IST AUSTRIA
Institute of Science and Technology

Faster Algorithms for Weighted Recursive State Machines

Krishnendu Chatterjee
Bernhard Kragl
Samarth Mishra
Andreas Pavlogiannis

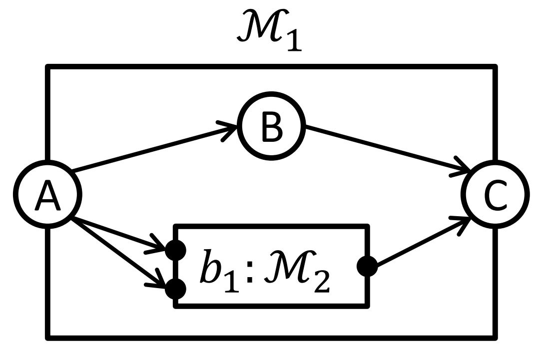
Recursive State Machines (RSMs)

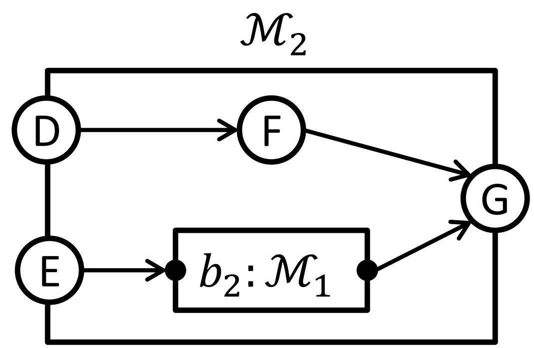
Formal model of recursive computation

Linearly equivalent to pushdown systems (PDSs)

Advantages:

- Natural modeling
- Many parameters
  - Number of modules f
  - Entry bound $\theta_e = \max_i |En_i|$
  - Exit bound $\theta_x = \max_i |Ex_i|$
  - $\theta = \max_i \min(|En_i|, |Ex_i|)$
  - …





$$\langle A, \varepsilon\rangle \Rightarrow \langle E, b_1\rangle \Rightarrow \langle A, b_2 b_1\rangle$$
$$\Rightarrow \langle B, b_2 b_1\rangle \Rightarrow \langle\langle b_2, C\rangle, b_1\rangle$$
$$\Rightarrow \langle\langle b_1, G\rangle, \varepsilon\rangle$$

RSMs over Semirings

Label RSM transitions with weights from idempotent semiring $\langle W, \oplus, \otimes, 0, 1 \rangle$

weight of computation: $\otimes$ weight of computation set: $\oplus$

	$\boldsymbol{W}$	$\oplus$	$\otimes$	**0**	**1**
Reachability	$\mathbb{B}$	$\vee$	$\wedge$	$\bot$	$\top$
Shortest path	$\mathbb{R}^+ \cup \{\infty\}$	min	$+$	∞	0
Most probable path	$[0,1]$	max	$\cdot$	0	1
IFDS	$2^D \xrightarrow{d} 2^D$	$\sqcap$	$\circ$	$\lambda x. \top$	$\lambda x. x$

Canonical partial order

$$a \leq b \Leftrightarrow a \oplus b = a$$

Monotonicity

$$a \leq b \Rightarrow a \otimes c \leq b \otimes c$$

Finite-height: $H \in \mathbb{N}$ longest descending chain in $\leq$

Distance Problems

Given a set of initial configurations S

- Configuration distance

$$d(S, \langle u, b_1 \cdots \mathrm{b}_n \rangle)$$

- Superconfiguration distance

$$d(S, \langle u, \mathcal{M}_1 \cdots \mathcal{M}_n \rangle)$$

- Node distance

$$d(S, u)$$

Our Solution

1. Configuration automata

   Representation structures for sets of RSM configurations [BEM'97]

   Initial automaton C, s.t. $\mathcal{L}(C) = S$

2. Dynamic programming algorithm

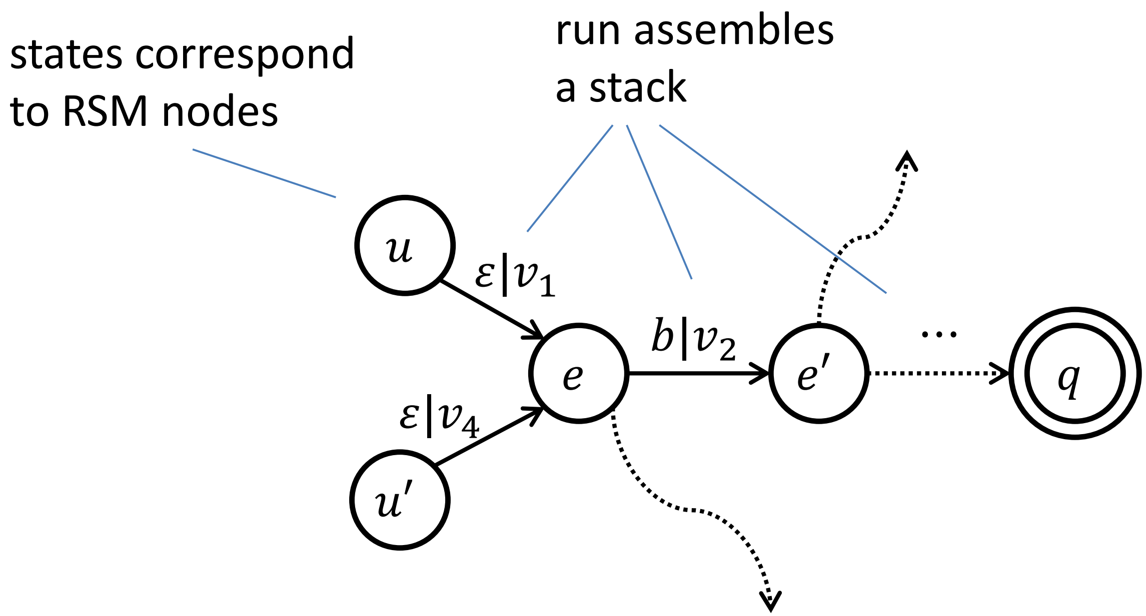
   Compute C^*, representing reachable configurations and distances

   Key: Entry-to-Exit summaries [ABEGRY'05]

3. Distance extraction algorithms

   Query configuration/superconfiguration/node distances from C^*

Configuration Automata

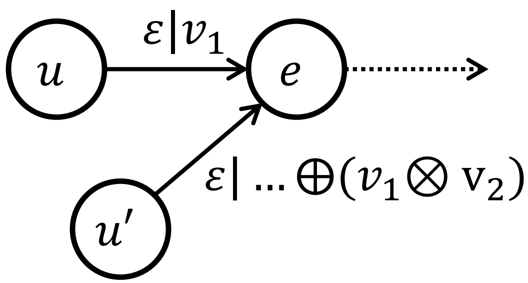


$\langle u, b \cdots \rangle$ is an accepted configuration

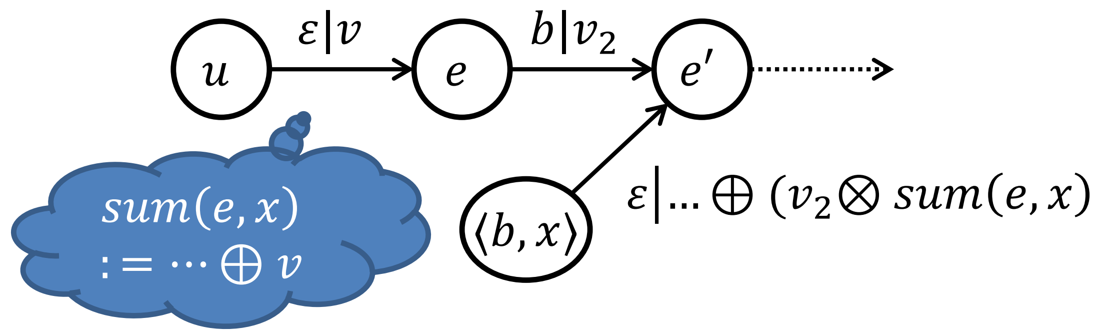
Weight of configuration is $\bigoplus$ over all accepting runs

Relaxation Steps

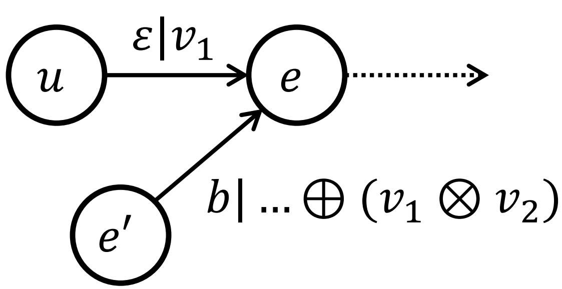
Internal transition: $u \xrightarrow{v_2} u'$



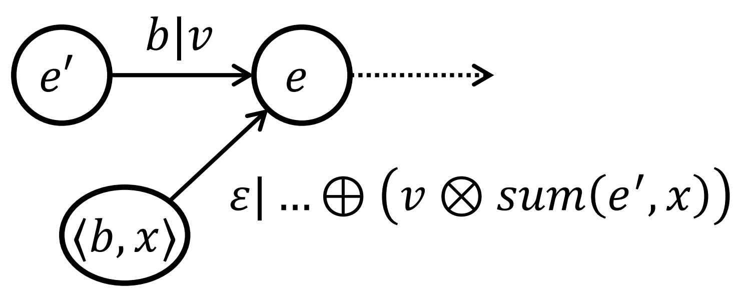
Exit transition: $u \to x$



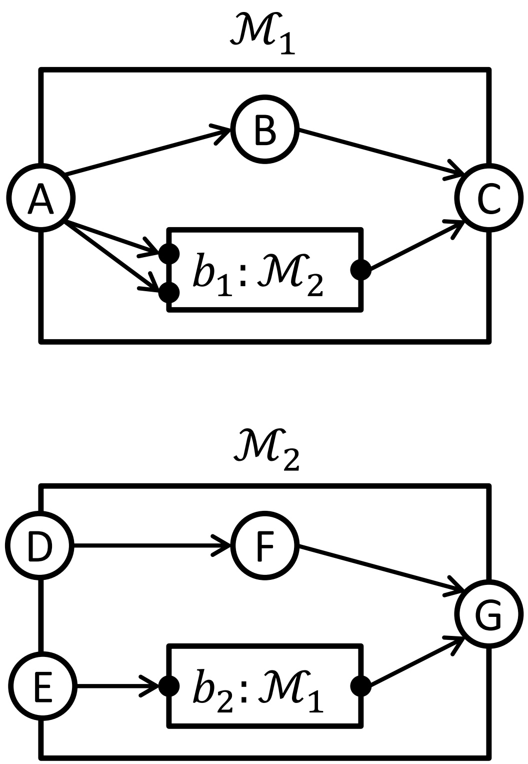
Call transition: $u \xrightarrow{v_2} \langle b, e' \rangle$

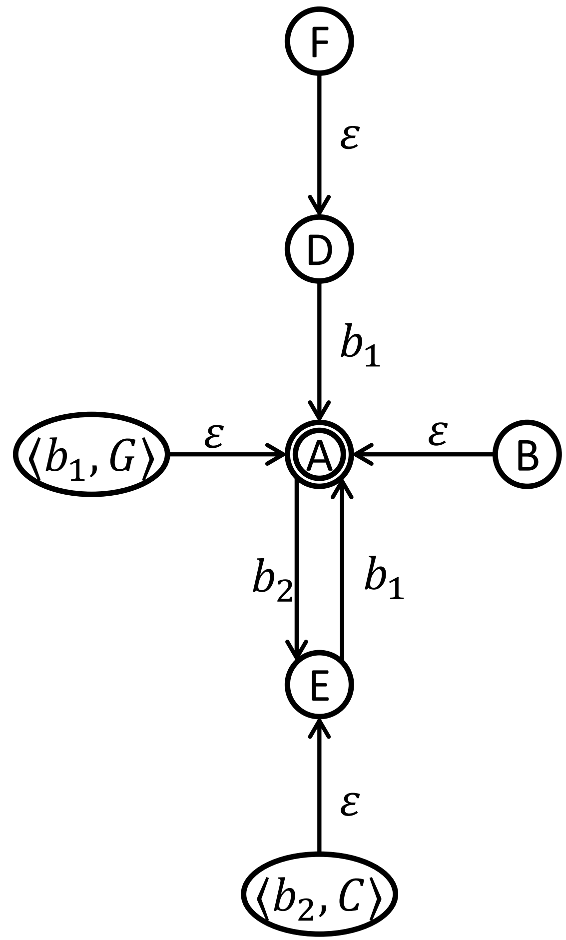


Using summary: $sum(e', x)$



Reachability Example





Summaries:

$A \rightsquigarrow C$

$E \rightsquigarrow G$

$D \rightsquigarrow G$

Correctness and Complexity

For every configuration c: $d(\mathcal{L}(C), c) = C^*(c)$

C^* is constructed in time

$$\mathcal{O}\big(H \cdot (|\mathcal{R}| \cdot \theta_e + \theta_e \cdot \theta_x \cdot |Call|)\big)$$

Compared to PDS algorithm

$$\mathcal{O}(H \cdot |\mathcal{R}| \cdot \theta_e \cdot \theta_x \cdot f)$$

Factor ${}^{|\mathcal{R}| \cdot f} \Big/ {}_{\frac{|\mathcal{R}|}{\theta_x} + |Call|}$ improvement

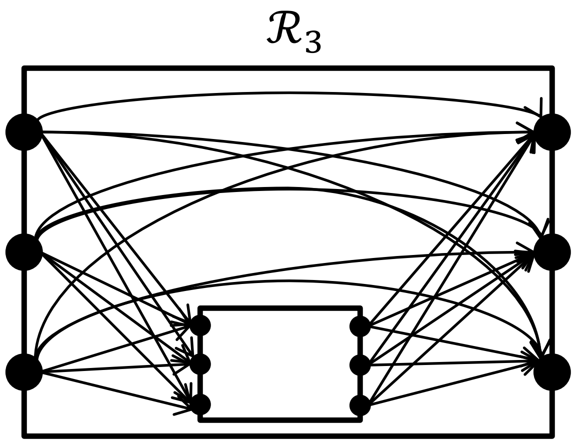
Empirical Evaluation

RSM-based algorithm vs. PDS-based algorithm

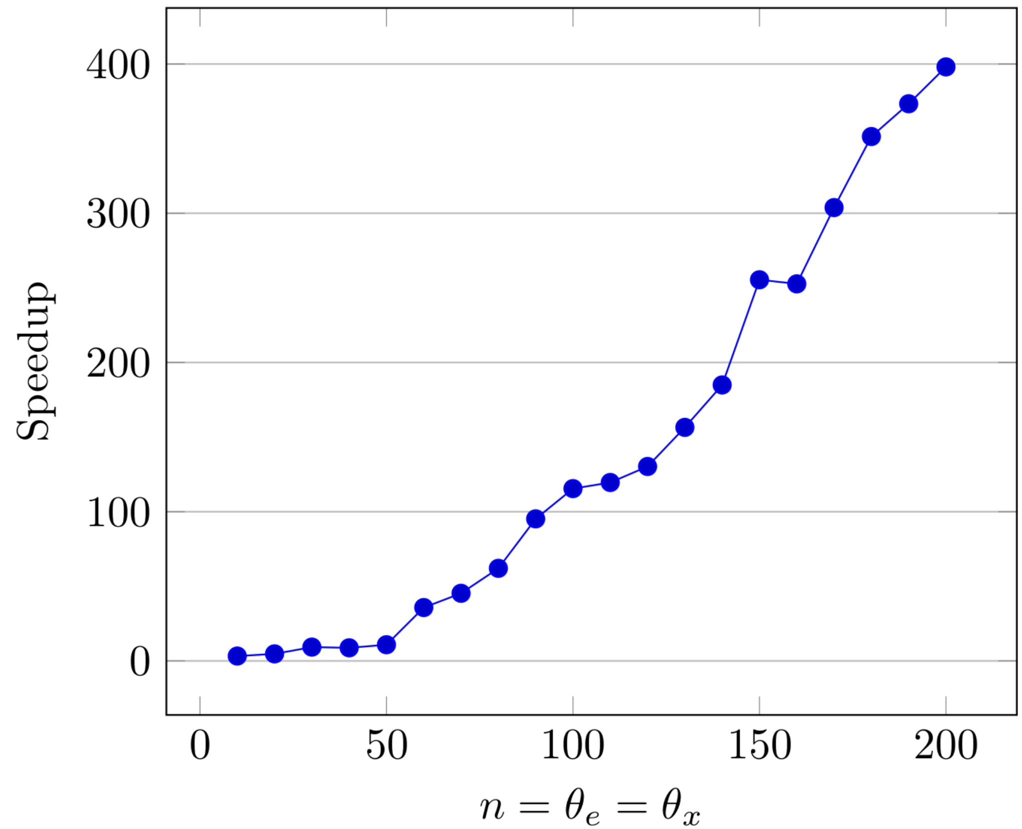
Our tool jMoped

1. Scaling on artificial examples
2. Interprocedural program analysis problems

A Family of Dense RSMs

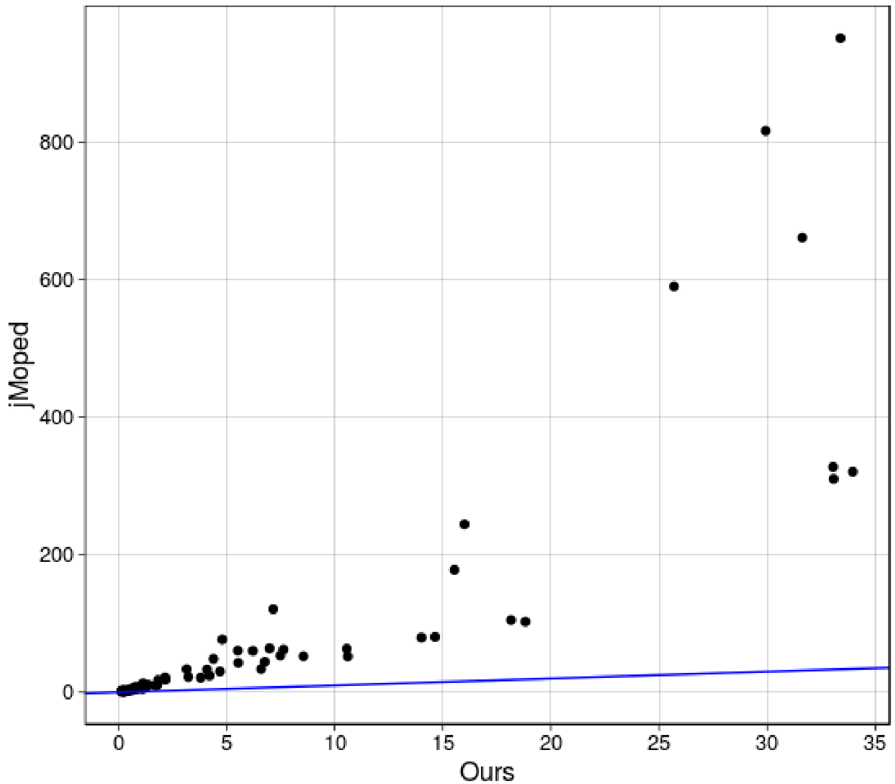


$$|\mathcal{R}| \cdot f \Big/ \frac{|\mathcal{R}|}{\theta_x} + |Call| = \frac{n^2 \cdot 1}{\frac{n^2}{n} + n} = \boldsymbol{n}$$

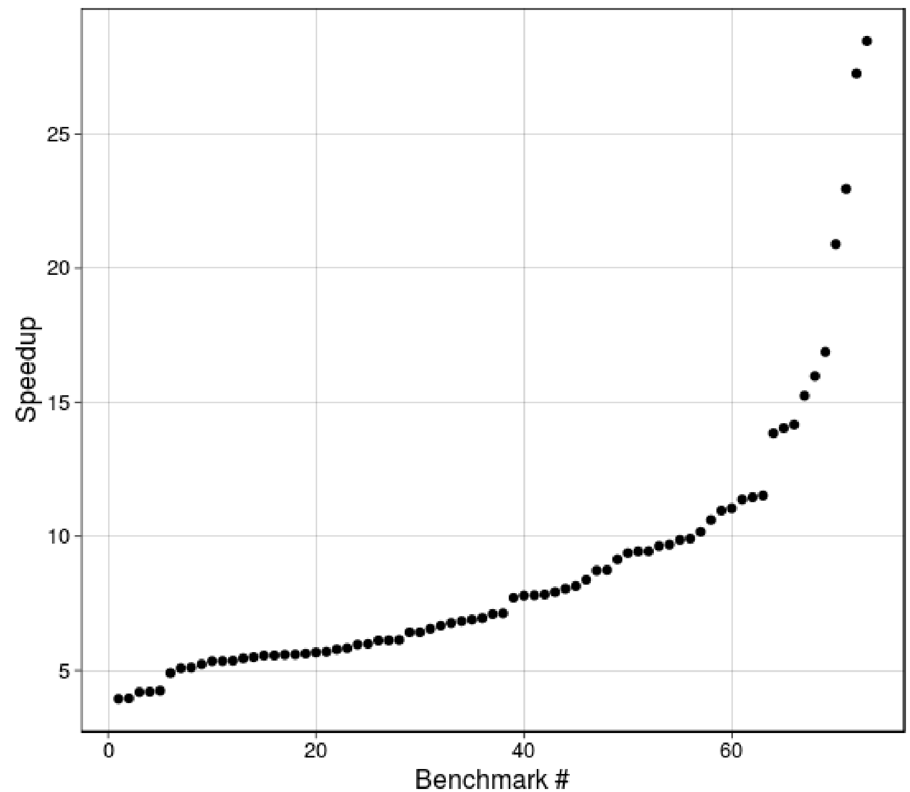


Boolean Programs from SLAM/SDV

Absolute runtime (seconds)



Relative speedup



Our Solution

1. Configuration automata

   Representation structures for sets of RSM configurations [BEM'97]

   Initial automaton C, s.t. $\mathcal{L}(C) = S$

2. Dynamic programming algorithm

   Compute C^*, representing reachable configurations and distances

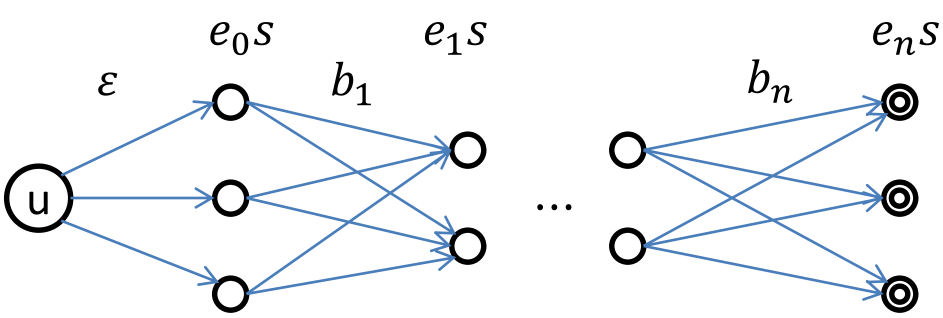
   Key: Entry-to-Exit summaries [ABEGRY'05]

3. Distance extraction algorithms

   Query configuration/superconfiguration/node distances from C^*

Distance Extraction

- Configuration distance for $\langle u, b_1 \cdots b_n \rangle$



Dynamic programming: $\mathcal{O}(n \cdot \theta_e^2)$

- Superconfiguration distance for $\langle u, \mathcal{M}_1 \cdots \mathcal{M}_n \rangle$

Replace $e \xrightarrow{b} e'$ with $e \xrightarrow{\mathcal{M}} e'$ where $\mathcal{M}$ is module of e'

- Node distance

Traditional single-source distance problem

Distances over Small Semirings

W^{θ_e} vector

C^* transitions $u \xrightarrow{\varepsilon} e$

$W^{\theta_e \times \theta_e}$ matrix

C^* transitions labeled b_i

$$1_{\theta_e} \cdot A_u \cdot A_{b_1} \cdots A_{b_n} \cdot 1_{\theta_e}^T$$

- Constant size semiring (Mailman's speedup)

$$\mathcal{O}\left(n \cdot \frac{\theta_e^2}{\log \theta_e}\right)$$

- Size $|W|$ semiring (William's speedup)

for $\varepsilon > 0$, $\mathcal{O}\left(n \cdot \frac{\theta_e^2}{\varepsilon^2 \log^2 \theta_e}\right)$ (some preprocessing)

- Binary semiring (Four-Russians speedup)

$$\mathcal{O}\left(|\mathcal{R}| \cdot \theta_e \cdot \frac{n}{\log n}\right)$$

Summary

- Faster interprocedural analysis (RSM > PDS)

- Configuration automata
  → Saturation algorithm (summaries)
  → Distance extraction

- More in the paper
  - Further distance extraction speedups
  - Implications for context-bounded analysis (concurrency)

Algorithm 1: `ConfDist`

Input: RSM $\mathcal{R}$ and $\mathcal{R}$-automaton $\mathcal{C}$ with $\ell(t) = \overline{1}$ for all transitions t in $\mathcal{C}$
Output: $\mathcal{R}$-automaton $\mathcal{C}_{post^*}$ with $\mathcal{C}_{post^*}(c) = d(\mathcal{L}(\mathcal{C}), c)$ for all configurations c

preprocess $\mathcal{C}$ as described in the main text
`// Initialization of worklist and summary function`
$\mathsf{WL} := \{t = q \xrightarrow{\varepsilon} q' \mid q \in I \text{ and } \ell(t) = \overline{1}\}$
$\mathsf{sum}(\langle e, m_e\rangle, x) := \overline{0}$ for all states $\langle e, m_e\rangle$ and $x \in Ex$
`// Main loop`
while $\mathsf{WL} \neq \emptyset$ **do**
    extract $t_{\mathcal{C}}$ from WL
    if $t_{\mathcal{C}} = \langle u, m_u\rangle \xrightarrow{\varepsilon} \langle e, m_e\rangle$ **then**
        let $\mathcal{M}_i$ be the module of node u
        `// Internal transitions from` u
        foreach $t_{\mathcal{R}} = \langle u, u'\rangle \in \delta_i$ *where* $u' \in In_i$ **do**
            `Relax`$(\langle u', \widehat{m}\rangle \xrightarrow{\varepsilon} \langle e, m_e\rangle, \ell(t_{\mathcal{C}}) \otimes w_i(t_{\mathcal{R}}))$
        `// Call transitions from` u
        foreach $t_{\mathcal{R}} = \langle u, \langle b, e'\rangle\rangle \in \delta_i$ **do**
            `Relax`$(\langle e', \widehat{m}\rangle \xrightarrow{b} \langle e, m_e\rangle, \ell(t_{\mathcal{C}}) \otimes w_i(t_{\mathcal{R}}))$
            add $\langle e', \widehat{m}\rangle \xrightarrow{\varepsilon} \langle e', \widehat{m}\rangle$ to WL, if it was never added before
        `// Exit transitions from` u
        foreach $t_{\mathcal{R}} = \langle u, x\rangle \in \delta_i$ *where* $x \in Ex_i$ **do**
            if $\mathsf{sum}(\langle e, m_e\rangle, x) \not\sqsubseteq \ell(t_{\mathcal{C}})$ **then**
                $\mathsf{sum}(\langle e, m_e\rangle, x) := \mathsf{sum}(\langle e, m_e\rangle, x) \oplus \ell(t_{\mathcal{C}})$
                foreach $\langle e, m_e\rangle \xrightarrow{b/v} \langle e', m_{e'}\rangle$ **do**
                    `Relax`$(\langle\langle b, x\rangle, \widehat{m}\rangle \xrightarrow{\varepsilon} \langle e', m_{e'}\rangle, v \otimes \mathsf{sum}(\langle e, m_e\rangle, x))$
    else if $t_{\mathcal{C}} = \langle e, m_e\rangle \xrightarrow{b} \langle e', m_{e'}\rangle$ **then**
        let $\mathcal{M}_i$ be the module of node e
        `// Using entry-to-exit summaries`
        foreach $x \in Ex_i$ **do**
            `Relax`$(\langle\langle b, x\rangle, \widehat{m}\rangle \xrightarrow{\varepsilon} \langle e', m_{e'}\rangle, \ell(t_{\mathcal{C}}) \otimes \mathsf{sum}(\langle e, \widehat{m}\rangle, x)$

Procedure `Relax`(t, v)
    if $\ell(t) \neq \ell(t) \oplus v$ **then**
        $\ell(t) := \ell(t) \oplus v$
        add t to WL