary}_S(c) \geq d$? (given S, c, d)

Example: Weak Computation of Multiplication by Two

$$S \vdash -1\, S\, 1\, 1 \mid \varepsilon$$

$$L_S = \{(-1)^n(11)^n \mid n \in \mathbb{N}\}$$

For every $c, d \in \mathbb{N}$,

$$\begin{aligned} c \xrightarrow{S} d &\iff \exists n \in \mathbb{N} : c \xrightarrow{(-1)^n(11)^n} d \\ &\iff \exists n \leq c : c \xrightarrow{(-1)^n} c - n \xrightarrow{(11)^n} c + n = d \\ &\iff c \leq d \leq 2c \end{aligned}$$

$$\mathit{Summary}_S(c) = 2c$$

Example: Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$

Example: Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$

$$\begin{aligned} A_0(n) &= n+1 \\ A_1(n) &= n+2 \\ A_2(n) &= 2n+3 \\ A_3(n) &= 2^{n+3} - 3 \\ &\vdots \end{aligned}$$

Example: Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$

$$\begin{aligned} A_0(n) &= n+1 \\ A_1(n) &= n+2 \\ A_2(n) &= 2n+3 \\ A_3(n) &= 2^{n+3} - 3 \\ &\vdots \end{aligned}$$

$$\begin{aligned} X_0 &\vdash \mathbf{1} \\ X_1 &\vdash -\mathbf{1}\ X_1X_0 \mid \mathbf{1}\ X_0 \\ X_2 &\vdash -\mathbf{1}\ X_2X_1 \mid \mathbf{1}\ X_1 \\ &\vdots \\ X_m &\vdash -\mathbf{1}\ X_mX_{m-1} \mid \mathbf{1}\ X_{m-1} \end{aligned}$$

Example: Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$

$$\begin{aligned} A_0(n) &= n+1 \\ A_1(n) &= n+2 \\ A_2(n) &= 2n+3 \\ A_3(n) &= 2^{n+3}-3 \\ &\vdots \end{aligned}$$

$$\begin{aligned} X_0 &\vdash \mathbf{1} \\ X_1 &\vdash -\mathbf{1}\ X_1X_0 \mid \mathbf{1}\ X_0 \\ X_2 &\vdash -\mathbf{1}\ X_2X_1 \mid \mathbf{1}\ X_1 \\ &\vdots \\ X_m &\vdash -\mathbf{1}\ X_mX_{m-1} \mid \mathbf{1}\ X_{m-1} \end{aligned}$$

$$\begin{aligned} X_m &\stackrel{*}{\Longrightarrow} -\mathbf{1}^n X_m X_{m-1}^n \\ &\Longrightarrow -\mathbf{1}^n \mathbf{1} X_{m-1}^{n+1} \\ &\stackrel{*}{\Longrightarrow} \cdots \end{aligned}$$

$$A_m = \mathit{Summary}_{X_m}$$

Flow Trees

Certificates for $\mathit{Summary}_S(c) \geq d$? Annotated parse trees!

Flow Trees

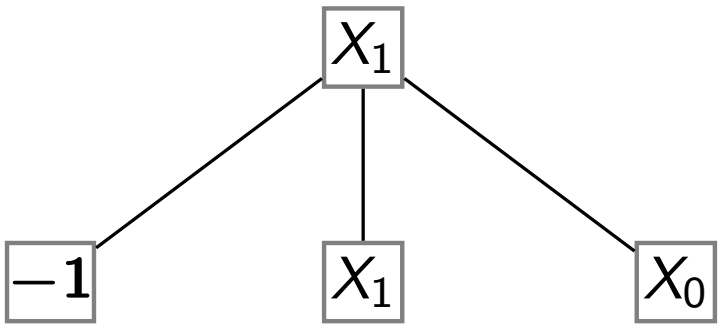
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!

$\boxed{X_1}$

$(Summary_{X_1}(5) \geq 3)$

Flow Trees

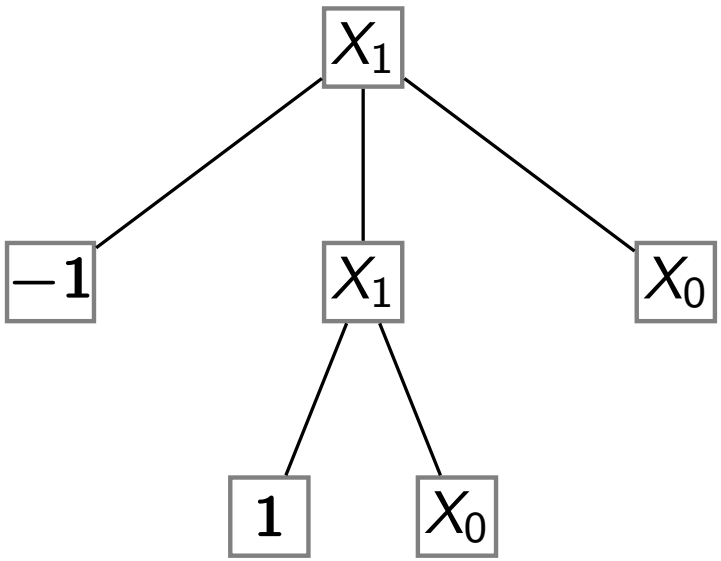
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

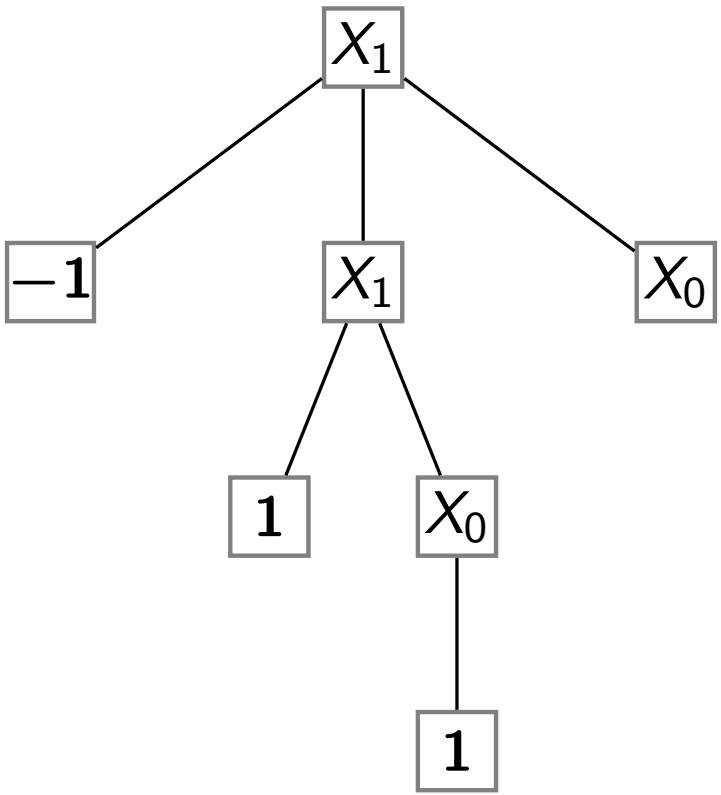
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

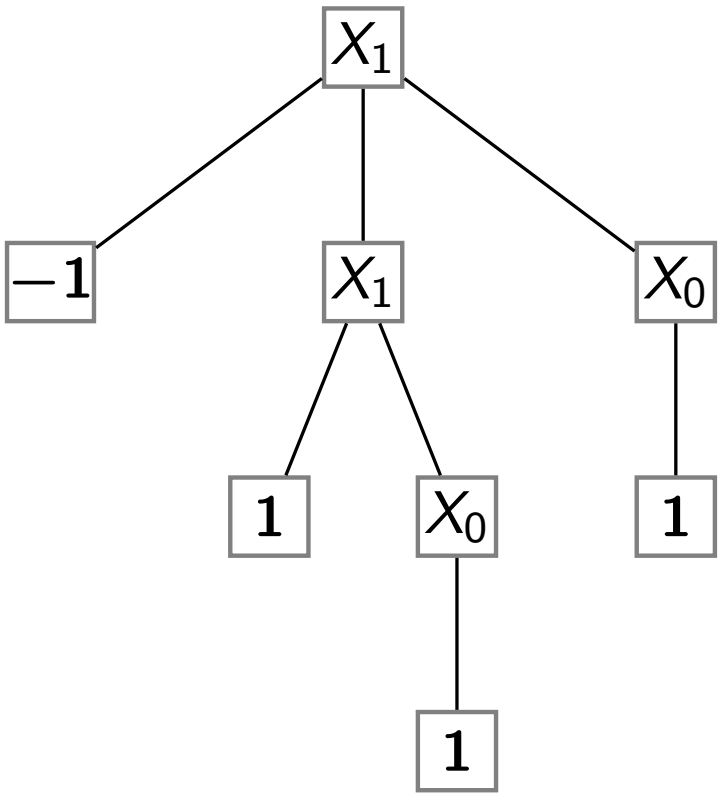
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

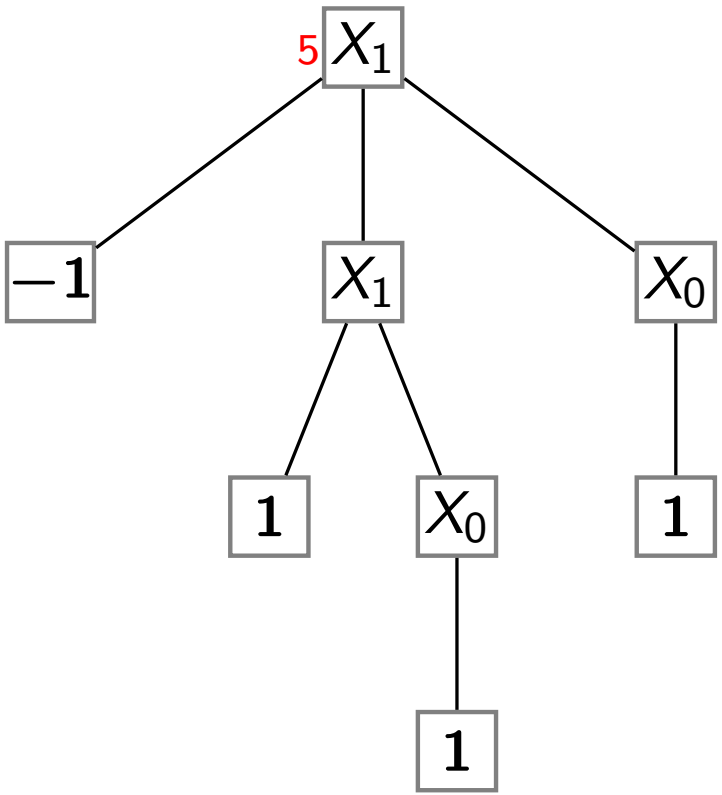
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

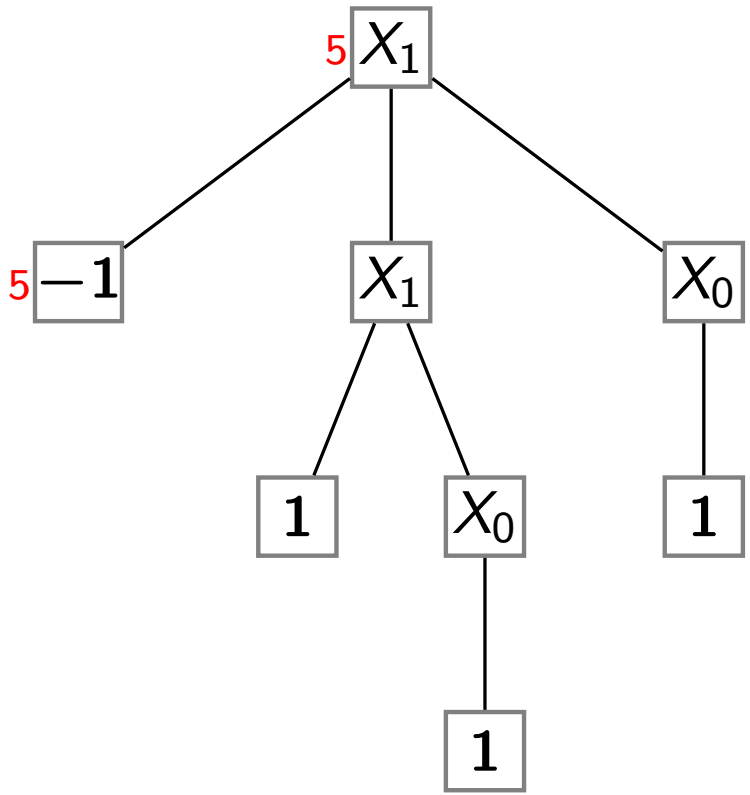
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

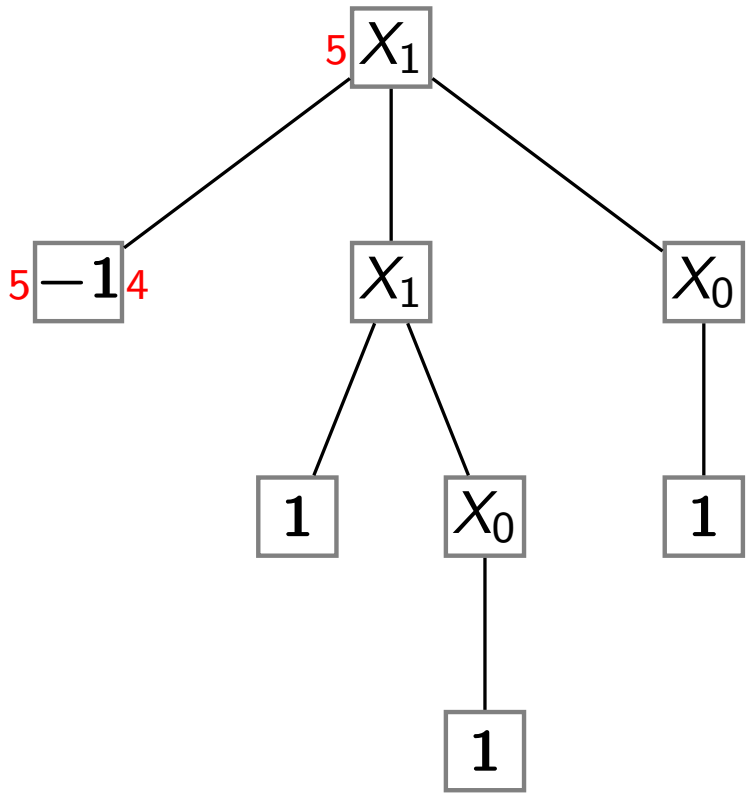
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

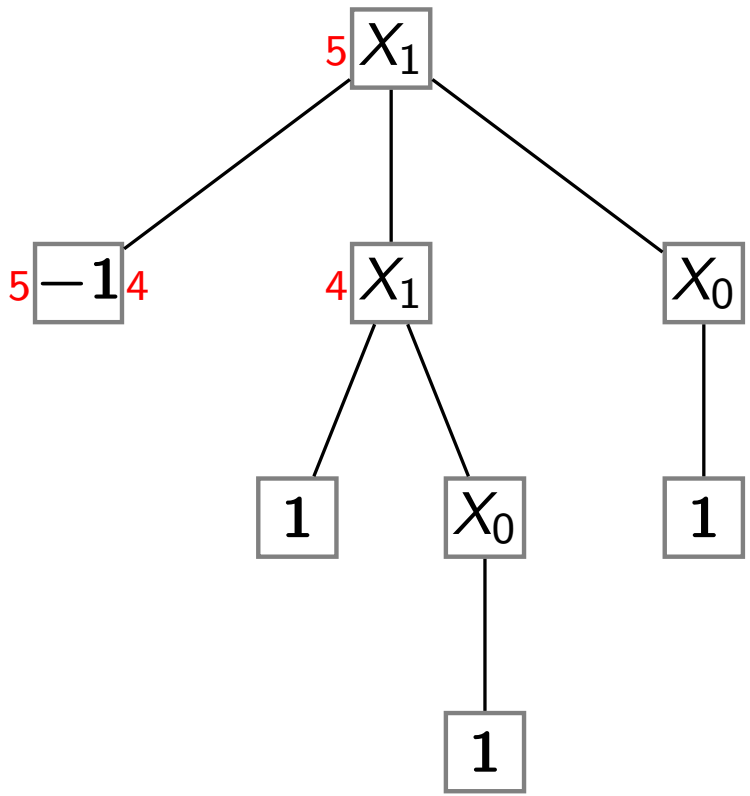
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

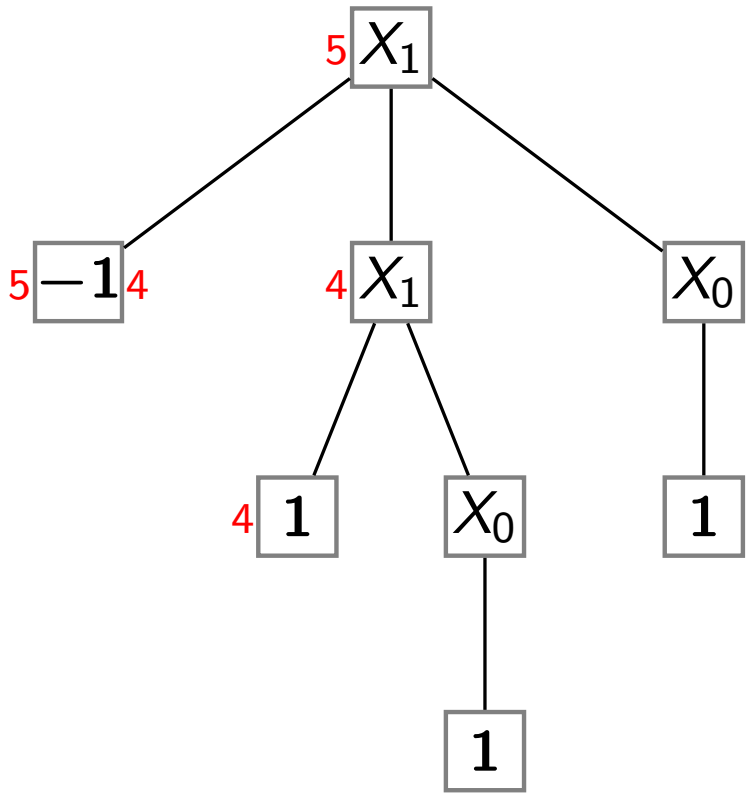
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

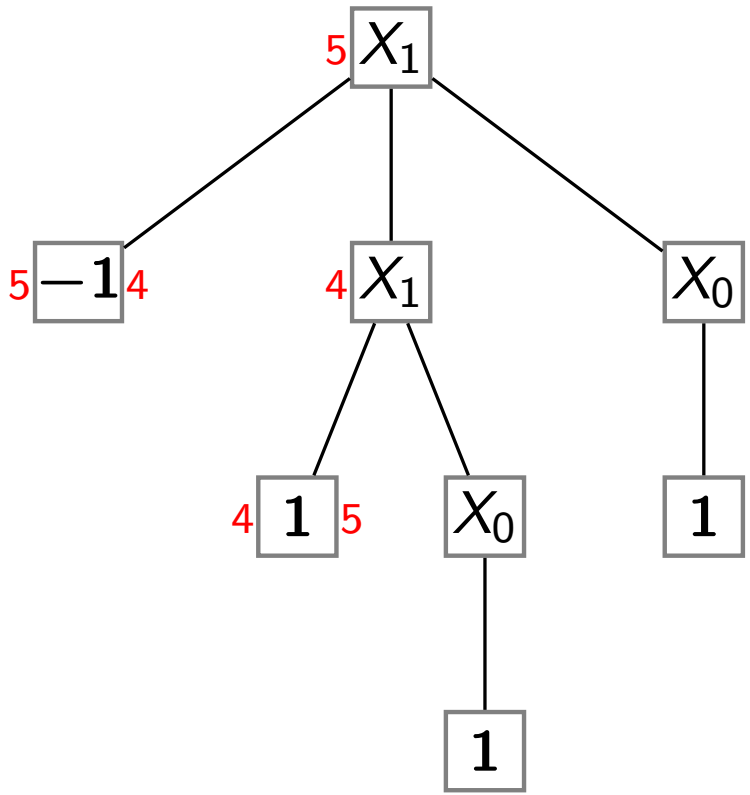
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

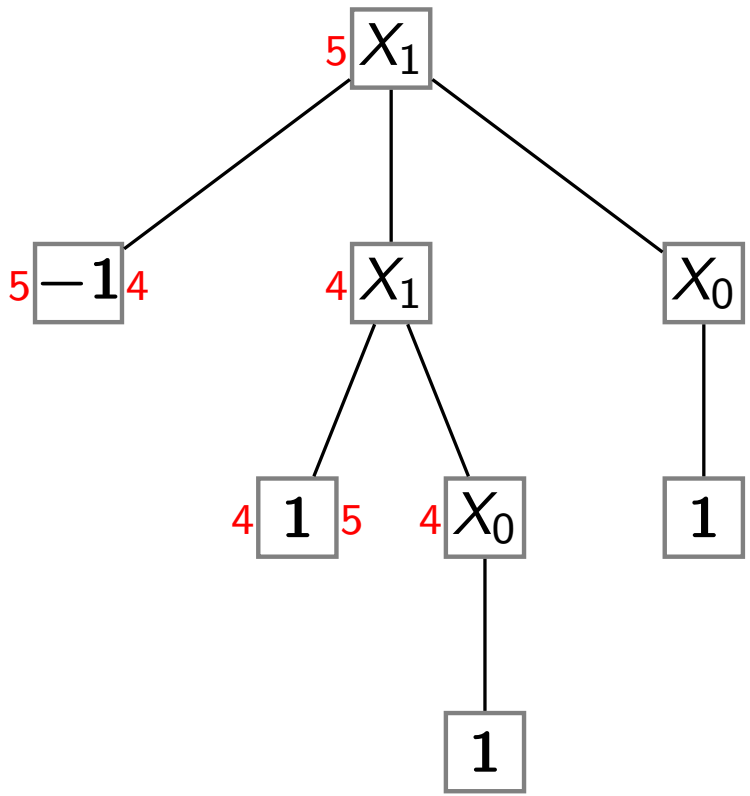
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

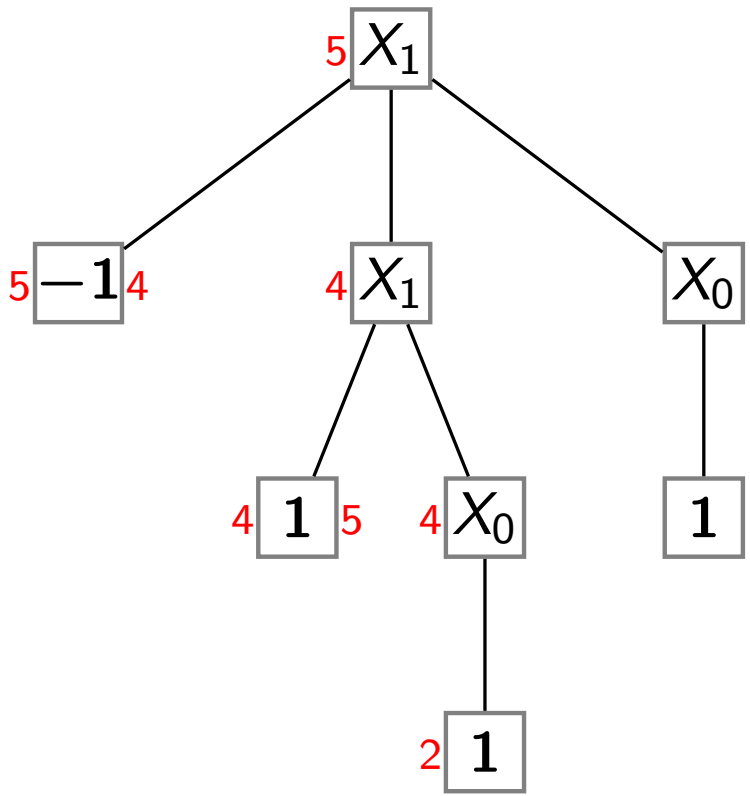
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

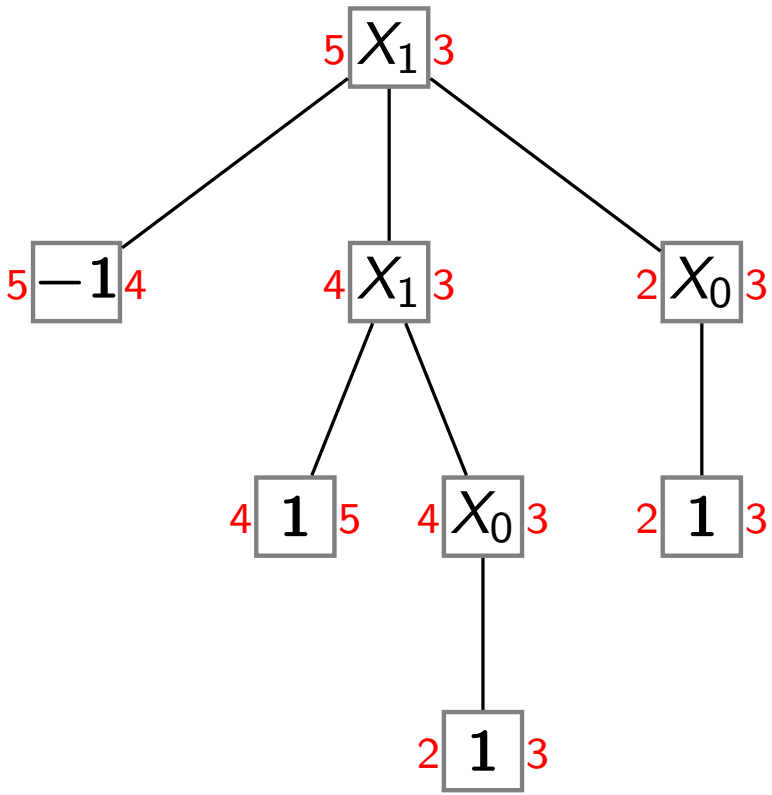
Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



$(Summary_{X_1}(5) \geq 3)$

Flow Trees

Certificates for $Summary_S(c) \geq d$? Annotated parse trees!



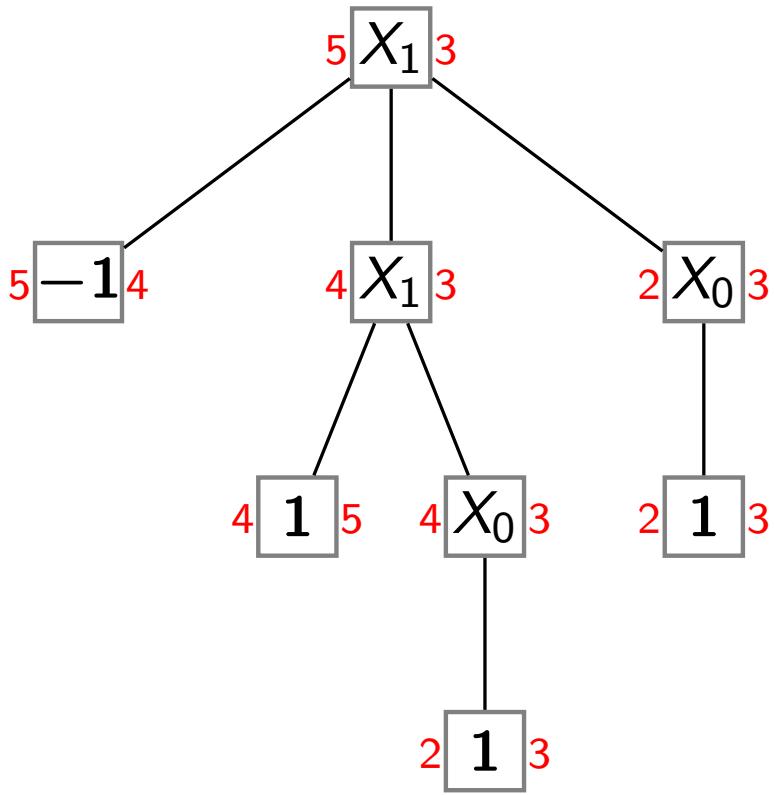
$(Summary_{X_1}(5) \geq 3)$

Flow Conditions

1. Nodes satisfy $Summary_X(IN) \geq OUT$
2. Labeling of neighboring nodes is consistent

Flow Trees . . . can be arbitrarily large!

Certificates for $Summary_S(c) \geq d$? Annotated parse trees!

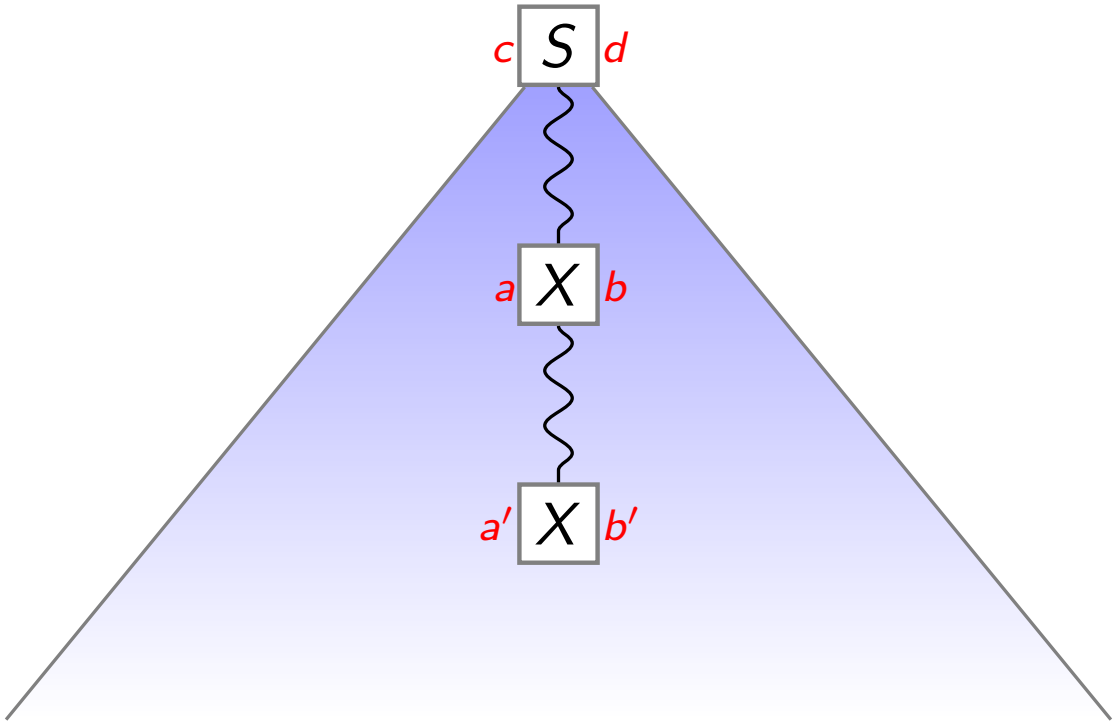


$(Summary_{X_1}(5) \geq 3)$

Flow Conditions

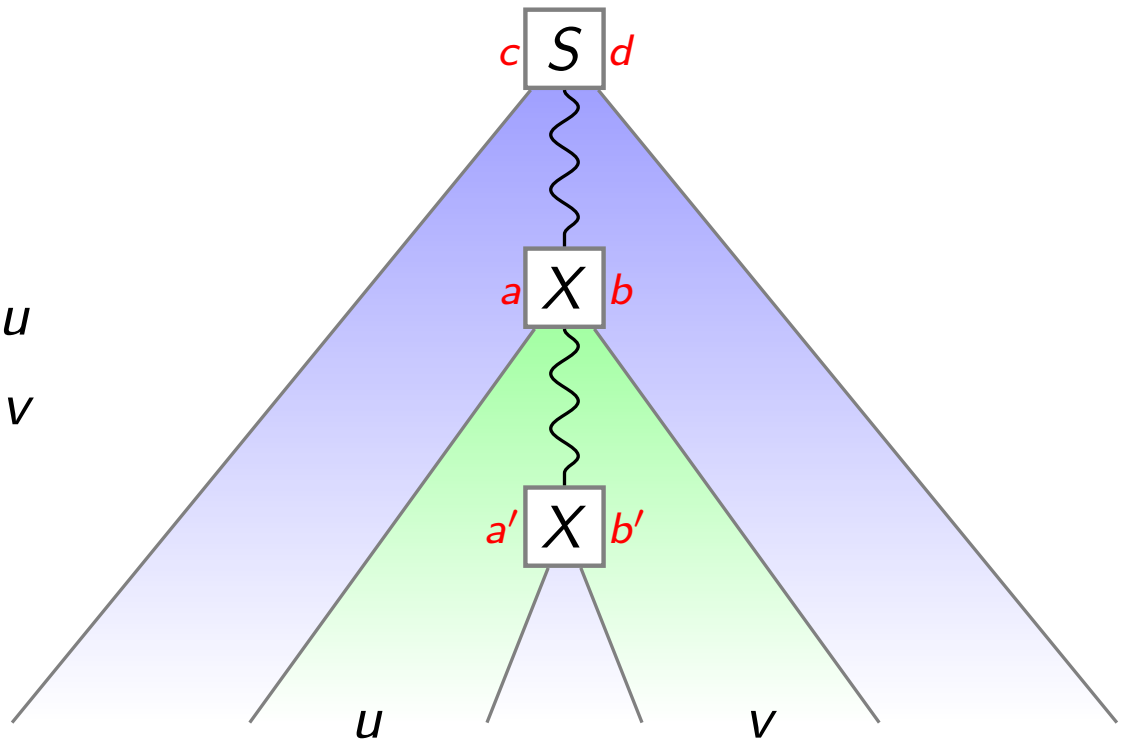
1. Nodes satisfy $Summary_X(IN) \geq OUT$
2. Labeling of neighboring nodes is consistent

Truncating and Collapsing Flow Trees



Truncating and Collapsing Flow Trees

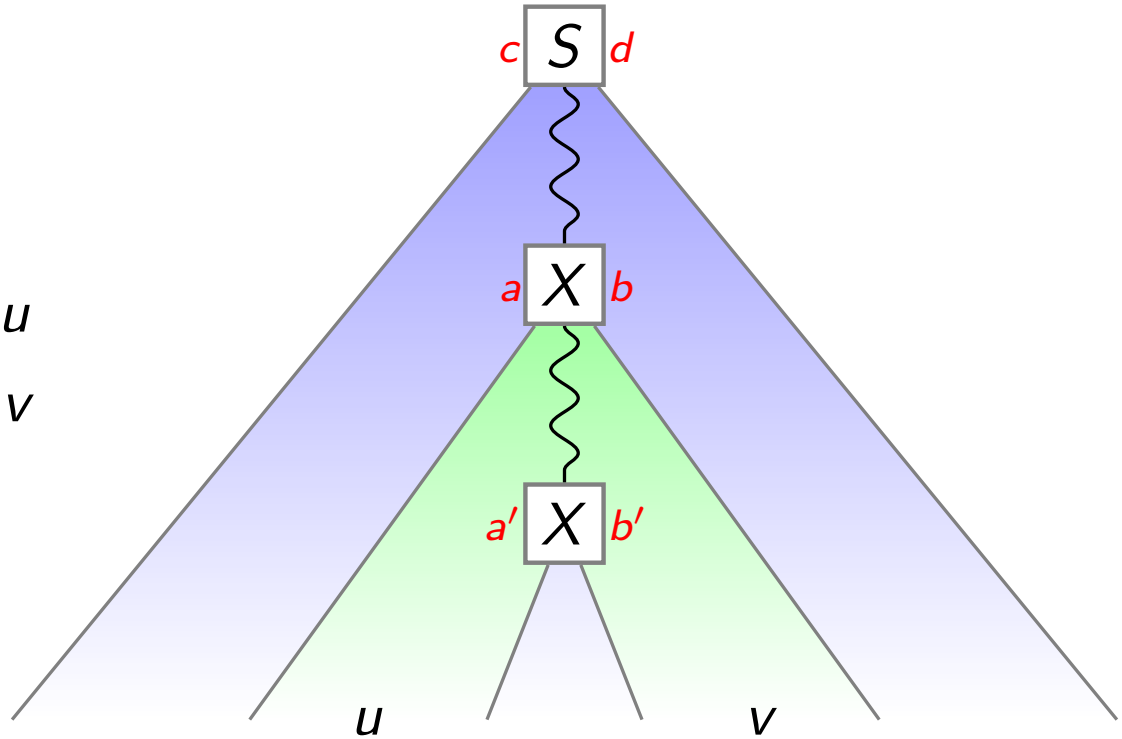
$$a' = a + \sum u$$
$$b' = b - \sum v$$



$$X \stackrel{*}{\Longrightarrow} uXv$$

Truncating and Collapsing Flow Trees

$$a' = a + \sum u$$
$$b' = b - \sum v$$



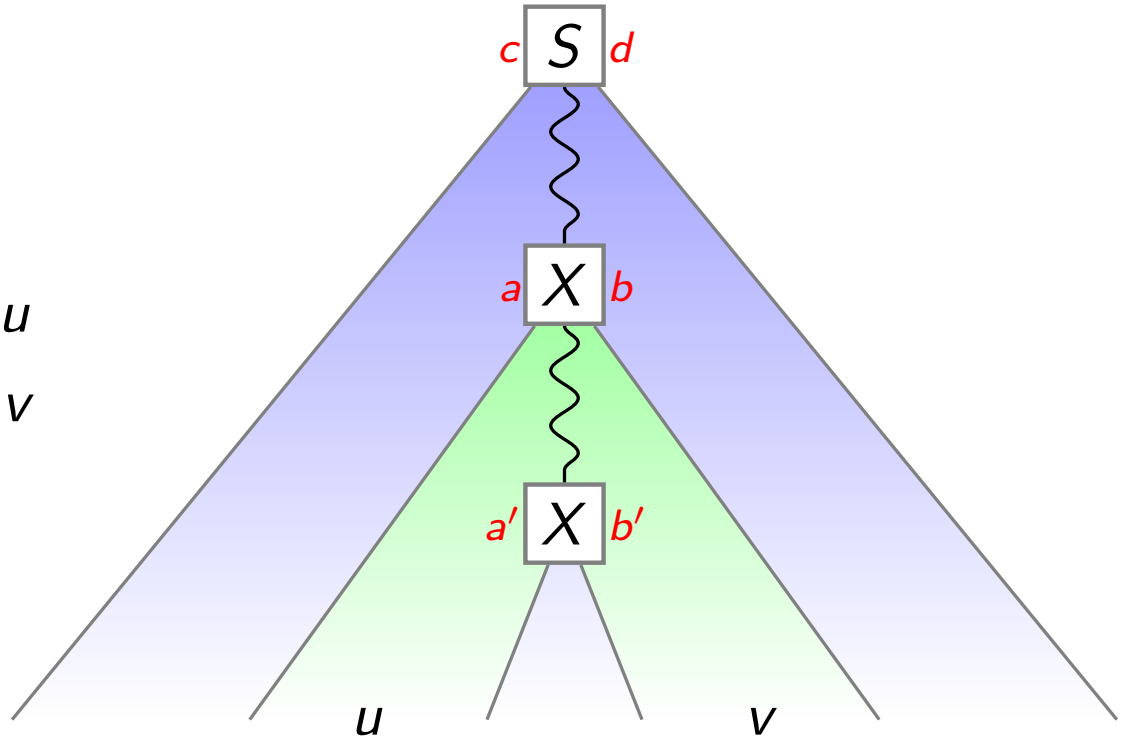
$$X \overset{*}{\Longrightarrow} uXv$$

$\sum u$	$\sum v$	a, a'	b, b'
≤ 0	≤ 0	$a \geq a'$	$b \leq b'$

Replace a' by a and b' by b and then collapse.

Truncating and Collapsing Flow Trees

$$a' = a + \sum u$$
$$b' = b - \sum v$$



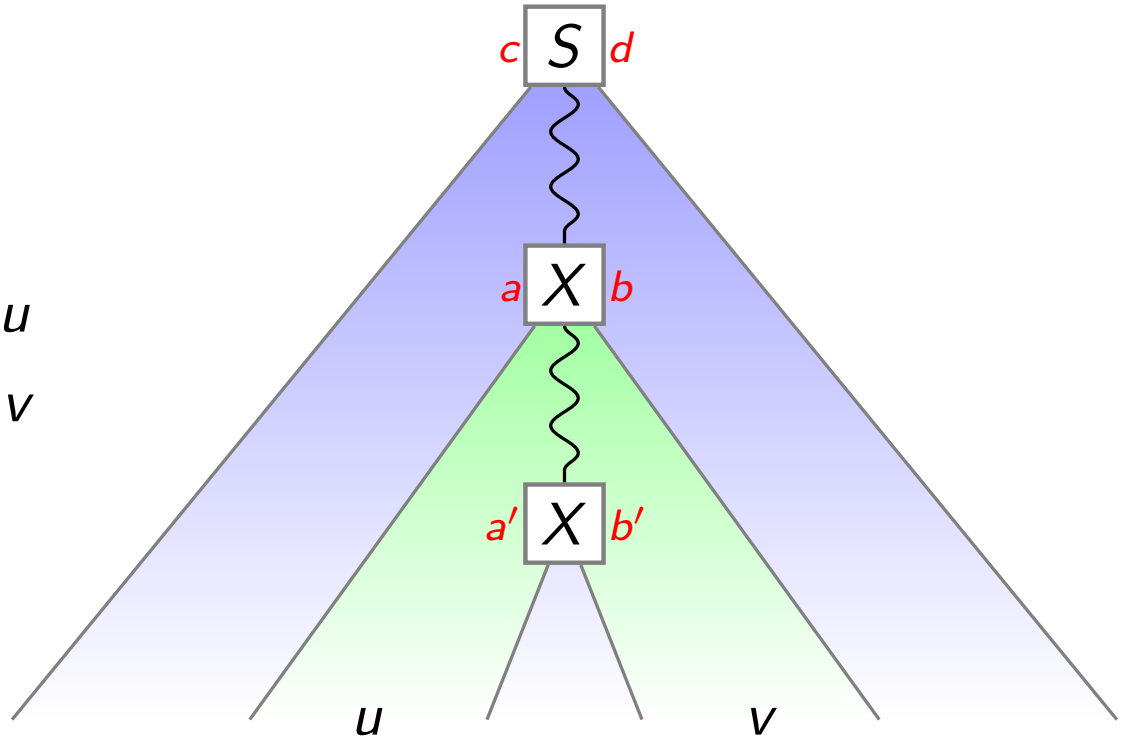
$$X \stackrel{*}{\Longrightarrow} uXv$$

$\sum u$	$\sum v$	a, a'	b, b'
> 0	≥ 0	$a < a'$	$b \geq b'$

Truncate at a' $\boxed{X}$ b' since we can iterate.

Truncating and Collapsing Flow Trees

$$a' = a + \sum u$$
$$b' = b - \sum v$$



$$X \stackrel{*}{\Longrightarrow} uXv$$

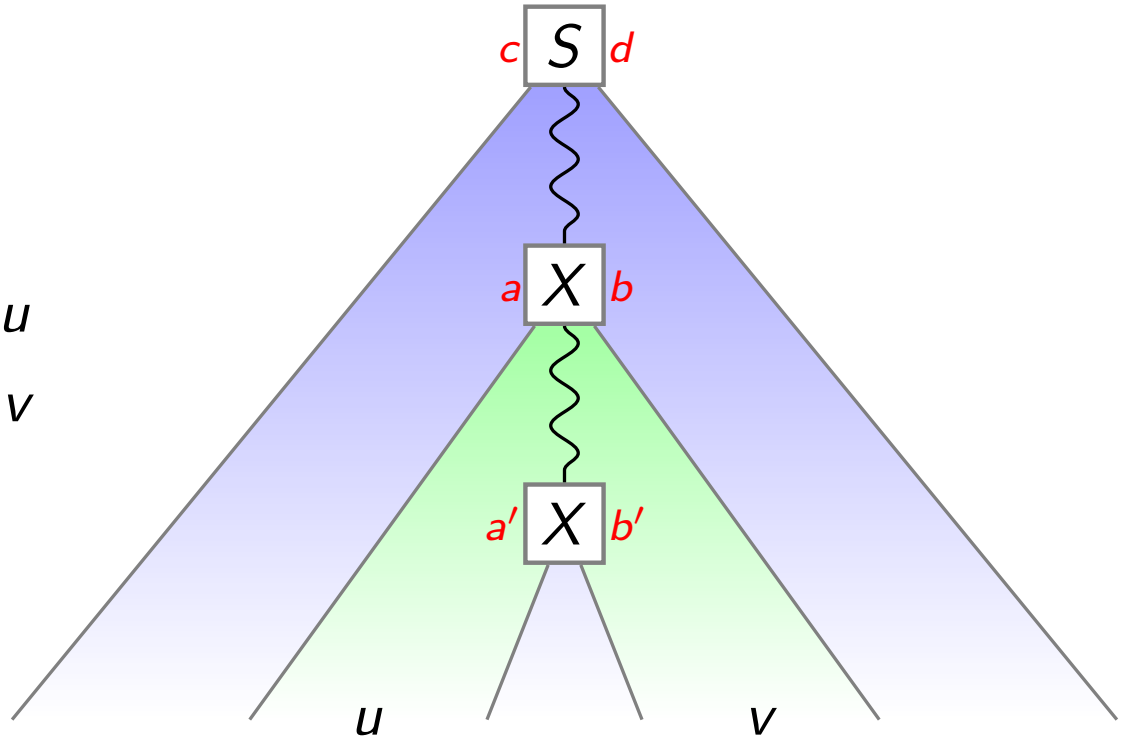
$\sum u$	$\sum v$	a, a'	b, b'
> 0	< 0	$a < a'$	$b < b'$

If $\sum u + \sum v > 0$ then truncate at a' $\boxed{X}$ b'.

If $\sum u + \sum v \leq 0$ then ?

Truncating and Collapsing Flow Trees

$$a' = a + \sum u$$
$$b' = b - \sum v$$



$$X \stackrel{*}{\Longrightarrow} uXv$$

$\sum u$	$\sum v$	a, a'	b, b'
< 0	> 0	$a > a'$	$b > b'$

If $\sum u + \sum v \leq 0$ then shift by $-\sum u$ and collapse.
If $\sum u + \sum v > 0$ then ?

Asymptotic Ratios

Definition (Ratio of a Variable)

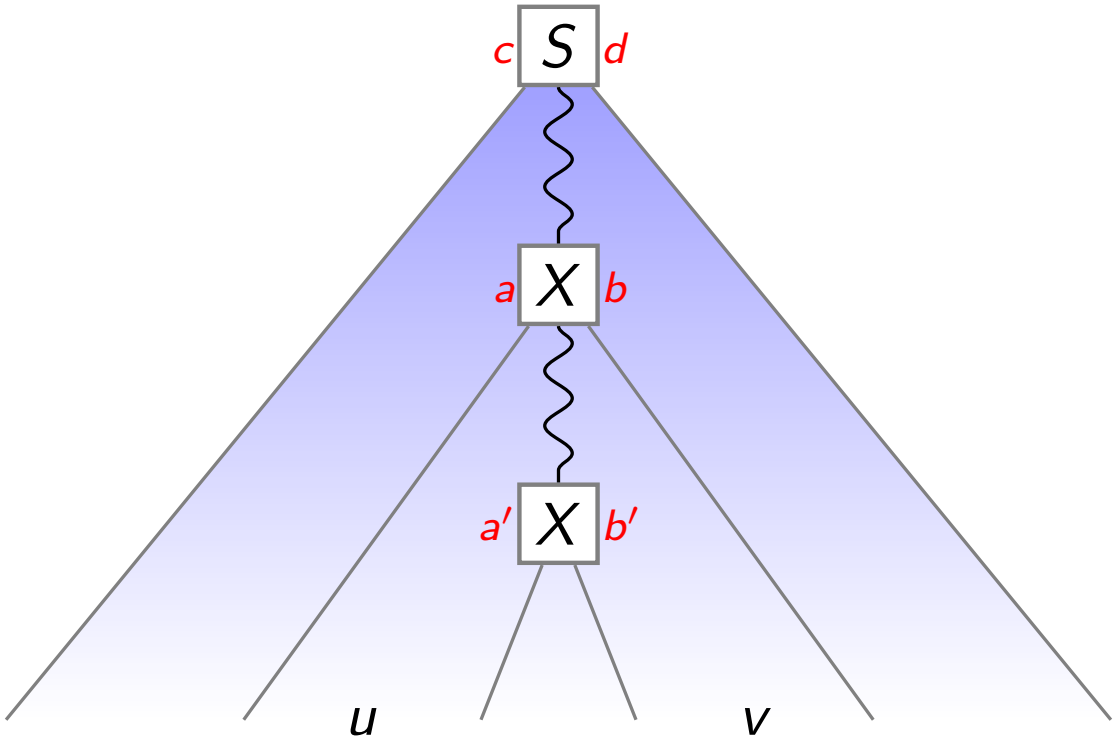
$$Ratio_X \quad = \quad \liminf_{n\to\infty} \frac{Summary_X(n)}{n}$$

Grammar for Ackermann Functions A_m

$$Summary_{X_m} \;=\; A_m$$

$$\begin{aligned} A_0(n) &= n+1 & \qquad Ratio_{X_0} &= 1 \\ A_1(n) &= n+2 & \qquad Ratio_{X_1} &= 1 \\ A_2(n) &= 2n+3 & \qquad Ratio_{X_2} &= 2 \\ A_3(n) &= 2^{n+3}-3 & \qquad Ratio_{X_3} &= \infty \end{aligned}$$

Pruning Flow Trees

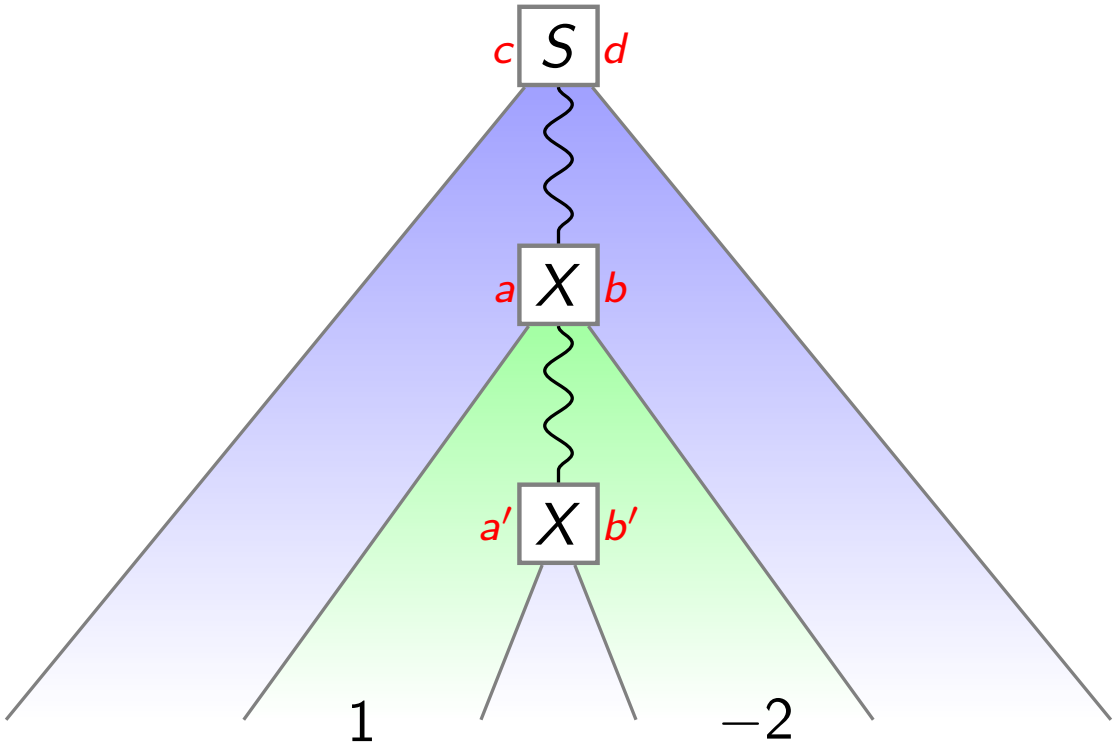


$$X \overset{*}{\Longrightarrow} uXv$$

Pruning Flow Trees

$a' = a + 1$

$b' = b + 2$

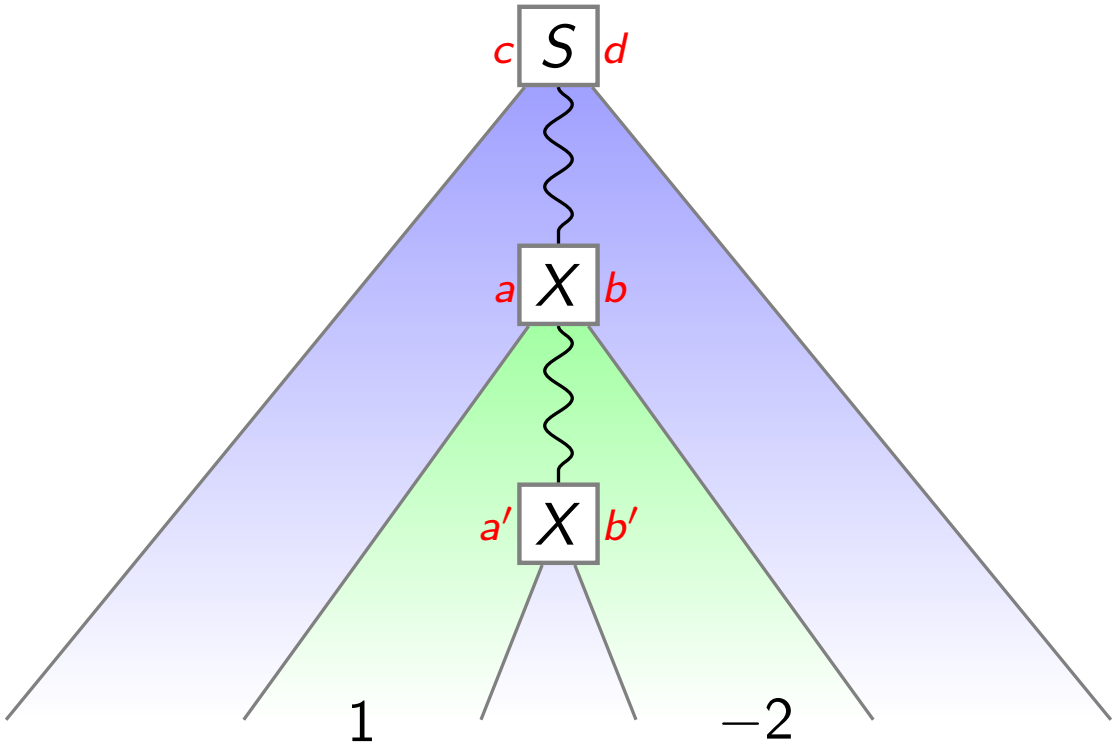


$X \overset{*}{\Longrightarrow} uXv$

Pruning Flow Trees

$a' = a + 1$

$b' = b + 2$



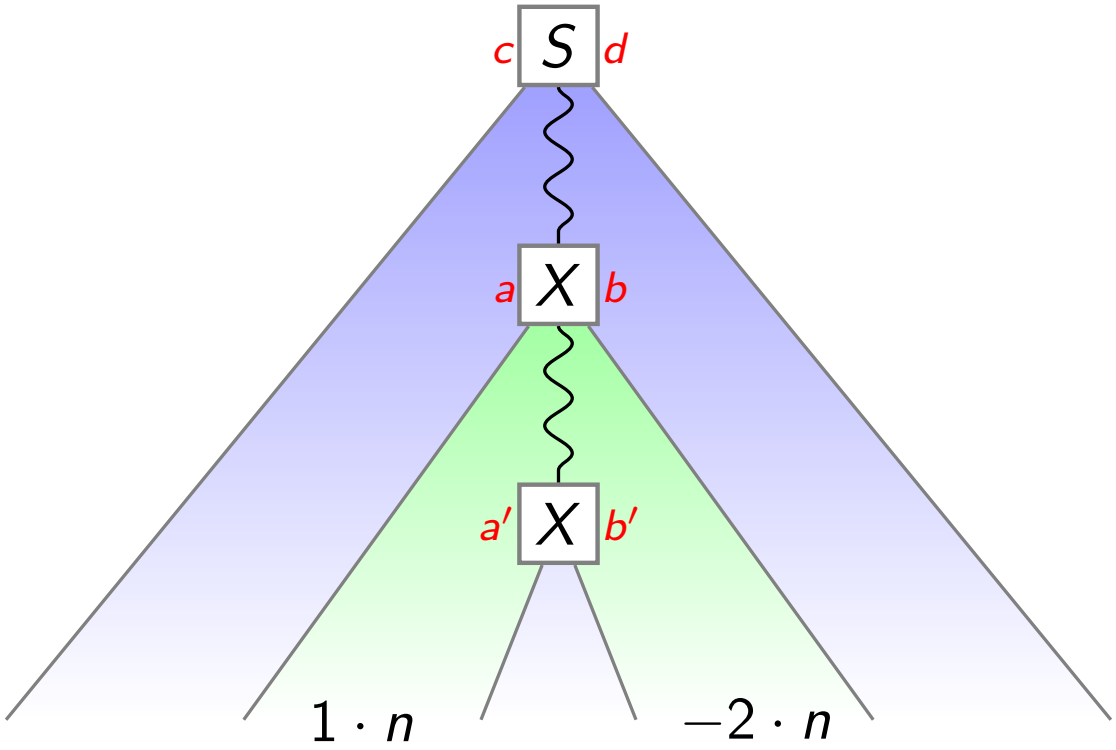
$X \stackrel{*}{\Longrightarrow} uXv$

Assume $Ratio_X = \infty$. There exists n_0 such that $Summary_X(n) \geq 3 \cdot n$ for all $n \geq n_0$.

Pruning Flow Trees

$a' = a + 1$

$b' = b + 2$



$X \stackrel{*}{\Longrightarrow} uXv$

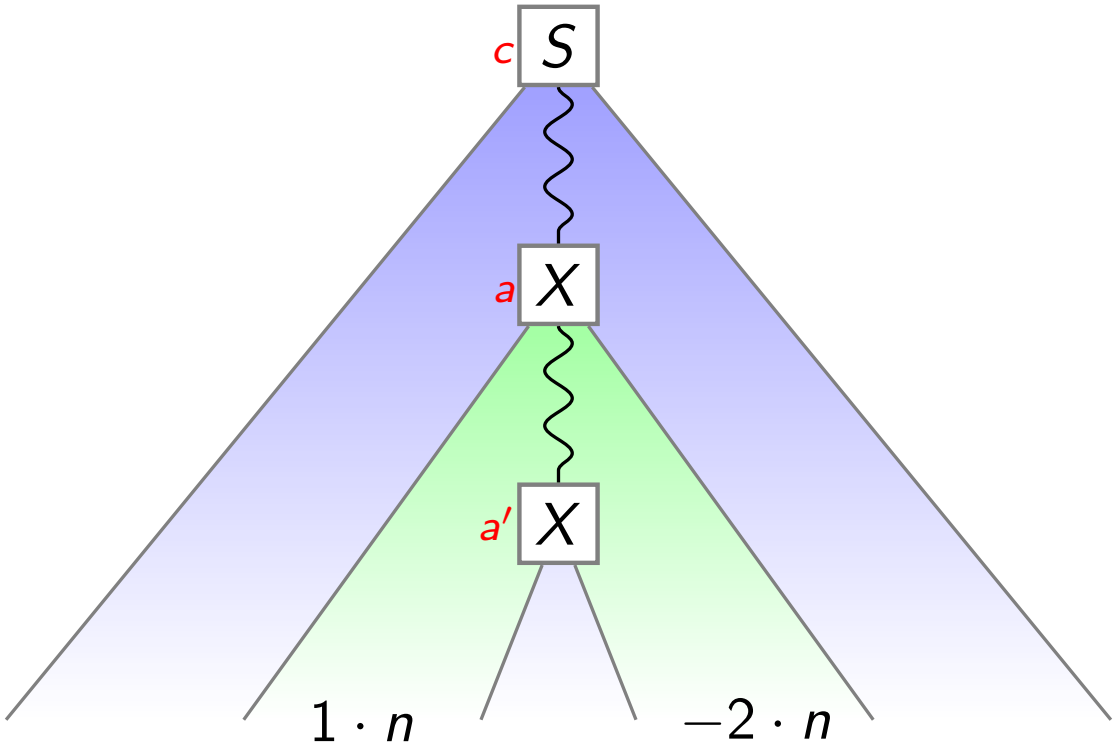
Assume $\mathit{Ratio}_X = \infty$. There exists n_0 such that $\mathit{Summary}_X(n) \geq 3 \cdot n$ for all $n \geq n_0$.

$$a \xrightarrow{u^n} a + n \xrightarrow{X} n' \geq 3a + 3n \xrightarrow{v^n} 3a + n \geq n$$

Pruning Flow Trees

$a' = a + 1$

$b' = b + 2$



$X \stackrel{*}{\Longrightarrow} uXv$

Assume $Ratio_X = \infty$. There exists n_0 such that $Summary_X(n) \geq 3 \cdot n$ for all $n \geq n_0$.

$$a \xrightarrow{u^n} a + n \xrightarrow{X} n' \geq 3a + 3n \xrightarrow{v^n} 3a + n \geq n$$

Hence, $Summary_X(a) = \infty$.

Small Certificates

Definition

A *certificate* is a partial flow tree such that, for every leaf $_{c}\boxed{X}_{d}$,

- either $\mathit{Ratio}_X < \infty$, or
- $\mathit{Ratio}_X = \infty$ and there is an ancestor $_{c'}\boxed{X}_{d'}$ with $c' < c$.

Proposition

$\mathit{Summary}_S(c) \geq d$ iff there is a certificate with root $_{c}\boxed{S}_{d}$ of at most exponential height and exponential input/output labels.

Small Certificates

Definition

A *certificate* is a partial flow tree such that, for every leaf $c\boxed{X}d$,

- either $\mathit{Ratio}_X < \infty$, or
- $\mathit{Ratio}_X = \infty$ and there is an ancestor $c'\boxed{X}d'$ with $c' < c$.

Proposition

$\mathit{Summary}_S(c) \geq d$ iff there is a certificate with root $c\boxed{S}d$ of at most exponential height and exponential input/output labels.

Guess-and-check algorithm

Need to check that an annotated partial parse tree is a certificate

Small Certificates and Decision Procedure

Definition

A *certificate* is a partial flow tree such that, for every leaf $c \boxed{X} d$,

- either $Ratio_X < \infty$, or
- $Ratio_X = \infty$ and there is an ancestor $c' \boxed{X} d'$ with $c' < c$.

Proposition

The question whether $Ratio_X = \infty$ is decidable. If $Ratio_X < \infty$, then $Summary_X$ is computable.

Guess-and-check algorithm

Need to check that an annotated partial parse tree is a certificate

Table of Contents

Summary

➪ Extension of the reduced reachability tree from VAS to pushdown VAS

- In fact to pushdown well-structured transition systems

➪ Boundedness and termination are decidable for pushdown VAS

- Hyper-Ackermannian (F_{ω^ω}) worst-case running time
- Tight bounds on the reachability set when it is finite

➪ Coverability is decidable for 1-dim pushdown VAS

(Counter-)boundedness for 1-dim pushdown VAS is solvable in exponential time

Open Problems

- ➪ Complexity of the boundedness problem for pushdown VAS
  - Lower bound: tower of exponentials (F_3) from [Lazić 2012]
  - Upper bound: hyper-Ackermann (F_{ω^ω})

- ➪ Decidability of coverability / reachability for pushdown VAS
  - Reachability open even in dimension 1

- ➪ Complexity of boundedness and coverability for 1-dim pushdown VAS
  - Both are NP-hard by reduction from SubsetSum
  - Boundedness is in ExpTime and Coverability is (?) in ExpSpace

Open Problems

- Complexity of the boundedness problem for pushdown VAS
  - Lower bound: tower of exponentials (F_3) from [Lazić 2012]
  - Upper bound: hyper-Ackermann (F_{ω^ω})
- Decidability of coverability / reachability for pushdown VAS
  - Reachability open even in dimension 1
- Complexity of boundedness and coverability for 1-dim pushdown VAS
  - Both are NP-hard by reduction from SubsetSum
  - Boundedness is in ExpTime and Coverability is (?) in ExpSpace

Thank You!

Pushdown Vector Addition Systems — Semantics

The semantics of a pushdown VAS $\langle Q, \Gamma, \Delta \rangle$ is the transition system $\langle Q \times \Gamma^* \times \mathbb{N}^d, \rightarrow \rangle$ whose transition relation $\rightarrow$ is given by

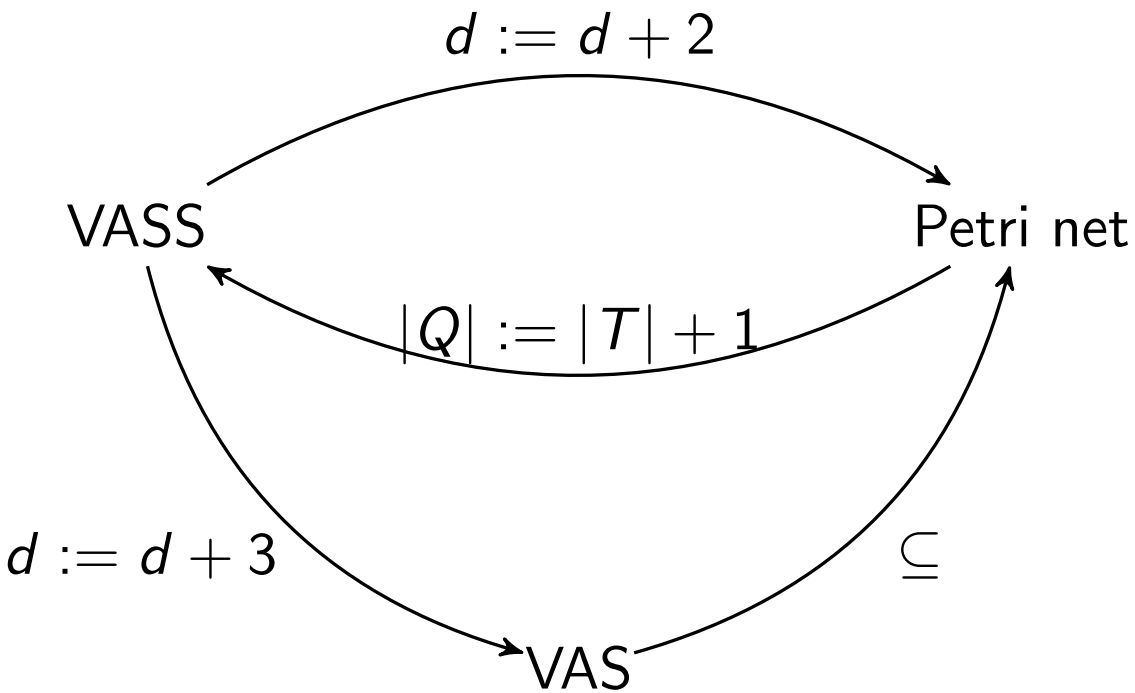
$$\frac{(p, \mathtt{nop}, \boldsymbol{a}, q) \in \Delta \ \wedge \ \boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{a} \geq \boldsymbol{0}}{(p, \sigma, \boldsymbol{v}) \rightarrow (q, \sigma, \boldsymbol{v}')}$$

$$\frac{(p, \mathtt{push}(\gamma), \boldsymbol{a}, q) \in \Delta \ \wedge \ \boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{a} \geq \boldsymbol{0}}{(p, \sigma, \boldsymbol{v}) \rightarrow (q, \gamma \cdot \sigma, \boldsymbol{v}')}$$

$$\frac{(p, \mathtt{pop}(\gamma), \boldsymbol{a}, q) \in \Delta \ \wedge \ \boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{a} \geq \boldsymbol{0}}{(p, \gamma \cdot \sigma, \boldsymbol{v}) \rightarrow (q, \sigma, \boldsymbol{v}')}$$

Additional Feature of Petri nets

Test $\mathtt{x} \geq cst$ without modifying $\mathtt{x}$



Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$

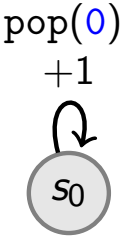
Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$

$$\begin{aligned} A_0(n) &= n+1 \\ A_1(n) &= n+2 \\ A_2(n) &= 2n+3 \\ A_3(n) &= 2^{n+3} - 3 \\ &\vdots \end{aligned}$$

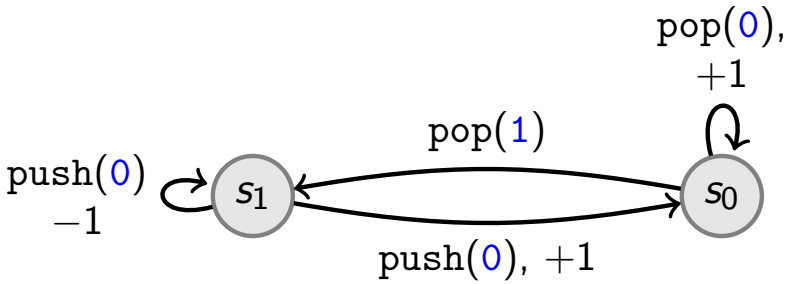
Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$



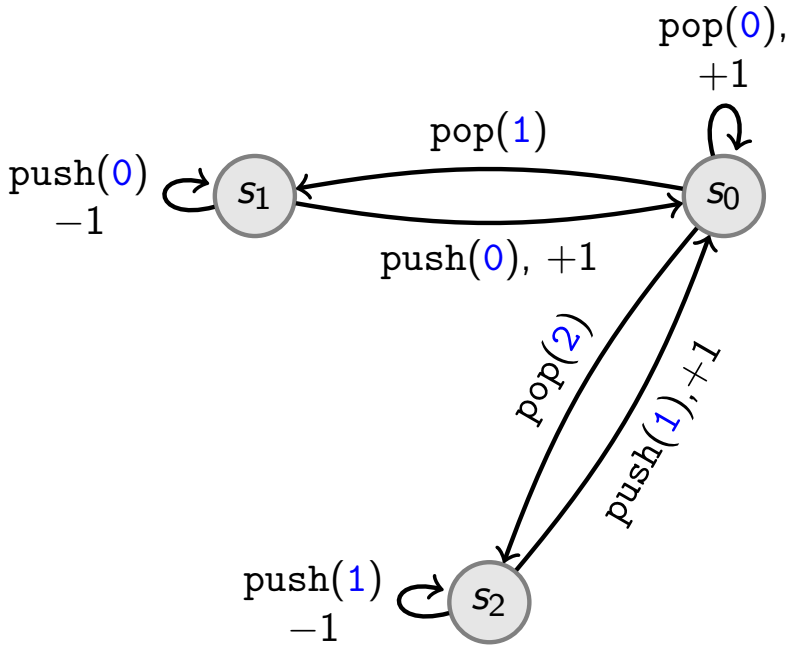
Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$



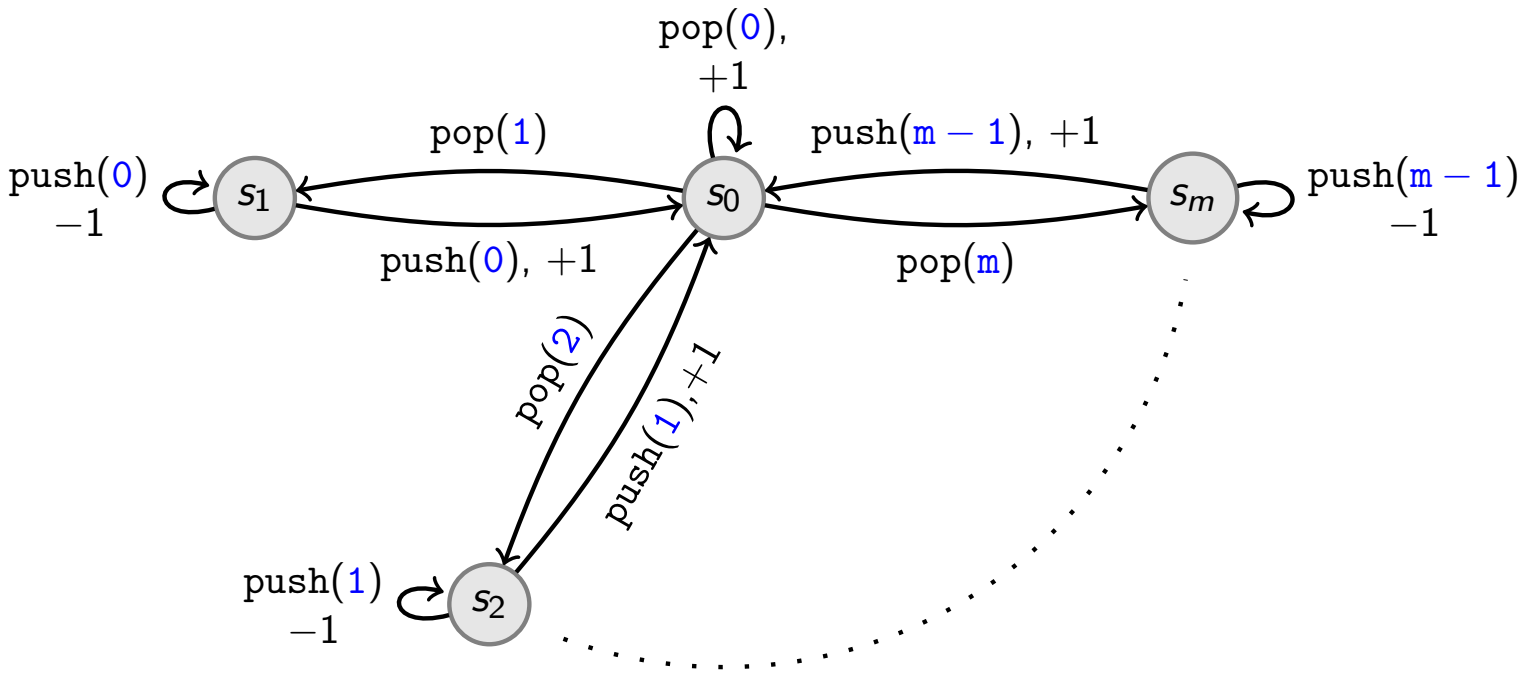
Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$



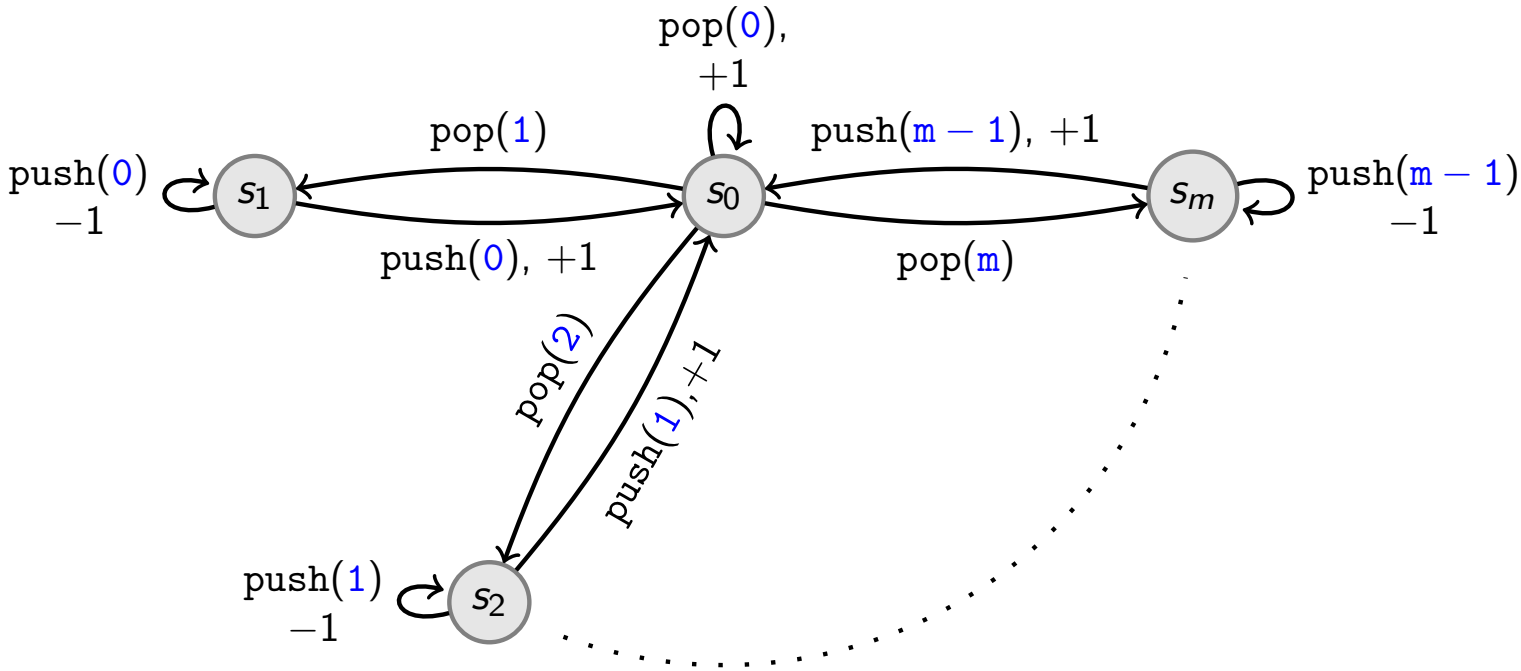
Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$



Weak Computation of Ackermann Functions

$$A_m(n) = \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$



$$(s_0, \mathtt{m}\bot, n) \xrightarrow{*} (s_0, \bot, A_m(n))$$

$$\text{If } (s_0, \mathtt{m}\bot, n) \xrightarrow{*} (s_0, \bot, n') \text{ then } n' \leq A_m(n)$$