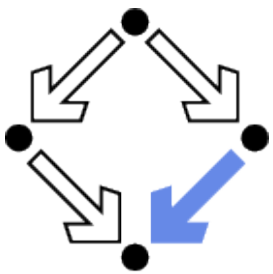
Measuring the Gap: Algorithmic Approximation Bounds for the Space Complexity of Stream Specifications

David M. Cerna and Wolfgang Schreiner

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria



April 8^{th}, 2017

JOHANNES KEPLER
UNIVERSITY LINZ

Introduction

- LogicGuard: A coordination language for runtime monitoring of network traffic.
- Stream monitors are written in a fragment of predicate logic.

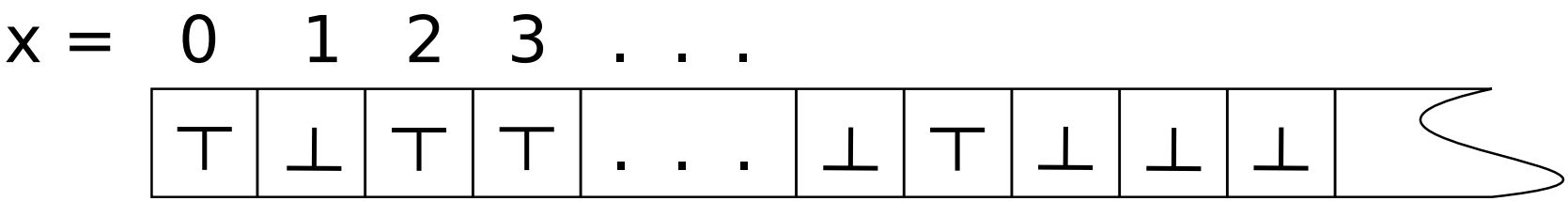
Introduction

- LogicGuard: A coordination language for runtime monitoring of network traffic.
- Stream monitors are written in a fragment of predicate logic.
- Monitor instances are evaluated using an operational semantics.
- Violations, monitor instances evaluating to false, are flagged.

Introduction

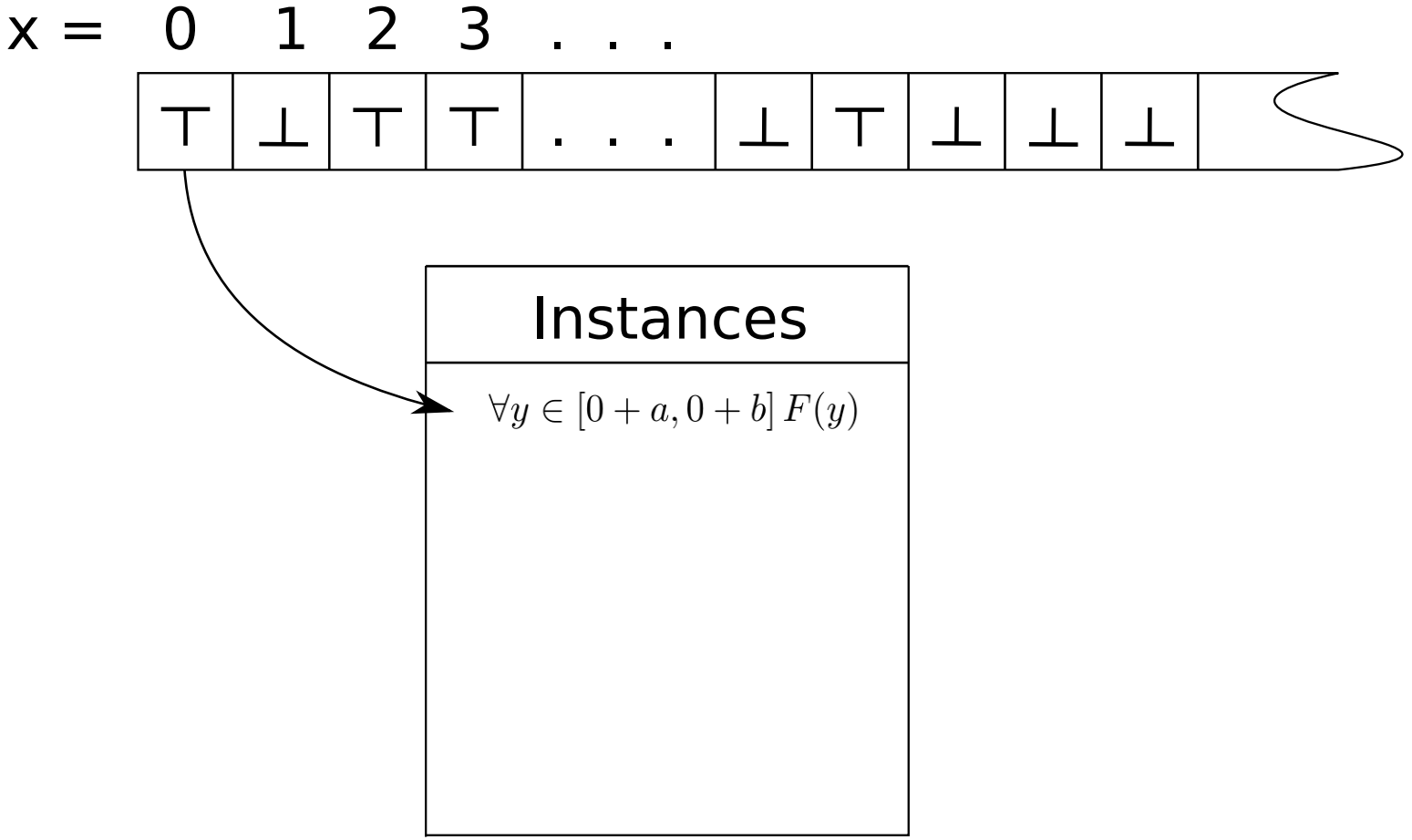
- LogicGuard: A coordination language for runtime monitoring of network traffic.
- Stream monitors are written in a fragment of predicate logic.
- Monitor instances are evaluated using an operational semantics.
- Violations, monitor instances evaluating to false, are flagged.
- The work presented here is a solution and analysis of the following problem:

The Space Problem



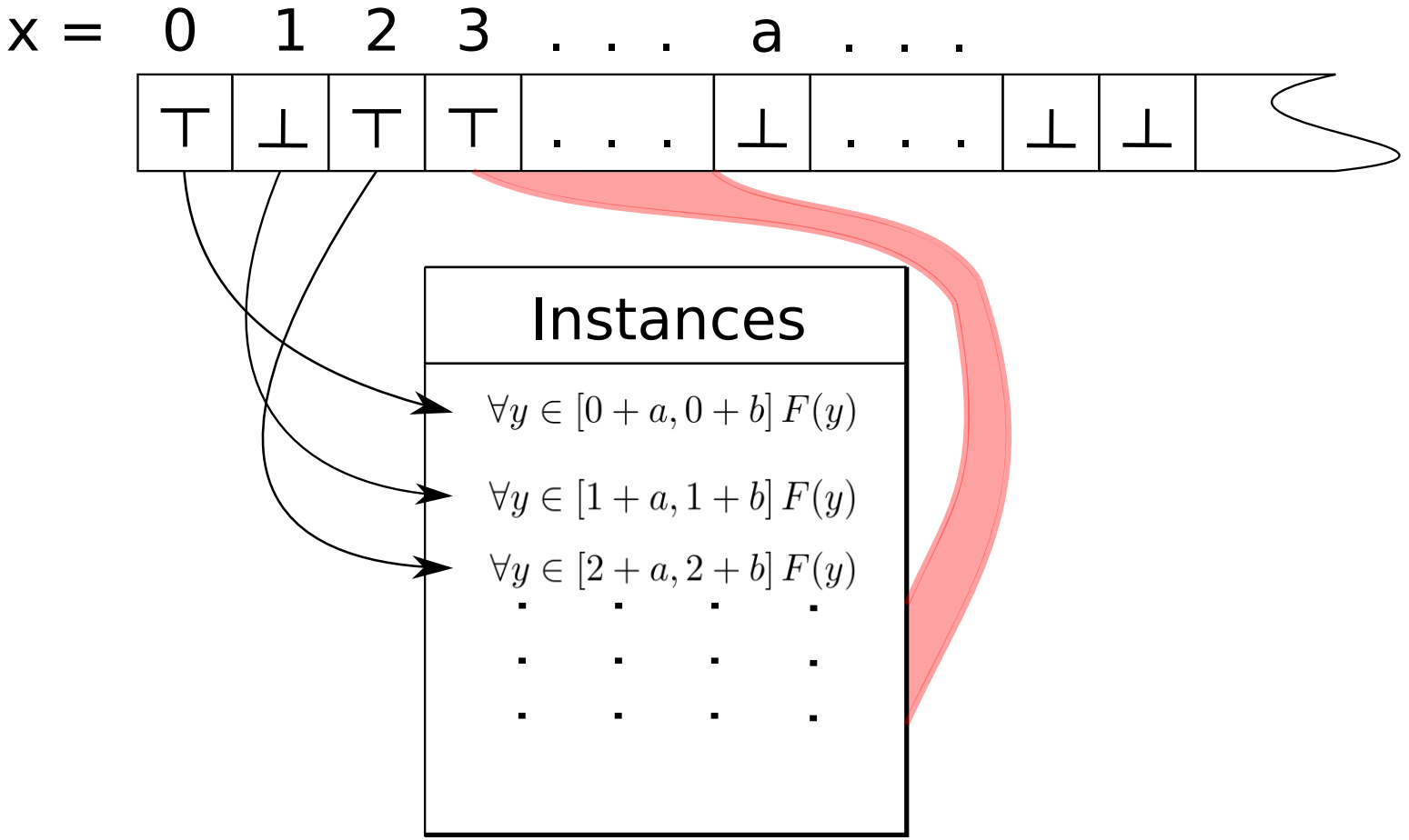
The Space Problem

Monitor: $\forall_{0 \leq x} \forall y \in [x + a, x + b]\, F(y)$



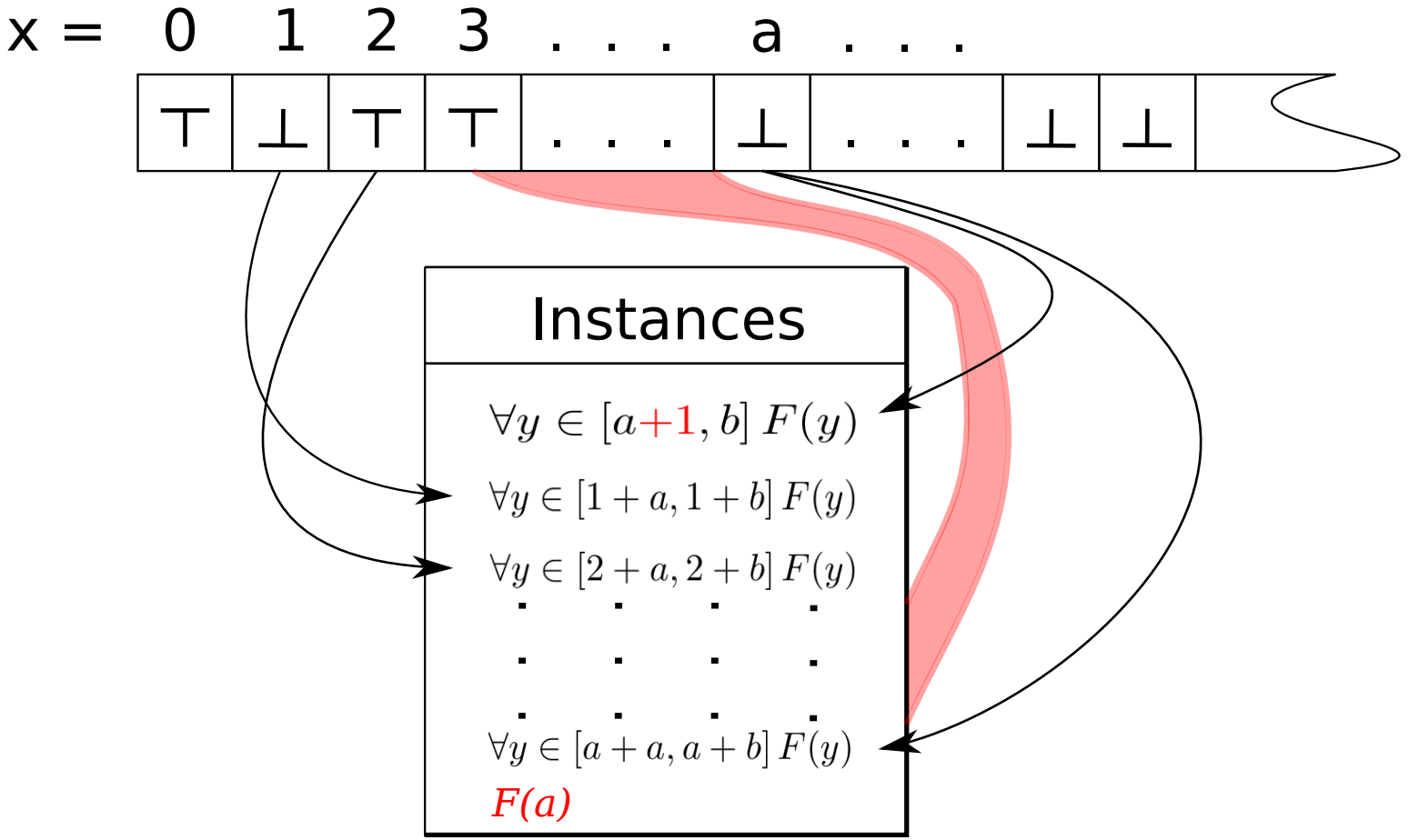
The Space Problem

Monitor: $\forall_{0 \leq x} \forall y \in [x + a, x + b]\, F(y)$



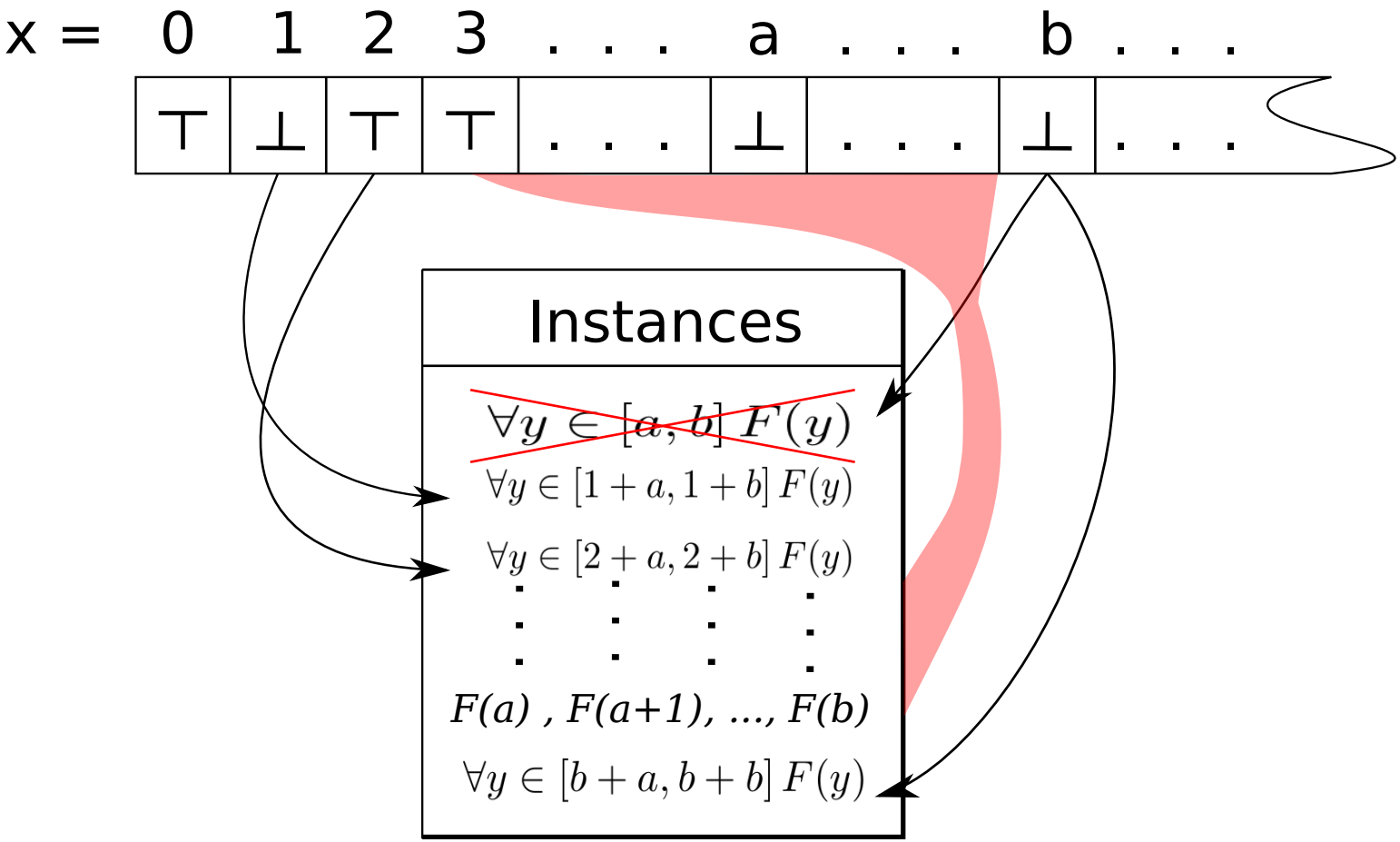
The Space Problem

Monitor: $\forall_{0 \leq x} \forall y \in [x+a, x+b]\, F(y)$



The Space Problem

Monitor: $\forall_{0 \leq x} \forall y \in [x+a, x+b]\, F(y)$



- We will refer to this measurement of space complexity as runtime representation size.

Previous Work

- Previous work focused on analysis of the stream history [Kutsia and Schreiner 2014], and space complexity results [Cerna et al. 2016a-b].
- In this work we perform an analysis of the previous results and provide a categorization of specifications into two types based on the our analysis.

The Core Language

$$
\begin{array}{rcl}
M & ::= & \forall_{0 \leq V} : F. \\
F & ::= & @V \mid \neg F \mid F \ \wedge \ F \mid F \ \& \ F \mid \forall_{V \in [B,B]} : F. \\
B & ::= & 0 \mid \infty \mid V \mid B \pm N. \\
V & ::= & x \mid y \mid z \mid \ \ldots \\
N & ::= & 0 \mid 1 \mid 2 \mid \ \ldots
\end{array}
$$

$$\forall_{0 \leq x} : \forall_{y \in [x+1, x+5]} : ((\forall_{z \in [y, x+3]} : \neg @z \ \& \ @z) \ \& \ G(x, y))$$

$$G(x, y) = \forall_{w \in [x+2, y+2]} : (\neg @y \ \& \ (\forall_{m \in [y, w]} : \neg @x \ \& \ @m))$$

Nested Variables

— Lets consider the example monitor again:

$$\forall_{0 \leq x} : \forall_{y \in [x+1, x+5]} : ((\forall_{z \in [y, x+3]} : \neg @z \ \& \ @z) \ \& \ G(x, y))$$
$$G(x, y) = \forall_{w \in [x+2, y+2]} : (\neg @y \ \& \ (\forall_{m \in [y, w]} : \neg @x \ \& \ @m))$$

Nested Variables

- Lets consider the example monitor again:

$$\forall_{0 \leq x} : \forall_{y \in [x+1, x+5]} : ((\forall_{z \in [y, x+3]} : \neg @z \ \& \ @z) \ \& \ G(x, y))$$
$$G(x, y) = \forall_{w \in [x+2, y+2]} : (\neg @y \ \& \ (\forall_{m \in [y, w]} : \neg @x \ \& \ @m))$$

- In [Cerna et al. 2016a] we developed the concept of dominating monitor tranformation to remove nested variables.
- It was also shown that the space requirements of a dominating monitor upper bound the original monitor.

Example Monitor Transformed

— The dominating monitor of

$$\forall_{0\leq x} : \forall_{y\in[x+1,x+5]} : ((\forall_{z\in[y,x+3]} : \neg @z \ \&\ @z) \ \&\ G(x,y))$$
$$G(x,y) = \forall_{w\in[x+2,y+2]} : (\neg @y \ \&\ (\forall_{m\in[y,w]} : \neg @x \ \&\ @m))$$

is the following monitor

$$\forall_{0\leq x} : \forall_{y\in[x+1,x+5]} : ((\forall_{z\in[x+1,x+3]} : \neg @z \ \&\ @z) \ \&\ G(x,y))$$
$$G(x,y) = \forall_{w\in[x+2,x+7]} : (\neg @y \ \&\ (\forall_{m\in[x+1,x+7]} : \neg @x \ \&\ @m))$$

Example Monitor Transformed

— The dominating monitor of

$$\forall_{0\leq x} : \forall_{y\in[x+1,x+5]} : ((\forall_{z\in[y,x+3]} : \neg @z \;\&\; @z) \;\&\; G(x,y))$$
$$G(x,y) = \forall_{w\in[x+2,y+2]} : (\neg @y \;\&\; (\forall_{m\in[y,w]} : \neg @x \;\&\; @m))$$

is the following monitor

$$\forall_{0\leq x} : \forall_{y\in[x+1,x+5]} : ((\forall_{z\in[x+1,x+3]} : \neg @z \;\&\; @z) \;\&\; G(x,y))$$
$$G(x,y) = \forall_{w\in[x+2,x+7]} : (\neg @y \;\&\; (\forall_{m\in[x+1,x+7]} : \neg @x \;\&\; @m))$$

— This transformation is a key component of the space requirements algorithm of [Cerna et al. 2016b].

Dealing with the Runtime Representation Size

- Runtime representation size equals instances kept in memory while evaluating a monitor.

Dealing with the Runtime Representation Size

- Runtime representation size equals instances kept in memory while evaluating a monitor.
- Consider the following simple monitor:

$$\forall_{0 \leq x} : \forall_{y \in [x,x+4]} : \forall_{z \in [x,x+4]} : \forall_{r \in [x,x+4]} : @r$$

Dealing with the Runtime Representation Size

- Runtime representation size equals instances kept in memory while evaluating a monitor.
- Consider the following simple monitor:

$$\forall_{0 \leq x} : \forall_{y \in [x,x+4]} : \forall_{z \in [x,x+4]} : \forall_{r \in [x,x+4]} : @r$$

- We can simplify its representation:

$$[0,4]\,[0,4]\,[0,4]$$

Dealing with the Runtime Representation Size

- Runtime representation size equals instances kept in memory while evaluating a monitor.
- Consider the following simple monitor:

$$\forall_{0 \leq x} : \forall_{y \in [x,x+4]} : \forall_{z \in [x,x+4]} : \forall_{r \in [x,x+4]} : @r$$

- We can simplify its representation:

$$[0,4]\,[0,4]\,[0,4]$$

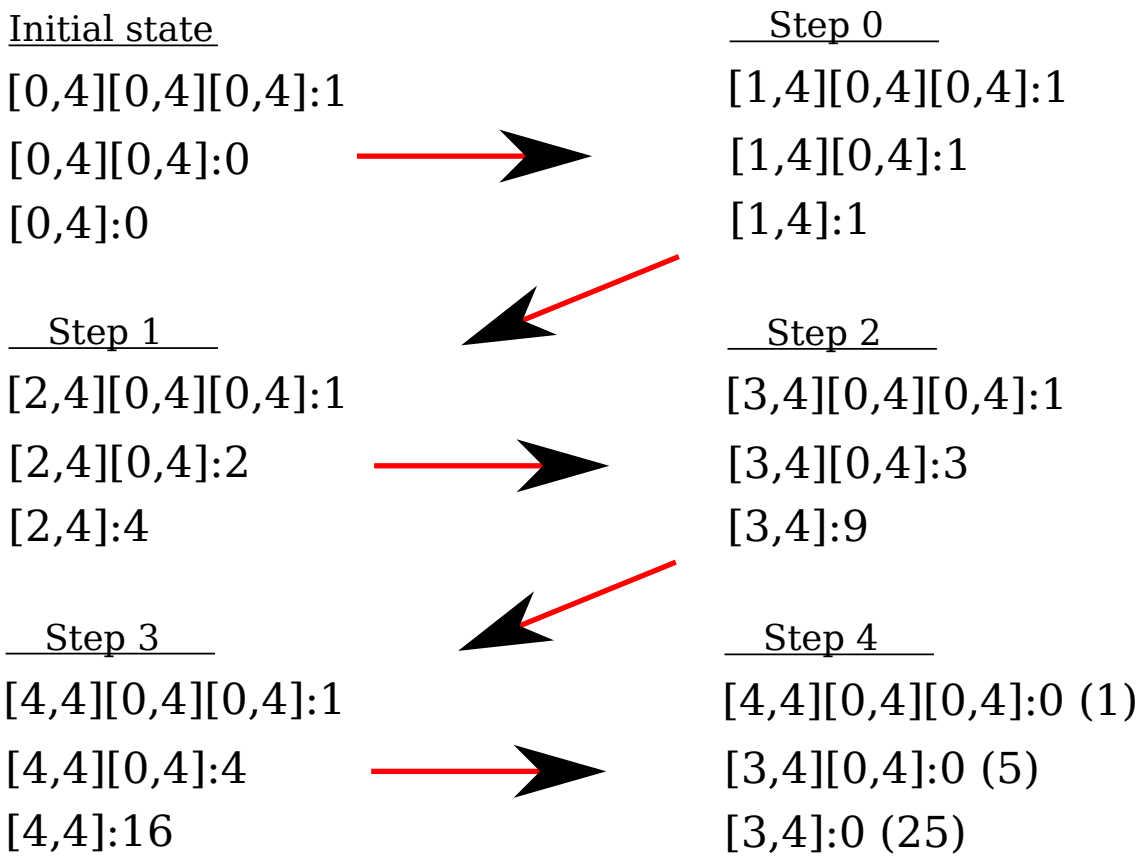
- Now let us consider its behaviour as it is evaluated.

Initial State

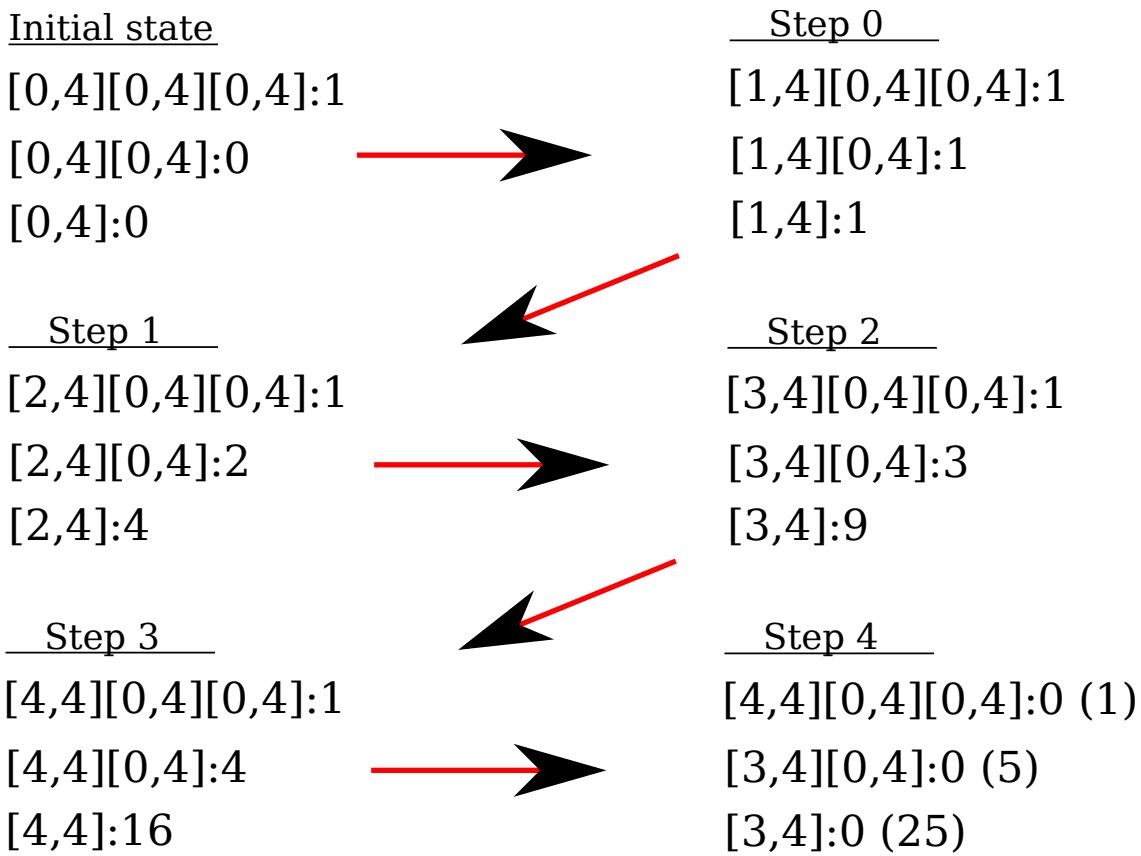
Initial state

[0,4][0,4][0,4]:1	$\forall_{y\in[0,4]} : \forall_{z\in[0,4]} : \forall_{r\in[0,4]} : @r$
[0,4][0,4]:0	$\forall_{z\in[0,4]} : \forall_{r\in[0,4]} : @r$
[0,4]:0	$\forall_{r\in[0,4]} : @r$

Evaluation


Initial state
[0,4][0,4][0,4]:1
[0,4][0,4]:0
[0,4]:0
Step 0
[1,4][0,4][0,4]:1
[1,4][0,4]:1
[1,4]:1
Step 1
[2,4][0,4][0,4]:1
[2,4][0,4]:2
[2,4]:4
Step 2
[3,4][0,4][0,4]:1
[3,4][0,4]:3
[3,4]:9
Step 3
[4,4][0,4][0,4]:1
[4,4][0,4]:4
[4,4]:16
Step 4
[4,4][0,4][0,4]:0 (1)
[3,4][0,4]:0 (5)
[3,4]:0 (25)

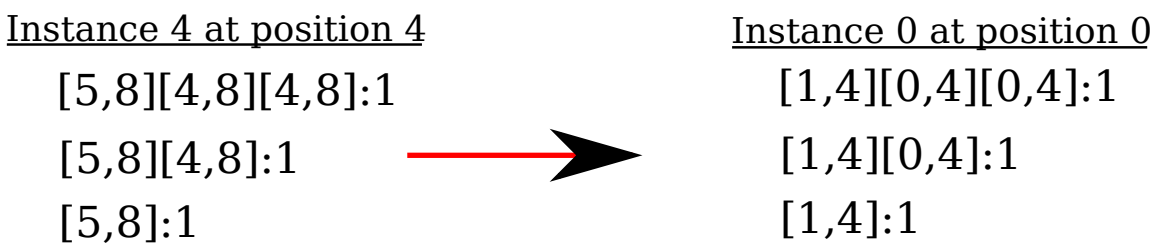
Evaluation



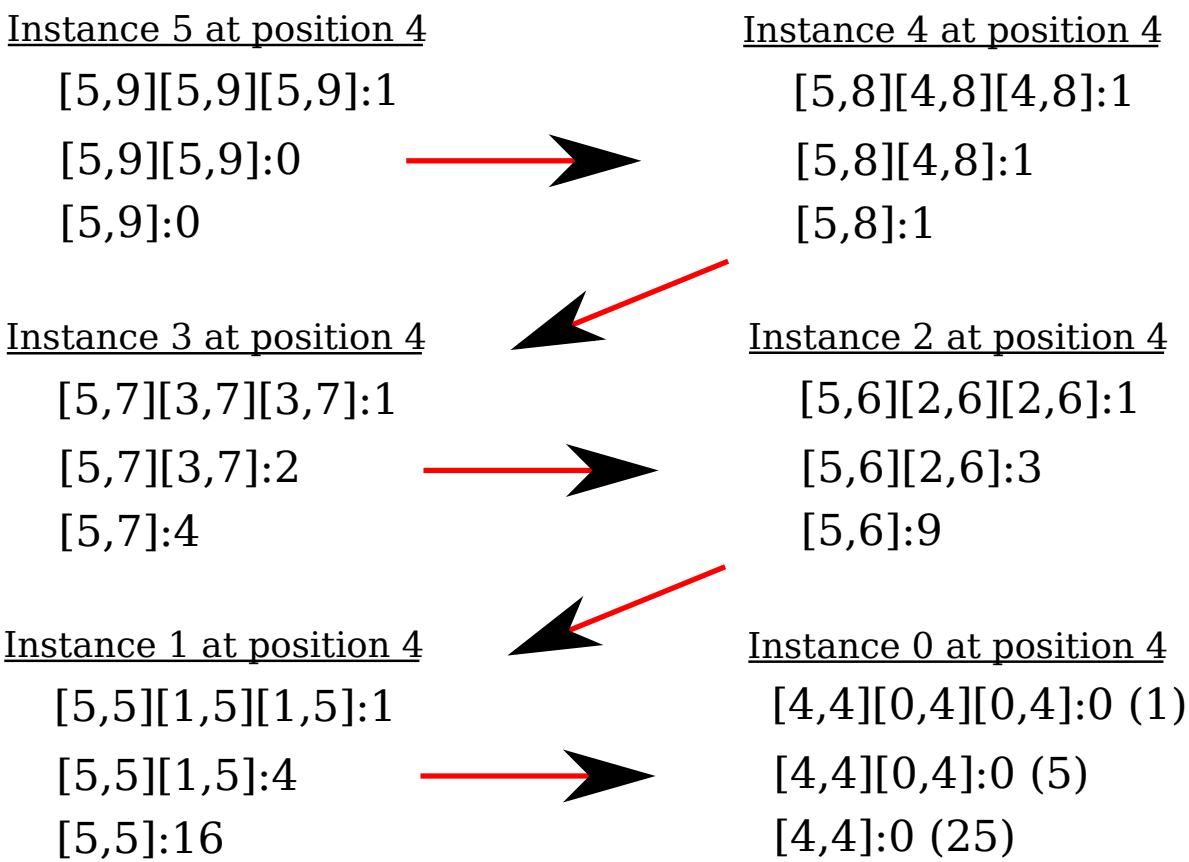
- Notice that we did not add new instances.
- How does this relate to true evaluation?

Instance to Position Mapping

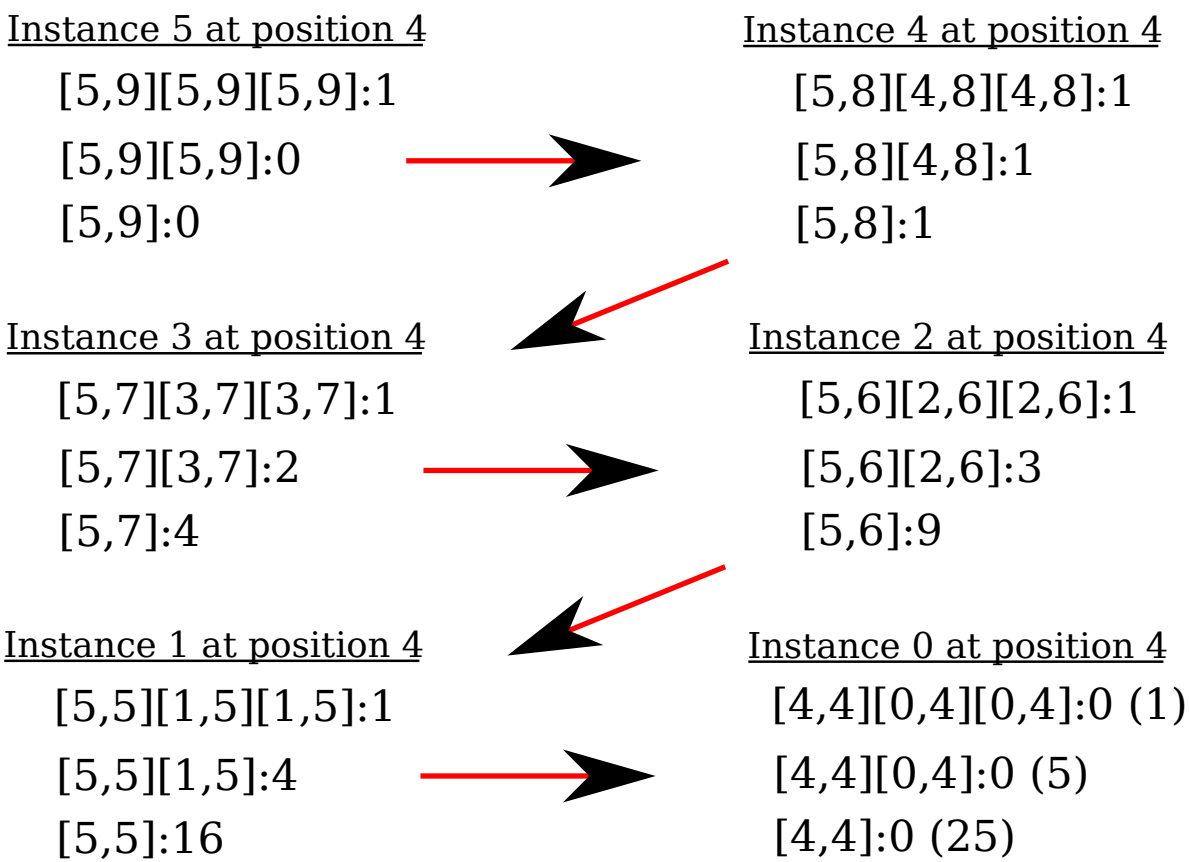
- It turns out that there is a mapping from the evaluation of a single instance at various positions to the evaluation of multiple instances at a single position.



Instance to Position Mapping

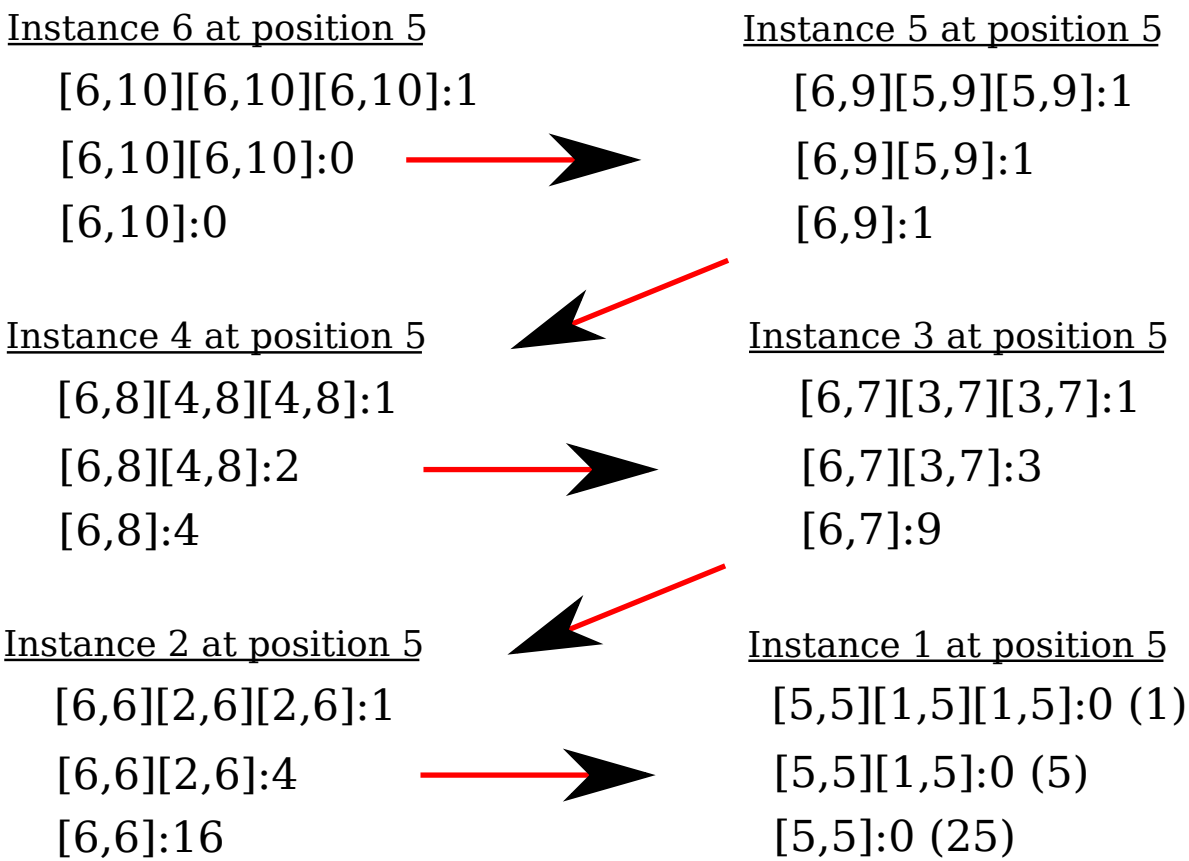


Instance to Position Mapping

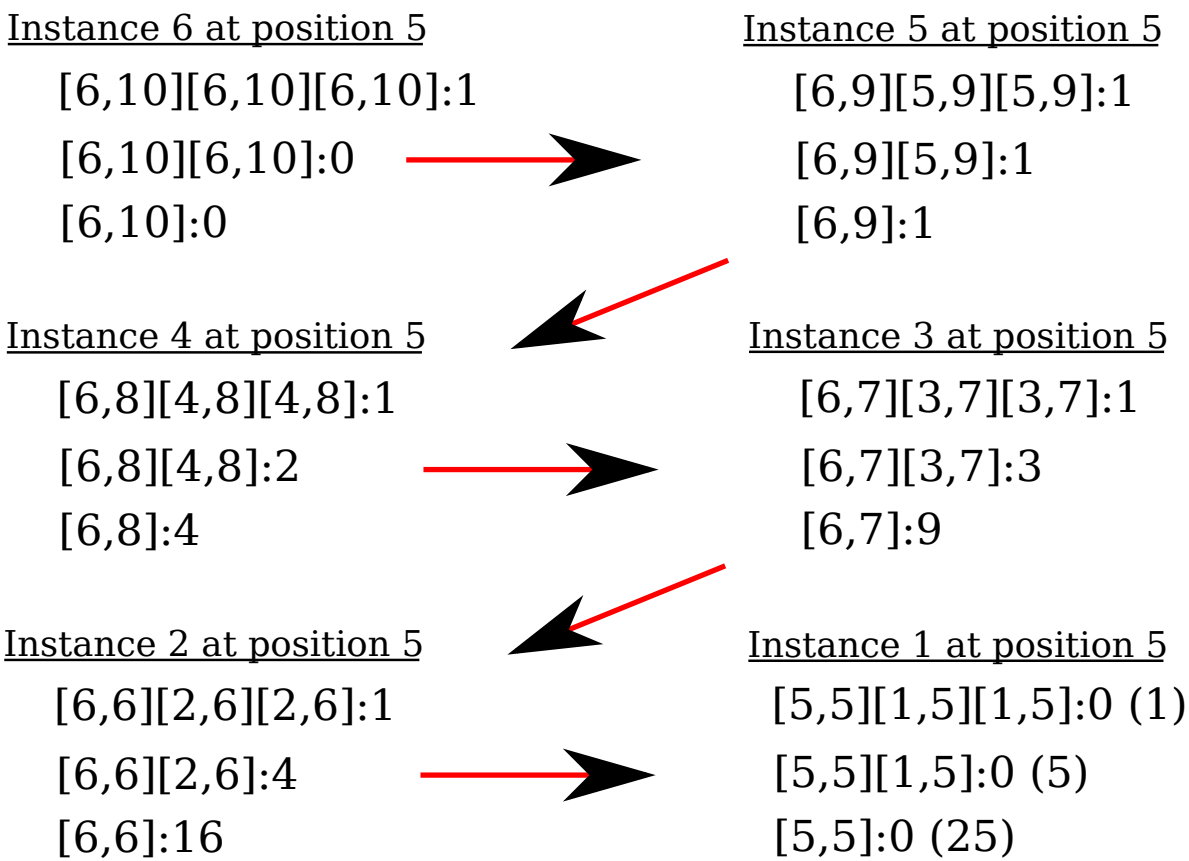


— Notice that going to the next position does not change anything

Instance to Position Mapping, Next Position



Instance to Position Mapping, Next Position

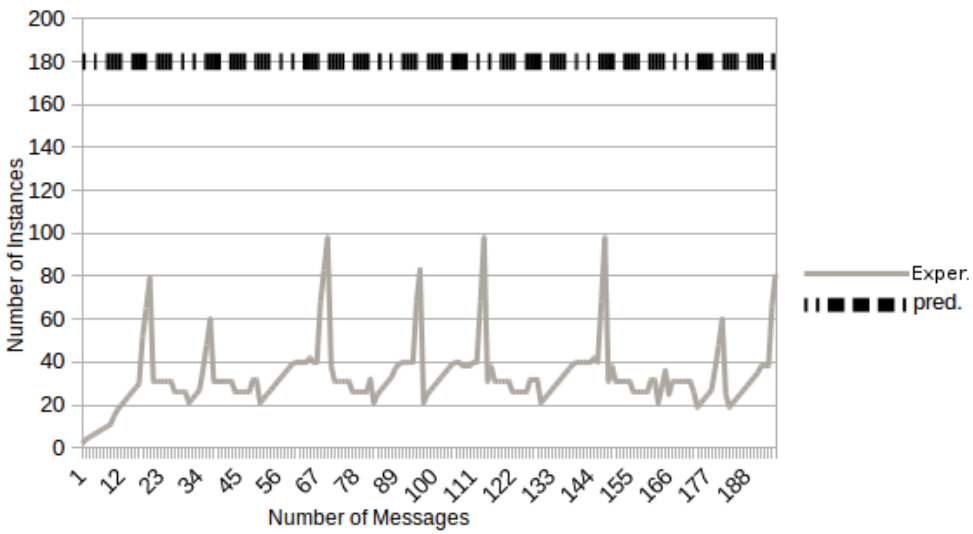


- Essentially, we only need to look at the behaviour of one instance up to the largest upper bound. This is the key to the algorithm.

Experimental Results: Realistic Monitoring

– We ran the algorithm on the following monitor written in the full specification language:

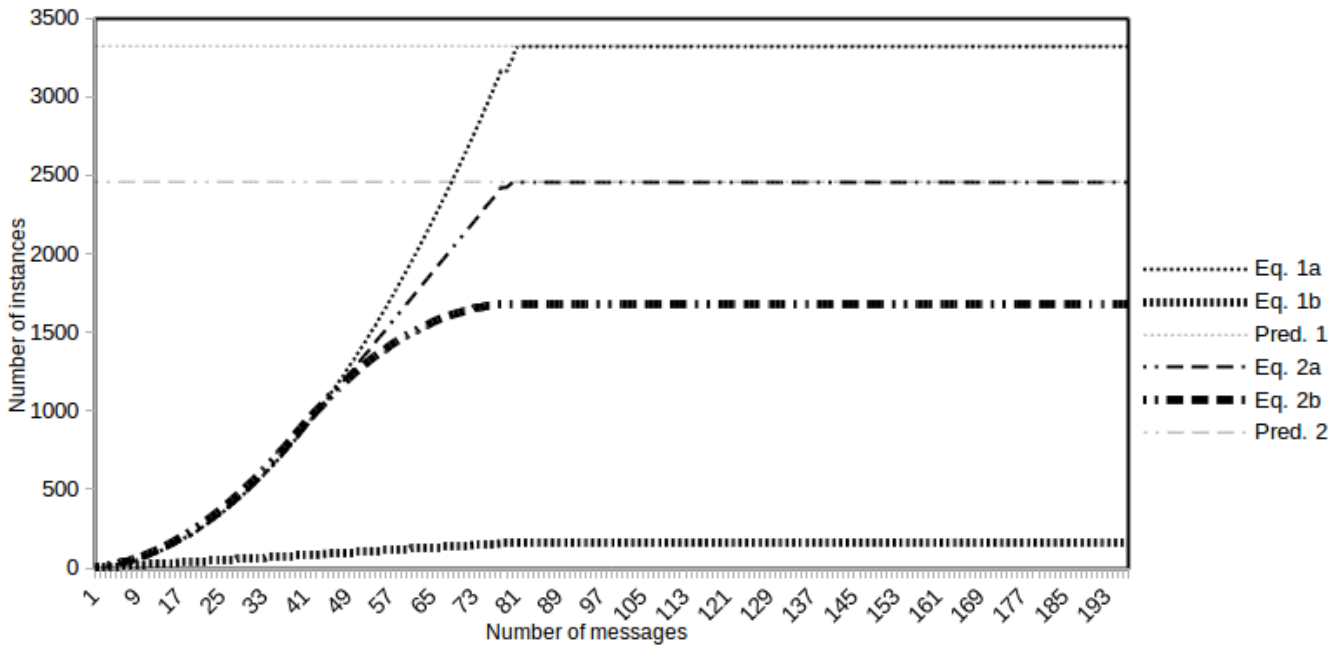
```
type int; type message; stream<int> IP;
stream<int> S = stream<IP> x satisfying @x>=0 :
      value[seq,@x,plus]<IP> y with x < _ <=# x+10000: @y;
monitor<S> M = monitor<S> x :
      forall<S> y with x < _ <=# x+15000:
        exists<S> z with y < _ <=# y+4000: IsEven(#z);
```



Experimental Results: Interesting

– We also ran the algorithm on the following monitor specifications:

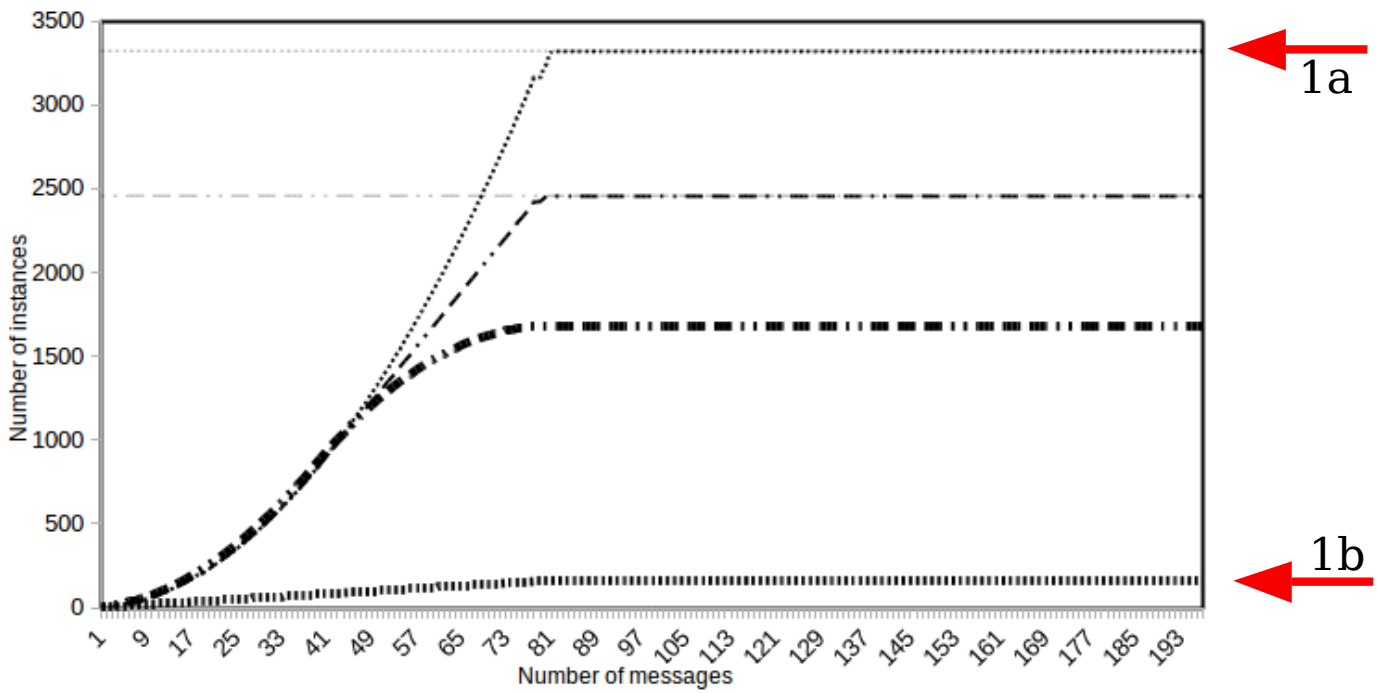
$$\forall_{0\leq x} : \forall_{y\in[x,x+80]} : \forall_{z\in[x,x+80]} : @z \quad (1a)$$
$$\forall_{0\leq x} : \forall_{y\in[x,x+80]} : \forall_{z\in[x,y]} : @z \quad (1b)$$
$$\forall_{0\leq x} : \forall_{y\in[x,x+40]} : \forall_{z\in[x,x+80]} : @z \quad (2a)$$
$$\forall_{0\leq x} : \forall_{y\in[x,x+40]} : \forall_{z\in[x,y+40]} : @z \quad (2b)$$



Experimental Results: Interesting

– We also ran the algorithm on the following monitor specifications:

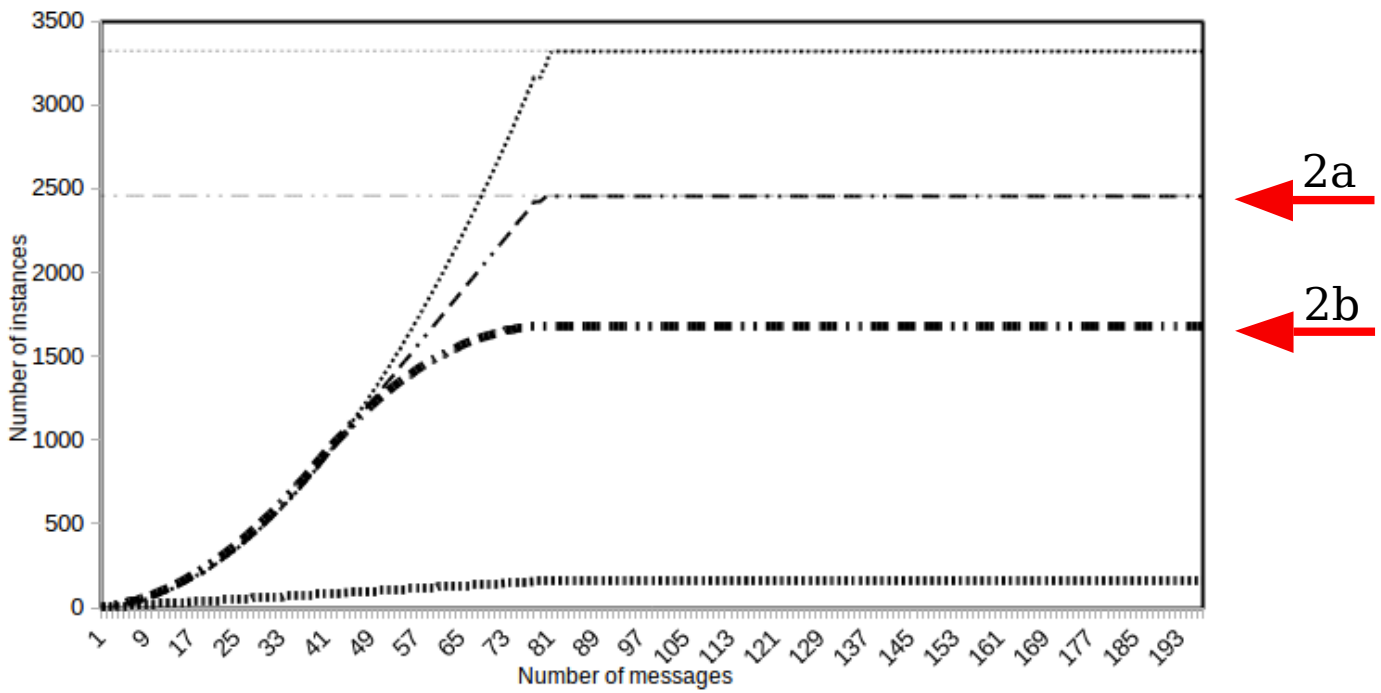
$$\forall_{0\leq x} : \forall_{y\in[x,x+80]} : \forall_{z\in[x,x+80]} : @z \quad (1a)$$
$$\forall_{0\leq x} : \forall_{y\in[x,x+80]} : \forall_{z\in[x,y]} : @z \quad (1b)$$
$$\forall_{0\leq x} : \forall_{y\in[x,x+40]} : \forall_{z\in[x,x+80]} : @z \quad (2a)$$
$$\forall_{0\leq x} : \forall_{y\in[x,x+40]} : \forall_{z\in[x,y+40]} : @z \quad (2b)$$



Experimental Results: Interesting

– We also ran the algorithm on the following monitor specifications:

$$\forall_{0\leq x} : \forall_{y\in[x,x+80]} : \forall_{z\in[x,x+80]} : @z \quad (1a)$$
$$\forall_{0\leq x} : \forall_{y\in[x,x+80]} : \forall_{z\in[x,y]} : @z \quad (1b)$$
$$\forall_{0\leq x} : \forall_{y\in[x,x+40]} : \forall_{z\in[x,x+80]} : @z \quad (2a)$$
$$\forall_{0\leq x} : \forall_{y\in[x,x+40]} : \forall_{z\in[x,y+40]} : @z \quad (2b)$$



Experimental Results: Interesting

- Notice that the prediction for (1a) is roughly 21 times greater than the actual complexity of (1b).

Experimental Results: Interesting

- Notice that the prediction for (1a) is roughly 21 times greater than the actual complexity of (1b).
- Also the prediction for (2a) is roughly 1.5 times greater than the actual complexity of (2b).

Experimental Results: Interesting

- Notice that the prediction for (1a) is roughly 21 times greater than the actual complexity of (1b).
- Also the prediction for (2a) is roughly 1.5 times greater than the actual complexity of (2b).
- Strange?

Experimental Results: Interesting

- Notice that the prediction for (1a) is roughly 21 times greater than the actual complexity of (1b).
- Also the prediction for (2a) is roughly 1.5 times greater than the actual complexity of (2b).
- Strange? We thought so too.

Distinguishing Dominating Like Specifications

- Naively, (1b) looks less like (1a) than (2b) looks like (2a).

Distinguishing Dominating Like Specifications

- Naively, (1b) looks less like (1a) than (2b) looks like (2a).
- In some sense we can say (2b) is a dominating-like monitor.

Distinguishing Dominating Like Specifications

- Naively, (1b) looks less like (1a) than (2b) looks like (2a).
- In some sense we can say (2b) is a dominating-like monitor.
- Can this property be formalized in such a way allowing one to distinguish dominating-like monitors from the rest.

Distinguishing Dominating Like Specifications

- Naively, (1b) looks less like (1a) than (2b) looks like (2a).
- In some sense we can say (2b) is a dominating-like monitor.
- Can this property be formalized in such a way allowing one to distinguish dominating-like monitors from the rest.
- The key seems to be how the variable nesting is constructed.

Relationship Between Upper Bounds: Good Monitors

- It turns out that the relationship between the constants within the quantifier intervals plays an important role.
- Let m be a monitor and $a_1, b_1, a_2, b_2 \in \mathbb{Z}$ such that $m = \forall_{0 \leq x} : \forall_{y \in [x+a_1, x+b_1]} : \forall_{z \in [x+a_2, y+b_2]} : @z.$

Relationship Between Upper Bounds: Good Monitors

- It turns out that the relationship between the constants within the quantifier intervals plays an important role.
- Let m be a monitor and $a_1, b_1, a_2, b_2 \in \mathbb{Z}$ such that $m = \forall_{0 \leq x} : \forall_{y \in [x+a_1, x+b_1]} : \forall_{z \in [x+a_2, y+b_2]} : @z.$
- We place the following constraints on the constants:
  - $0 < b_1$ and $a_1 \leq b_1$
  - $d = min_{i \in [a_1, b_1]} \{0 < b_2 + i\}$
  - $a_2 \leq b_2 + d$, $k \geq 1$, and $b_1 = k + b_2 + d$

Relationship Between Upper Bounds: Good Monitors

- It turns out that the relationship between the constants within the quantifier intervals plays an important role.
- Let m be a monitor and $a_1, b_1, a_2, b_2 \in \mathbb{Z}$ such that $m = \forall_{0 \leq x} : \forall_{y \in [x+a_1, x+b_1]} : \forall_{z \in [x+a_2, y+b_2]} : @z.$
- We place the following constraints on the constants:
  - $0 < b_1$ and $a_1 \leq b_1$
  - $d = min_{i \in [a_1, b_1]} \{0 < b_2 + i\}$
  - $a_2 \leq b_2 + d$, $k \geq 1$, and $b_1 = k + b_2 + d$
- Then the ratio between the runtime representation size of the monitor and its dominating form is $O(k)$ (propositional in k).

Relationship Between Upper Bounds: Bad Monitors

- The following constraints pinpoint the dominating-like monitors.
- Let m be a monitor and $a_1, b_1, a_2, b_2 \in \mathbb{Z}$ such that $m = \forall_{0 \leq x} : \forall_{y \in [x+a_1, x+b_1]} : \forall_{z \in [x+a_2, y+b_2]} : @z.$
- We place the following constraints on the constants:
  - $0 < b_1$ and $a_1 \leq b_1$
  - $d = min_{i \in [a_1, b_1]} \{0 < b_2 + i\}$
  - $a_2 \leq b_2 + d$ and $b_1 \leq b_2 + d$
- Then the ratio between the runtime representation size of the monitor and its dominating form is $O(1)$ (constant).

Categorization of the Core Language

- The following Lemma is needed to extend the results to the core language.

Lemma

Let $m \in \mathbb{M}$, $a_1, b_1, a_2, b_2 \in \mathbb{Z}$, $x, y, z \in V$, *and* $Q, Q' \subset \mathbb{QT}$ *such that* $q \in Q$ *iff* $D(q) \in Q'$ *and* $|Q'|_{sc}$ *is a* $O(k_1^r)$*-approximation, where* $r \in \{0, 1\}$. *If*

$$D(QT(m)) = (x, 0, 0, (y, x+a_1, x+b_1, (z, x+a_2, x+\max\{a_1, b_1\}+b_2, \emptyset)))$$

is an $O(k_2^{r'})$*-approximation, where* $r' \in \{0, 1\}$, *of* $QT(m) = (x, 0, 0, (y, x + a_1, x + b_1, (z, x + a_2, y + b_2, \emptyset)))$, *then*

$$D(QT(m)) = (x, 0, 0, (y, x+a_1, x+b_1, (z, x+a_2, x+\max\{a_1, b_1\}+b_2, Q')))$$

is at most an $O(\max\{k_1, k_2\}^{\max\{r, r'\}})$*-approximation of* $QT(m) = (x, 0, 0, (y, x + a_1, x + b_1, (z, x + a_2, y + b_2, Q)))$.

Categorization of the Core Language

Definition

Let $q \in \mathbb{QT}$. We say that $q.\nu$ is related to $q.\nu.\mu$, where $q.\nu.\mu \neq q.\nu$, if $(q.\nu.\mu)_3 = y + b_2$, where $y \in V$ and $b_2 \in \mathbb{Z}$, and $(q.\nu)_1 = y$. We define the set of pairs $\mathbf{SR}_q$ as follows: $(q', q'') \in \mathbf{SR}_q$ if $\exists\ q.\nu = q'$ and $q'.\mu = q''$ such that $q' \neq q'.\mu$ and q' is related to q''.

Definition

Let $q \in \mathbb{QT}$. We say that a set $S \subseteq \mathbf{SR}_q$ is a relation chain of q if the undirected graph (V, S), where
$V = \{q \mid \exists r \in \mathbb{QT}((q, r) \in S \vee (r, q) \in S)\}$, is a connected path.

Theorem

Let $m \in \mathbb{M}$ and S the longest relation chain of $QT(m)$. Then $|QT(D(m))|_{sc}$ is at most an $O(n)$-approximation of $|QT(m)|_{sc}$, where n is dependent on $C_i(QT(m))$.

Future Work

- Recently, we investigated the time complexity of LogicGuard monitor specifications.
- For time complexity we count the number of variable assignments per message.
- Though this measure is closely related to runtime representation size it behaves quite differently.
- We plan to perform a similar analysis of monitor specifications based on our time complexity measure, that is a categorization of specifications similar to the one presented here.