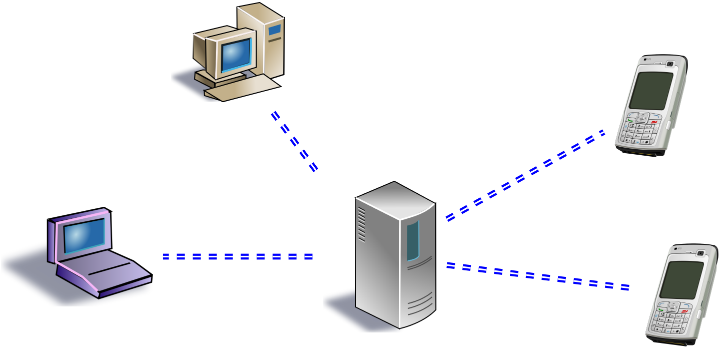
# On the Context-Freeness Problem for Vector Addition Systems

Jérôme Leroux, Vincent Penelle, M. Praveen and Grégoire Sutre

Univ. Bordeaux & CNRS, LaBRI, UMR 5800, Talence, France

LIFO Seminar, University of Orléans, May 12th 2014

# Verification of Concurrent Systems (1)



➪ Concurrent systems everywhere!

➪ Hard to design correctly

- Complex and unforeseen interactions between components
- Need for to automated verification tools

# Verification of Concurrent Systems (2)

## Model-checking

$$\mathcal{M} \quad \overset{?}{\models} \quad \varphi$$

Need models $\mathcal{M}$ with

- enough expressive power to represent the system to verify
- support for automatic verification (model-checking decidable)

Vector Addition Systems $\simeq$ Petri nets

- Classical model for (parametrized) concurrent systems
  - Rendez-vous synchronization
  - Asynchronous communication via unbounded unordered buffers
  - Dynamic process creation
- Fundamental class that is often used as a toolbox

# Table of Contents

# Table of Contents

# Vector Addition Systems with States in a Nutshell

A vector addition system with states is a finite-state automaton that is

- equipped with finitely many counters $\mathtt{x}, \mathtt{y}, \ldots$
- counters range over the set $\mathbb{N}$ of natural numbers
- counter operations are:
  - increment:

$$\mathtt{x} := \mathtt{x} + 1$$

  - guarded decrement:

$$\mathtt{assert}(\mathtt{x} > 0) \ ; \ \mathtt{x} := \mathtt{x} - 1$$

Vector addition systems with states are similar to Petri nets

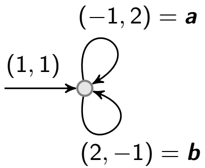
# Vector Addition Systems — Syntax

## Definition

A vector addition system is a pair $\langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A} \rangle$ where

- $\boldsymbol{v}_{\mathrm{init}} \in \mathbb{N}^d$ : initial vector
- $\boldsymbol{A} \subseteq \mathbb{Z}^d$ : finite set of actions

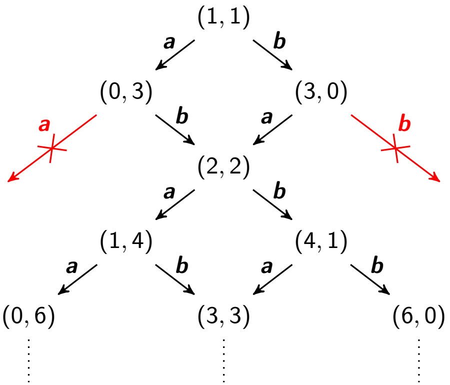
$$\begin{aligned} \boldsymbol{A} &= \{(-1,2),(2,-1)\} \\ \boldsymbol{v}_{\mathrm{init}} &= (1,1) \end{aligned}$$

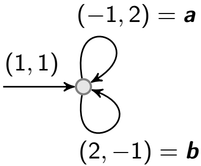


# Vector Addition Systems — Semantics

The semantics of a VAS $\langle \boldsymbol{v}_{\text{init}}, \boldsymbol{A} \rangle$ is the transition system $\langle \mathbb{N}^d, \boldsymbol{v}_{\text{init}}, \rightarrow \rangle$ whose transition relation $\rightarrow$ is given by

$$\frac{\boldsymbol{a} \in \boldsymbol{A} \ \wedge \ \boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{a} \geq \boldsymbol{0}}{\boldsymbol{v} \rightarrow \boldsymbol{v}'}$$



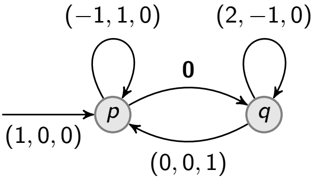
# Vector Addition Systems with States — Syntax

## Definition

A vector addition system with states is a tuple $\langle Q, q_{\mathrm{init}}, \boldsymbol{v}_{\mathrm{init}}, \Delta \rangle$ where

- $Q$ : finite set of states
- $q_{\mathrm{init}} \in Q$ : initial state
- $\boldsymbol{v}_{\mathrm{init}} \in \mathbb{N}^d$ : initial vector
- $\Delta \subseteq Q \times \mathbb{Z}^d \times Q$ : finite set of transition rules

$$
\begin{array}{rcl}
Q & = & \{p, q\} \\
q_{\mathrm{init}} & = & p \\
\boldsymbol{v}_{\mathrm{init}} & = & (1, 0, 0) \\
\Delta & = & \{(p, (-1, 1, 0), p), \ldots\}
\end{array}
$$



# Vector Addition Systems with States — Syntax

## Definition

A vector addition system with states is a tuple $\langle Q, q_{\text{init}}, \boldsymbol{v}_{\text{init}}, \Delta \rangle$ where

- $Q$ : finite set of states
- $q_{\text{init}} \in Q$ : initial state
- $\boldsymbol{v}_{\text{init}} \in \mathbb{N}^d$ : initial vector
- $\Delta \subseteq Q \times \mathbb{Z}^d \times Q$ : finite set of transition rules

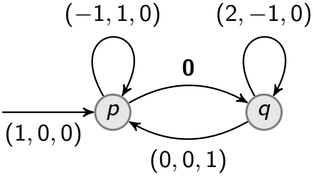
| VAS | | VASS |
|---|---|---|
| $\langle \boldsymbol{v}_{\text{init}}, \boldsymbol{A} \rangle$ | $\simeq$ | $\langle \{q\}, q, \boldsymbol{v}_{\text{init}}, \{q\} \times \boldsymbol{A} \times \{q\} \rangle$ |

# Vector Addition Systems with States — Semantics

The semantics of a VASS $\langle Q, q_{\mathrm{init}}, \boldsymbol{v}_{\mathrm{init}}, \Delta \rangle$ is the transition system $\langle Q \times \mathbb{N}^d, (q_{\mathrm{init}}, \boldsymbol{v}_{\mathrm{init}}), \rightarrow \rangle$ whose transition relation $\rightarrow$ is given by

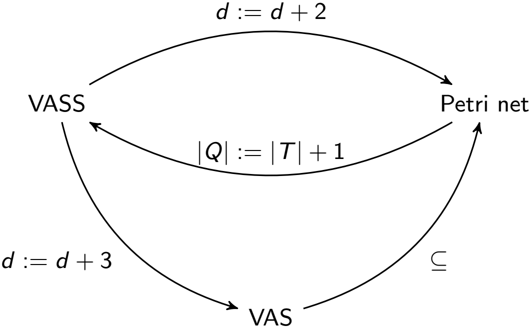
$$\frac{(q, \boldsymbol{a}, q') \in \Delta \;\wedge\; \boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{a} \geq \mathbf{0}}{(q, \boldsymbol{v}) \rightarrow (q', \boldsymbol{v}')}$$

$$\begin{array}{ccccc}
(p,1,0,0) & \rightarrow & (p,0,1,0) & & \\
 & & \downarrow & & \\
(q,2,0,0) & \leftarrow & (q,0,1,0) & & \\
\downarrow & & & & \\
(p,2,0,1) & \rightarrow & (p,1,1,1) & \rightarrow & (p,0,2,1) \\
 & & & & \downarrow \\
(q,4,0,1) & \leftarrow & (q,2,1,1) & \leftarrow & (q,0,2,1) \\
\downarrow & & & & \\
(p,4,0,2) & \rightarrow & (p,3,1,2) & \cdots\cdots\rightarrow &
\end{array}$$



### Additional Feature of Petri nets

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



# State of the Art — Reachability, Coverability, ...

**Input**: A VAS $\langle \boldsymbol{v}_{\text{init}}, \boldsymbol{A} \rangle$ and a final vector $\boldsymbol{v}_{\text{final}}$

**Reachability Problem — Whether $\boldsymbol{v}_{\text{init}} \xrightarrow{*} \boldsymbol{v}_{\text{final}}$**

Decidable [Mayr 81, Kosaraju 82], ExpSpace-hard [Lipton 76]

**Coverability Problem — Whether $\exists \boldsymbol{v} : \boldsymbol{v}_{\text{init}} \xrightarrow{*} \boldsymbol{v} \geq \boldsymbol{v}_{\text{final}}$**

Decidable [Karp&Miller 69], ExpSpace-complete [Rackoff 78]

**Boundedness Problem — Whether $\{\boldsymbol{v} \in \mathbb{N}^d \mid \boldsymbol{v}_{\text{init}} \xrightarrow{*} \boldsymbol{v}\}$ is finite**

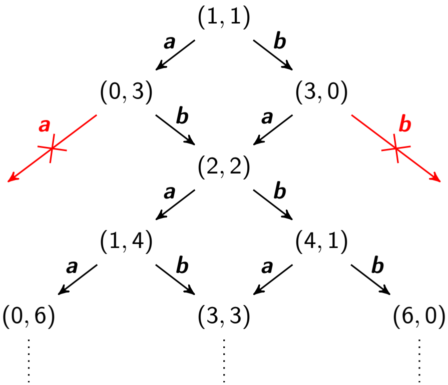
Decidable [Karp&Miller 69], ExpSpace-complete [Rackoff 78]

- Many ExpSpace-complete problems (place boundedness, ...)
- Some undecidable problems (equality of reachability sets, ...)

# Trace Language

A trace of a VAS $\langle \boldsymbol{v}_{\text{init}}, \boldsymbol{A} \rangle$ is a sequence $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ such that

$$\boldsymbol{v}_{\text{init}} \xrightarrow{\boldsymbol{a}_1} \boldsymbol{v}_1 \cdots \boldsymbol{v}_{n-1} \xrightarrow{\boldsymbol{a}_n} \boldsymbol{v}_n$$




- ***abb*** is a trace
- ***aa*** is not a trace

**Definition**

The trace language is the set of all traces

# The Regularity and Context-Freeness Problems

## Definition

**Input**: A VAS $\langle \boldsymbol{v}_{\text{init}}, \boldsymbol{A} \rangle$
**Output**: Is the trace language of $\langle \boldsymbol{v}_{\text{init}}, \boldsymbol{A} \rangle$ regular/context-free?

## Regularity

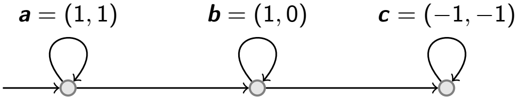
- Decidable [Valk&Vidal-Naquet 81, Ginzburg&Yoeli 80]
- EXPSPACE-complete [Demri 10]
  - Witness: trace $u_1\sigma_1 \cdots u_k\sigma_k$ such that $L \cap (u_1\sigma_1^\star \cdots u_k\sigma_k^\star)$ is not regular
  - Length at most doubly-exponential (in $|\boldsymbol{A}|$)

## Context-Freeness

- Decidable [Schwer 92]
  - Complex criterion, based on the coverability graph
  - Intricate proof with flaws

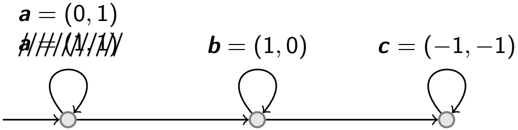
# Table of Contents

# Example 1




Its trace language is context-free:

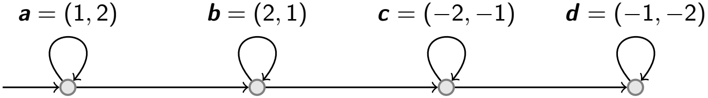
$$L \ = \ \{\boldsymbol{a}^n \boldsymbol{b}^m \boldsymbol{c}^p \mid n \geq p \wedge m \geq 0\}$$



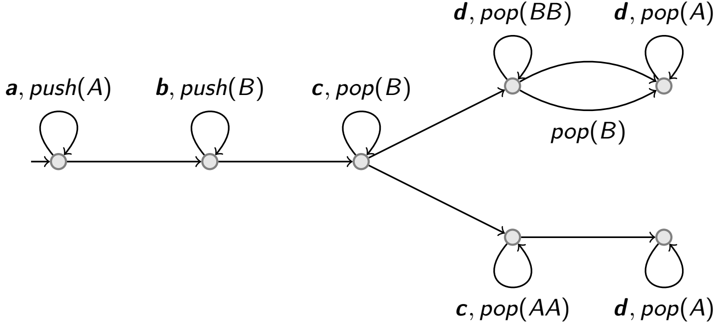

Its trace language is not context-free:

$$L = \{ \boldsymbol{a}^n \boldsymbol{b}^m \boldsymbol{c}^p \mid n \geq p \quad \wedge \quad \overset{m \geq p}{\cancel{m \geq 0}} \}$$

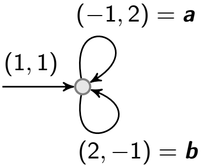
# Example 3




Its trace language is context-free:



# Example 4 (Running Example)




Its trace language is not context-free:

$$L \ \cap \ (\boldsymbol{ab})^\star \boldsymbol{a}^\star \boldsymbol{b}^\star \quad = \quad \{(\boldsymbol{ab})^n \boldsymbol{a}^m \boldsymbol{b}^p \mid n+1 \geq m \wedge n+1+2m \geq p\}$$

# Simulation of a VAS by a Pushdown Automaton: Main Ideas

Goal: recognize the traces of a VAS with a pushdown automaton

Use the stack to store the values of the counters

For each vector read from the input tape

- Positive vector $\rightarrow$ Push it onto the stack
- Non-positive vector $\rightarrow$ Match it with the stack

Problem: some components may be lost by matching

# Match and Remainder

$\boldsymbol{u} \geq \mathbf{0}$, $\boldsymbol{v} \not\geq \mathbf{0}$, and $\lambda$ the greatest rational such that $\boldsymbol{u} + \lambda \boldsymbol{v} \geq \mathbf{0}$,

$$\mathrm{mat}(\boldsymbol{u}, \boldsymbol{v}) \;=\; \begin{cases} (1-\lambda) \cdot \boldsymbol{v} & \text{if } \lambda < 1 \\ (1-\frac{1}{\lambda}) \cdot \boldsymbol{u} & \text{if } \lambda \geq 1 \end{cases}$$

$$\mathrm{rem}(\boldsymbol{u}, \boldsymbol{v}) \;=\; \boldsymbol{u} + \boldsymbol{v} - \mathrm{mat}(\boldsymbol{u}, \boldsymbol{v}) \geq \mathbf{0}$$

$$\xrightarrow{\mathit{read}(\boldsymbol{u})} \; \xrightarrow{\mathit{push}(\boldsymbol{u})} \; \cdots\!\!\dashrightarrow \; \xrightarrow{\mathit{read}(\boldsymbol{v})} \; \xrightarrow{\boldsymbol{p} \leftarrow \boldsymbol{v}} \; \xrightarrow{\mathit{pop}(\boldsymbol{u})} \; \begin{cases} \xrightarrow{\lambda \geq 1} \; \mathit{push}(\mathrm{mat}(\boldsymbol{u}, \boldsymbol{p})) \\ \xrightarrow{\lambda < 1} \; \boldsymbol{p} \leftarrow \mathrm{mat}(\boldsymbol{u}, \boldsymbol{p}) \end{cases}$$

# Match and Remainder: Example with $\lambda < 1$

Take $\boldsymbol{u} = (1, 2, 1)$ and $\boldsymbol{v} = (-1, -3, 2)$

The greatest $\lambda$ such that $\boldsymbol{u} + \lambda \boldsymbol{v} \geq \mathbf{0}$ is $\lambda = \dfrac{2}{3}$

After matching $\boldsymbol{u}$ and $\boldsymbol{v}$, we are left with

$$\mathrm{mat}(\boldsymbol{u}, \boldsymbol{v}) \ = \ (1 - \lambda) \cdot \boldsymbol{v} \ = \ (-\frac{1}{3}, -1, \frac{2}{3})$$

The remainder is

$$\mathrm{rem}(\boldsymbol{u}, \boldsymbol{v}) \ = \ \boldsymbol{u} + \lambda \boldsymbol{v} \ = \ (\frac{1}{3}, 0, \frac{7}{3})$$

# Simulation of a VAS by a Pushdown Machine (Variables)

## Global variables

- local buffer $\mathtt{w} \in \boldsymbol{A}^*$
- accumulated remainder $\mathtt{r} \in \mathbb{Q}^d_{\geq 0}$
- $\mathtt{stack}$ of extracted vectors in $\mathbb{Q}^d_{\geq 0}$

They represent the VAS configuration:

$$\boldsymbol{v}_{\mathrm{init}} + \Delta(\mathtt{w}) + \mathtt{r} + \Delta(\mathtt{stack})$$

Initialize ()

$\mathtt{w} \leftarrow \varepsilon$
$\mathtt{r} \leftarrow \mathbf{0}$
$\mathtt{stack} \leftarrow \varepsilon$

$\Delta(\boldsymbol{a}_1 \cdots \boldsymbol{a}_n) = \boldsymbol{a}_1 + \cdots + \boldsymbol{a}_n$

# Simulation of a VAS by a Pushdown Machine (Read)

VAS configuration: $\boldsymbol{v}_{\mathrm{init}} + \Delta(\texttt{w}) + \texttt{r} + \Delta(\texttt{stack})$

Read ($\boldsymbol{a} \in \boldsymbol{A}$)

**if** $\boldsymbol{v}_{\mathrm{init}} + \Delta(\texttt{w}) + \texttt{r} + \Delta(\texttt{stack}) + \boldsymbol{a} \geq \mathbf{0}$ **then**

$\qquad \texttt{w} \leftarrow \texttt{w} \cdot \boldsymbol{a}$

$\qquad$ Simplify ()

**else**

$\qquad$ *reject*

# Simulation of a VAS by a Pushdown Machine (Simplify)

Simplify ()

**while** $\Delta(\sigma) + \mathtt{r} + \Delta(\mathtt{stack}) \geq \mathbf{0}$ for some suffix $\sigma \neq \varepsilon$ of $\mathtt{w}$ **do**
    Pick such a suffix $\sigma$
    $\mathtt{w} \leftarrow \mathtt{w} \cdot \sigma^{-1}$
    $\mathrm{p} \leftarrow \Delta(\sigma)$
    **while** $\mathrm{p} \not\geq \mathbf{0}$ **do**
        **if** $\mathtt{stack}$ is empty **then**
            *fail*
        **else**
            pop $\gamma$ from $\mathtt{stack}$
            $(\mathrm{p}, \mathtt{r}) \leftarrow (\mathrm{mat}(\gamma, \mathrm{p}), \mathtt{r} + \mathrm{rem}(\gamma, \mathrm{p}))$
    **if** $\mathrm{p}(i) > 0 \Rightarrow \mathtt{r}(i) > 0$ for every index $i$ **then**
        $\mathtt{r} \leftarrow \mathtt{r} + \mathrm{p}$
    **else**
        push $\mathrm{p}$ onto $\mathtt{stack}$

# Properties of the Simulation

**Proposition**

*If fail is not reachable, then the pushdown machine recognizes the trace language of the VAS*

Proof: easy

**Proposition**

*The language recognized by the pushdown machine is context-free*

Proof: next slide

**Corollary**

*If fail is not reachable, the trace language of the VAS is context-free*

# Proof: The Pushdown Machine Recognizes a CFL

### Proposition

*The set of reachable values of* $\mathtt{w}$ *is finite*

### Proposition

*The reachable alphabet for* $\mathtt{stack}$ *is finite*

➪ Standard PDA, with stack alphabet $\Gamma \subseteq \mathbb{Q}^d_{\geq 0}$, augmented with:

- counters $\mathtt{r} \in \mathbb{Q}^d_{\geq 0}$, updated by assignments $\mathtt{r} \leftarrow \mathtt{r} + \boldsymbol{v}$ with $\boldsymbol{v} \geq \mathbf{0}$
- tests: $\mathtt{r}(i) + \Delta(\mathtt{stack})(i) \;\#\; z$ and $\mathtt{r}(i) \;\#\; z$ where $\# \in \{\leq, \geq\}$

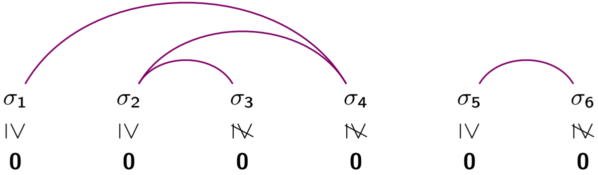
Replace $\mathbb{Q}^d_{\geq 0}$ by $\mathbb{N}^d_{\geq 0}$ and let $K \in \mathbb{N}$ be the maximum $z$ of the tests

Abstract by $\top$ components of $\mathtt{r}$ and $\Delta(\mathtt{stack})$ larger than $K$

➪ Store $\mathtt{r}$ in the state and $\Delta(\mathtt{stack})$ within the stack

# Table of Contents

# Matching Schemes




## Definition

A matching scheme is a tuple $(\sigma_1, \ldots, \sigma_k, U)$ such that

- $\sigma_1, \ldots, \sigma_k$ : words in $\boldsymbol{A}^\star$
- $U$ : binary relation on $\{1, \ldots, k\}$ that is nested:

$$\begin{array}{rcl} (s,t) \in U & \Rightarrow & s \leq t \\ (r,t) \in U \wedge (s,u) \in U & \Rightarrow & \neg(r < s < t < u) \end{array}$$

- For every $(s,t) \in U$, $\Delta(\sigma_s) \geq \mathbf{0}$ and $\Delta(\sigma_t) \not\geq \mathbf{0}$

# Loss by Matching $(s, t) \in U$

Consider the case of single actions: $\sigma_s = \boldsymbol{a}_s$ and $\sigma_t = \boldsymbol{a}_t$

$$\begin{array}{ccc} \boldsymbol{a}_s & & \boldsymbol{a}_t \\ \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\not\leq$} \\ \mathbf{0} & & \mathbf{0} \end{array}$$

$$\begin{aligned} \boldsymbol{a}_s &= (1, 1) \\ \boldsymbol{a}_t &= (-1, -2) \\ \lambda_{s,t} &= \tfrac{1}{2} \\ \mathrm{lost}(s, t) &= \{1\} \end{aligned}$$

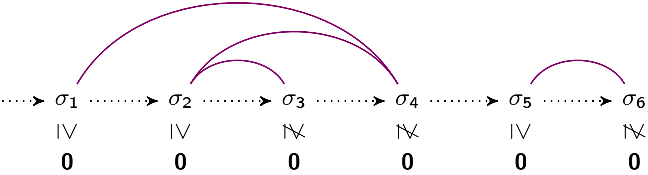
Use one $\boldsymbol{a}_s$ to do as many $\boldsymbol{a}_t$ as possible. What do we lose?

Ratio: $$\lambda_{s,t} = \max \{\lambda \in \mathbb{Q} \mid \boldsymbol{a}_s + \lambda \boldsymbol{a}_t \geq \mathbf{0}\}$$

**Definition**

$$\mathrm{lost}(s, t) = \|\boldsymbol{a}_s + \lambda_{s,t} \boldsymbol{a}_t\|^+ = \|\,\mathrm{rem}(\boldsymbol{a}_s, \boldsymbol{a}_t)\|^+$$

# Witnesses of Non-Context-Freeness




## Definition

A witness of non-context-freeness is a tuple $(\sigma_1, \ldots, \sigma_k, U)$ such that

- $(\sigma_1, \ldots, \sigma_k, U)$ : matching scheme
- $u_1\sigma_1 \cdots u_k\sigma_k$ is a trace for some $u_1, \ldots, u_k$
- $\Delta(\sigma_k) \not\geq \mathbf{0}$ and $\|\Delta(\sigma_k)\|^- \subseteq \bigcup_{(s,t)\in U} \mathrm{lost}(s,t)$
- $\|\Delta(\sigma_t)\|^- \subseteq \|\Delta(\sigma_{s_{\min}(t)})\|^+$ for all $(\cdot, t) \in U$ with $t < k$

# Characterization of Context-Freeness through Witnesses

**Proposition**

*If* *fail* *is reachable, then a witness of non-context-freeness can be constructed from a run reaching* *fail*

**Proposition**

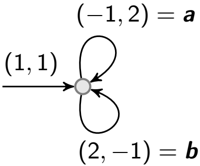
*If a VAS admits a witness of non-context-freeness, then its trace language is not context-free*

Proof: based on the characterization of bounded context-free languages by Ginsburg and Spanier

**Theorem**

$$\begin{array}{ll} & \textit{The trace language of a VAS } \langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A} \rangle \textit{ is not context-free} \\ \iff & \textit{fail is reachable} \\ \iff & \langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A} \rangle \textit{ admits a witness of non-context-freeness} \end{array}$$

# Witnesses of Non-Context-Freeness: Example




$L \ \cap \ (\boldsymbol{ab})^\star \boldsymbol{a}^\star \boldsymbol{b}^\star$ is not context-free

## Witness

$$(\boldsymbol{ab}, \boldsymbol{a}, \boldsymbol{b}, \{(1, 2)\})$$

$$\begin{array}{cccccc} \cdots\to & \boldsymbol{ab} & \cdots\to & \boldsymbol{a} & \cdots\to & \boldsymbol{b} \\ & \geq & & \not\geq & & \not\geq \\ & \mathbf{0} & & \mathbf{0} & & \mathbf{0} \end{array}$$

- The largest rational $\lambda$ such that $\Delta(\boldsymbol{ab}) + \lambda\Delta(\boldsymbol{a}) \geq \mathbf{0}$ is $\lambda_{1,2} = 1$
- $\mathrm{lost}(1, 2) = \|\Delta(\boldsymbol{ab}) + 1\Delta(\boldsymbol{a})\|^+ = \|(0, 3)\|^+ = \{2\}$ contains $\|\boldsymbol{b}\|^-$

# Towards an Encoding in Logic

## Definition

A witness of non-context-freeness is a tuple $(\sigma_1, \ldots, \sigma_k, U)$ such that

- $\Delta(\sigma_k) \not\geq \mathbf{0}$ and $\|\Delta(\sigma_k)\|^- \subseteq \bigcup_{(s,t) \in U} \mathrm{lost}(s,t)$
- $\cdots$

➪ Bound $k$

➪ Non-linear arithmetic constraints over the $\Delta(\sigma_j)$

$$i \in \mathrm{lost}(s,t) \quad \Leftrightarrow \quad \bigvee \begin{array}{l} \boldsymbol{\delta}_t(i) > 0 \ \wedge \ (\boldsymbol{\delta}_s(i) > 0 \vee \|\boldsymbol{\delta}_t\|^- \subseteq \|\boldsymbol{\delta}_s\|^+) \\ \\ \boldsymbol{\delta}_t(i) \leq 0 \ \wedge \ \bigvee_{j \neq i} \boldsymbol{\delta}_s(i) \cdot \boldsymbol{\delta}_t(j) < \boldsymbol{\delta}_s(j) \cdot \boldsymbol{\delta}_t(i) \end{array}$$

($\boldsymbol{\delta}_j$ stands for $\Delta(\sigma_j)$)

# Simpler Witnesses of Non-Context-Freeness

## Proposition

*The trace language of* $\langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A} \rangle$ *is not context-free iff* $\langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A} \rangle$ *admits a witness of non-context-freeness* $(\sigma_1, \ldots, \sigma_k, U)$ *such that*

- $k \leq 3d + 1$
- *for every* $(s, t) \in U$*, the ratio* $\lambda_{s,t}$ *is either* $0$ *or* $1$

**Proof**:

- Simplify $U$ by keeping only pairs that add new lost components
- At most two pairs $(s, t)$ and $(s_{\min}(t), t)$ for each $i \in \{1, \ldots, d\}$
- Remove useless $\sigma_j$ but keep $\sigma_k$

# Simpler Witnesses of Non-Context-Freeness

### Proposition

*The trace language of $\langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A} \rangle$ is not context-free iff $\langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A} \rangle$ admits a witness of non-context-freeness $(\sigma_1, \ldots, \sigma_k, U)$ such that*

- $k \leq 3d + 1$
- *for every $(s, t) \in U$, the ratio $\lambda_{s,t}$ is either $0$ or $1$*

**Proof**:

- Let $(s, t) \in U$ with positive ratio $\lambda_{s,t} = \frac{p}{q}$
- Replacing $\sigma_s$ by $(\sigma_s)^q$ and $\sigma_t$ by $(\sigma_t)^p$ yields the ratio $\frac{q}{p} \cdot \frac{p}{q} = 1$
- But this modifies the ratio of other pairs $(s', t)$ and $(s, t')$
- Follow nesting of $U$ to prevent conflicts

# Encoding by Linear Arithmetic Constraints

The requirement that $\lambda_{s,t} \in \{0,1\}$ simplifies the encoding of $\mathsf{lost}(s,t)$

$$
\begin{array}{rcl}
i \in \mathsf{lost}(s,t) & \Leftrightarrow & i \in \|\boldsymbol{\delta}_s + \lambda_{s,t}\boldsymbol{\delta}_t\|^{+} \\[2ex]
 & \Leftrightarrow & \begin{array}{l} \lambda_{s,t} = 0 \;\wedge\; \boldsymbol{\delta}_s(i) > 0 \\ \vee \\ \lambda_{s,t} = 1 \;\wedge\; \boldsymbol{\delta}_s(i) + \boldsymbol{\delta}_t(i) > 0 \end{array}
\end{array}
$$

$$
\begin{array}{rcl}
\lambda_{s,t} = 0 & \Leftrightarrow & \displaystyle\bigvee_{i=1}^{d} (\boldsymbol{\delta}_s(i) = 0 \;\wedge\; \boldsymbol{\delta}_t(i) < 0) \\[3ex]
\lambda_{s,t} = 1 & \Leftrightarrow & \boldsymbol{\delta}_s + \boldsymbol{\delta}_t \geq \mathbf{0} \;\wedge\; \displaystyle\bigvee_{i=1}^{d} (\boldsymbol{\delta}_s(i) + \boldsymbol{\delta}_t(i) = 0 \wedge \boldsymbol{\delta}_t(i) < 0)
\end{array}
$$

➪ Need linear relations between $\Delta(\sigma_1), \ldots, \Delta(\sigma_k)$

# Table of Contents

# Yet Another Logic for VAS Traces: Syntax

Given a trace of the form $u_1 \, \sigma_1 \, \cdots \, u_k \, \sigma_k$

$$\dashrightarrow \sigma_1 \dashrightarrow \sigma_2 \dashrightarrow \sigma_3 \dashrightarrow \sigma_4 \dashrightarrow \sigma_5 \dashrightarrow \sigma_6$$

The logic expresses properties of $\Delta(\sigma_1), \ldots, \Delta(\sigma_k)$

**Definition**

$$\begin{array}{lcll} t & ::= & z \, \boldsymbol{\delta}_j(i) \mid t + t & (z \in \mathbb{Z}, j \geq 1, 1 \leq i \leq d) \\ \phi & ::= & t \geq n \mid \phi \vee \phi \mid \phi \wedge \phi & (n \in \mathbb{N}) \end{array}$$

Variables $\boldsymbol{\delta}_j$ are interpreted as $\Delta(\sigma_j)$

# Yet Another Logic for VAS Traces: Semantics

**Definition (Demri 10)**

A trace $u_1\,\sigma_1\,\cdots\,u_k\,\sigma_k$ is self-covering when

$$\|\Delta(\sigma_j)\|^- \;\subseteq\; \|\Delta(\sigma_1)\|^+ \cup \cdots \cup \|\Delta(\sigma_{j-1})\|^+ \qquad (\forall j \leq k)$$

**Definition**

$$\begin{aligned} u_1\,\sigma_1\,\cdots\,u_k\,\sigma_k \models \phi \quad & \text{if} \quad \phi\,[\Delta(\sigma_j)\,/\,\boldsymbol{\delta}_j] \text{ holds} \\ \langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A}\rangle \models \phi \quad & \text{if} \quad u_1\,\sigma_1\,\cdots\,u_k\,\sigma_k \models \phi \text{ for some s.-c. trace} \end{aligned}$$

➪ The model-checking problem would become Reachability-hard if

- arbitrary traces were allowed
- intermediate steps were forbidden ($u_j = \varepsilon$)

# A Few Example Properties

Unboundedness: $$\bigvee_{i=1}^{d} \boldsymbol{\delta}_1(i) \geq 1$$

Place unboundedness: $$\bigvee_{j=1}^{d} \boldsymbol{\delta}_j(p) \geq 1$$

Non-regularity: $$\bigvee_{i=1}^{d} -\boldsymbol{\delta}_{d+1}(i) \geq 1$$

Non-context-freeness: $$\bigvee_{k=1}^{3d+1} \bigvee_{\substack{U \subseteq \{1,\ldots,k\} \\ \text{nested}}} \cdots \quad \boldsymbol{\delta}_s(i) + \boldsymbol{\delta}_t(i) = 0 \quad \cdots$$

# Small Model Property

**Theorem**

*If there is a self-covering trace in $\langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A} \rangle$ satisfying $\phi$, then there is one of length at most*

$$2^{p(|\boldsymbol{A}|+|\phi|)\cdot c^{(d\cdot k(\phi))^3}}$$

*where $p$ is a polynomial and $c$ is a constant*

**Corollary**

*The model-checking problem $\langle \boldsymbol{v}_{\mathrm{init}}, \boldsymbol{A} \rangle \overset{?}{\models} \phi$ is* ExpSpace*-complete*

**Corollary**

*The context-freeness problem for VAS is* ExpSpace*-complete*

Proof: ExpSpace-hardness by reduction from the boundedness problem

# Table of Contents

# Conclusion and Open Problems

New proof, simpler than the one of Schwer

- Characterization of non-context-freeness through simple witnesses
- The trace language of a VAS is context-free if, and only if, it has a context-free intersection with every bounded regular language

New logic for expressing properties of VAS traces

- Can express linear relations between cycles visited by a run
- Model-checking is ExpSpace-complete
- Incomparable with existing logics that are solvable in ExpSpace

Complexity of the context-freeness problem: ExpSpace-complete

Open Problems

- Existing ExpSpace logics are incomparable: try to unify them!
- Coverability and reachability for Pushdown VAS

# There is hopefully still time for . . .

Questions ?