Space Complexity of Operational Semantics for the LogicGuard Core Language [*]

David Cerna
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
David.Cerna@risc.uni-linz.ac.at

May 28, 2015

Abstract

In this work we provide a space complexity analysis of the core language of the LogicGuard framework. Our approach is to abstract away from the formula structure of the core language in order to construct invariants for a recursive function. This final recursive function provided an upper bound for the number of instances of the monitor that need to be kept in memory over any duration of time.

[*]Supported by the Austrian Research Promotion Agency (FFG) in the frame of the BRIDGE program by the projects 832207 "LogicGuard" and 846003 "LogicGuard II".

Contents

1 Introduction

The LogicGuard framework [2, 3] was designed to monitor network traffic by adding a monitor to a firewall which reports violations of security assertions. The language used to write the monitors is essentially a variant of predicate logic designed specifically to write monitors designed to observe network traffic. Rather then deciding the truth of the predicate logic monitors by providing a model, the model is instead generated real-time as packets are received by the firewall from the network. A specific semantics was constructed to deal with this real-time model building which is referred to as the *operational semantics.*

Previous work concerning properties of the operational semantics focused on the resource requirement of an arbitrary monitor [1]. By resource, we are referring to the amount of information which needs to be collected from the network in order to evaluate an instance of the given monitor. In this work we concern ourselves with the space complexity concerning the operational semantics [3] of the core language of the LogicGuard framework. Instead of a resource anaylsis, the space analysis is concerned with the problem, given access to the necessary resources, how much memory will be used to evaluate the given monitor. The core language, the language we will be focusing on in this work, can be written as a context-free grammar as described in Fig. 1; we refer to this grammar as **F**ull **F**ormula **g**rammar (FF_g).

To give a brief outline of the connectives used in the above grammars, & *sequential and*, i.e. first evaluate the left sub-formula and then the right, $\wedge$ *parallel and*, i.e. evaluate both sub-formula simultaneously, $\sim$ *negation*, and $\forall x_{[a,b]}\varphi(x)$ *interval quantification*, i.e. $\varphi(x)$ is true for all $x \in [a, b]$, where a and b are constructed from the term language of minimal arithmetic plus the symbol ∞. Interval quantification can also be written as $\forall x((a < x \wedge x < b) \rightarrow \varphi(x))$. Atoms are of the form $@x$ where x is some variable, of which when it is evaluated, is replaced with a numeral. Essentially, $@\overline{n}$, where $\overline{n}$ is a numeric constant, is a propositional variable which is assigned true or false depending on the truth value assigned to the position $\overline{n}$ in the external stream. Every formulae φ we will evaluate is encapsulated in a construct of the form $mon(x) : \varphi(x)$, where x is referred to as the *stream variable.* Consider the stream variable as an external quantifier over which all propositional symbols and internal quantifiers are evaluated. From now on we will only use x as the stream variable.

$$
\begin{array}{lcl}
M & \Longrightarrow & mon(X) : F(X). \\
F & \Longrightarrow & @X \mid {}^{\sim}F(X) \mid F(X)\ \&\ F(X) \mid F(X)\ \wedge\ F(X) \mid \\
 & & \forall X_{[B,B]} : F\ . \\
B & \Longrightarrow & 0 \mid \infty \mid X \mid B + N \mid B - N\ . \\
N & \Longrightarrow & k \in \mathbb{N}. \\
X & \Longrightarrow & x \in \mathbb{V}.
\end{array}
$$

Figure 1: We will refer to this grammar as FF_g *Full Formula grammar.*

The rest of this paper will focus on the space complexity of simple formulae constructable using FF_g. And, from these simple formulae we will show how one can recursively define a complexity bound for the entire core language.

Example 1 *Let us consider the following sentence constructed using FF_g:*

$$mon(x) : \forall y_{[x+2,x+4]} : (@y\&\&@x)$$

Evaluation of this monitor is performed as following by instantiating the monitor variable x with a value starting at zero:

$x = 0$	$x = 1$	$x = 2$
$\forall y_{[2,4]} : (@y\&\&@0)$	$\forall y_{[2,4]} : (@y\&\&@0)$	$\forall y_{[3,4]} : (@y\&\&@0)$
	$\forall y_{[3,5]} : (@y\&\&@1)$	$\forall y_{[3,5]} : (@y\&\&@1)$
		$\forall y_{[4,6]} : (@y\&\&@2)$
$x = 3$	$x = 4$	$x = 5$
$(@4\&\&@0)$	$(@5\&\&@1)$	$(@6\&\&@2)$
$\forall y_{[4,5]} : (@y\&\&@1)$	$\forall y_{[4,6]} : (@y\&\&@2)$	$\forall y_{[6,7]} : (@y\&\&@3)$
$\forall y_{[4,6]} : (@y\&\&@2)$	$\forall y_{[5,7]} : (@y\&\&@3)$	$\forall y_{[6,8]} : (@y\&\&@4)$
$\forall y_{[5,7]} : (@y\&\&@3)$	$\forall y_{[6,8]} : (@y\&\&@4)$	$\forall y_{[7,9]} : (@y\&\&@5)$

We assume that the truth value of every position on the stream is saved, thus, we can decide any propositional statement with positional variable index by 0 *up to the current value of x. For example when $x = 5$ we cannot remove* $(@6\&\&@2)$ *from memory because it relies on a future position. Notice that at no position did we consider the true value of any of the propositional variables. We do not care for the truth value of the positional* *sentences in this work, but rather we care to know if it is possible to evaluate them given the currently constructed model.*

2 Basics

Even though FF_g allows for spurious free variables, we will concern ourselves with closed formulae constructed from this grammar. We will write $S(FF_g)$ for the set of all sentences constructed from FF_g. Note that we consider $mon(X)$ to be a quantifier binding the outermost variable. However, we will not deal with FF_g in full formality till Sec. 9 . For the most part of this paper we concern ourselves with the simplified grammar depicted in Fig. 2.

$$\begin{array}{lcl} M & \Longrightarrow & mon(X) : F(X). \\ F & \Longrightarrow & @X \mid {\sim}F(X) \mid F(X)\ \&\ F(X) \mid F(X)\ \wedge\ F(X) \mid \\ & & \forall X_{[B,B]} : F\ . \\ B & \Longrightarrow & \infty \mid X \mid B+N \mid B-N\ . \\ N & \Longrightarrow & k \in \mathbb{N}. \\ X & \Longrightarrow & x \in \mathbb{V}. \end{array}$$

Figure 2: We will refer to this grammar as VB_g *Variable Bounds grammar.*

The only difference between the two grammars is that we dropped 0 from the line defining the quantifier bounds. Interestingly enough this gets rid of two significantly harder cases which we will discuss in future work.

In Sec. 4 we will only deal with an even simpler fragment of the language constructable using the grammar VB_g, which we will refer to as the *simplified stream fragment Ssf*. This fragment is defined as follows:

Definition 1 (*Ssf* formulae) *Let G be a formulae constructed using VB_g, and let F by a formula (without quantification) with two holes ($F[\bullet,\star]$) , one for a formula ($\bullet$) and one for a variable symbol ($\star$). Let F' (without quantification) be a formula with three holes ($F'[\bullet,\star_1,\star_2]$), one for a formula ($\bullet$) and two for variable symbols ($\star_1,\star_2$). Then $G \in Ssf$ if it has the following form:*

$$G \equiv mon(x) : F[\forall y_{[x+a,x+b]} : F'[\left(\forall z_{[x+c,x+c]} : @z\right), x, y], x]$$

where $a, b, c \in \mathbb{N}$ and $x, y, z \in \mathbb{V}$. We assume F contains only one occurrence of $\forall y_{[x+a,x+b]} : F'[\left(\forall z_{[x+c,x+c]} : @z\right), x, y]$

The fragment *Ssf* is designed to act as an invariant for a to be defined primitive recursive function. The innermost quantifier (the second $\forall$) of the *Ssf* fragment highlights the position at which the formula G can be evaluated. If we where to allow nesting of quantifiers, one could think of the second $\forall$ as a propositional variable indexed by a quantifier higher in the nesting. When concerning ourselves with the space complexity of evaluating formulae of the *Ssf* fragment, we can ignore the formulae F and F' in the definition because they are quantifier free, i.e. they are without arithmetic properties and will evaluate immediately when the correct position of the external stream is reached. Thus, we can summarize any formula of *Ssf* by the three constants used to construct the quantifier interval, namely a, b, and c. Also, if $a > b$ then we know that the number of instances we need to evaluate is zero given that the quantifier is empty. Given these abstractions of the standard formula structure, from now on, we will write members of the fragment *Ssf* as $\langle a, b, c\rangle_{Ssf}$, where $0 \leq a \leq b$ and $a, b, c \in \mathbb{N}$. We will refer to these objects as *Ssf*-triples.

3 Evaluating *Ssf*-triples

In this section we define the procedure for evaluating *Ssf*-triples in a similar manor as the evaluation steps shown in Ex. 1.

Definition 2 *Given an Ssf-triple* $\langle a, b, c\rangle_{Ssf}$*,* $\overline{n} \in \mathbb{N}$*, and two quantifier-free formulae with holes* $F[\bullet, \star]$ *and* $F'[\bullet, \star_1, \star_2]$, an instance of $\langle a, b, c\rangle_{Ssf}$ *is an object* $\langle a, b, c\rangle_{Ssf}(\overline{n})$ *which translates to the following formula instance:*

$$mon(\overline{n}) : F[\forall y_{[\overline{n}+a,\overline{n}+b]} : F'[\left(\forall z_{[\overline{n}+c,\overline{n}+c]} : @z\right), \overline{n}, y], \overline{n}]$$

We create instances of *Ssf*-triples by evaluating the *Ssf*-triples over an external stream with a start point α and an end point at β and instantiating the *Ssf*-triples with the current position of the stream. Essentially, instead of evaluating a given *Ssf*-triple t over the entire external stream, for now we will only consider evaluation of t over an interval $[\alpha, \beta]$. In this setting, consider $\triangleright$ to be a symbol corresponding to one place to the left of α, i.e. the start of the stream.

Definition 3 *An* evaluation structure *is a tuple of the form:*

$$[n, t, \mathbf{I}]$$

where $n \in \mathbb{N} \cup \{\triangleright\}$*,* t *is an Ssf-triple and* $\mathbf{I}$ *is a set of instances of* t*. when* $n = \triangleright$*, then the structure is called a* start evaluation structure *and* $\mathbf{I} \equiv \emptyset$*.*

Now that we have defined evaluation structures, let us consider how evaluation of a given structure over the fragment $[\alpha, \beta]$ of the external stream would be carried out.

Definition 4 *Given an evaluation structure (or start evaluation structure)*

$$[n, t, \mathbf{I}]$$

and an interval $[\alpha, \beta]$, where $n \in [\alpha, \beta) \cup \{\rhd\}$, the evaluator $\xrightarrow{[\alpha,\beta]}$ *is defined as*

$$[n, t, \mathbf{I}] \xrightarrow{[\alpha,\beta]} [n+1, t, \mathbf{I}']$$

where the structure to the right of the arrow is derived from the structure to the left of the arrow after incrementing the first component by one. The set $\mathbf{I}'$ is constructed using the following mechanism. First we construct the set

$$\mathbf{I}^{n+1} \equiv \mathbf{I} \cup \{t(n+1)\}$$

which is essentially the initial set from the previous evaluation structure with the addition of the instance $n+1$ of t. From $\mathbf{I}^{n+1}$ we can construct the set of all instances which can be partially unrolled*, i.e. the stream variable has reached a message corresponding to a point in the internal quantifier's interval. This set, $\mathbf{I}_u^{(n+1)}$, is constructed as follows:*

$$\mathbf{I}_u^{(n+1)} = \left\{ \quad (\mathbf{a}, i) \quad \left| \begin{array}{l} \exists a', b, c, \gamma \Big(\ \mathbf{a} \in \mathbf{I}^{n+1} \wedge \mathbf{a} = \langle a', b, c \rangle_{Ssf}(\gamma) \\ \\ \wedge \ a' \leq i \leq \ b \wedge i + \gamma = n+1 \wedge i \in \mathbb{N}) \end{array} \right. \right\}$$

Every instance number pair $(\mathbf{a}, i) \in \mathbf{I}_u^{(n+1)}$ can be split into multiple parts. The idea is that $\mathbf{I}_u^{(n+1)}$ contains all instances in the instance set which can be unrolled given position $n+1$ of the external stream.

Let us consider an object called the lower set for $(\mathbf{a}, i)$ at m , *namely $L_m[(\mathbf{a}, i)]$, that is, given an Ssf-triple $\langle a, b, c \rangle_{Ssf}(\gamma)$ paired with i such that $\gamma + i = m$, the set $L_m\left[(\langle a, b, c \rangle_{Ssf}(\gamma), i)\right]$ consists of all numerals w such that $a \leq w \leq i \leq b$. We define* the set of lower triples L_{n+1} *as follows:*

$$L_{n+1} = \bigcup_{(\mathbf{a},i) \in \mathbf{I}_u^{(n+1)}} \left\{ (\langle a', a', c \rangle_{Ssf}(n+1-i) \middle| a' \in L_{n+1}[(\mathbf{a}, i)] \right\}$$

Essentially, the set L_{n+1}, is a set of all positions in the internal quantifier's interval which can be considered given the current external stream position. We also define the set of upper triples *as follows:*

$$U_{n+1} = \bigcup_{(\langle a',b,c \rangle_{Ssf}(\gamma), i) \in \mathbf{I}_u^{(n+1)}} \left\{ \langle i+1, b, c \rangle_{Ssf}(\gamma) \middle| a' < i+1 < b \right\}$$

Essentially, U_{n+1} is the set of shortened internal quantifier intervals after the removal of all instances which are in the set L_{n+1}. Now we define the set $\mathbf{I}_0^{(n+1)}$ as follows:

$$\mathbf{I}_0^{(n+1)} = (\mathbf{I}^{n+1} - \mathbf{I}_u^{(n+1)}) \cup L_{n+1} \cup U_{n+1}$$

Essentially, $\mathbf{I}_0^{(n+1)}$ is the set of instances after taking the new external stream position into consideration, i.e. what internal quantifiers can be unrolled. The last set we need to define, prior to deriving $\mathbf{I}'$ is $\mathbf{I}_1^{(n+1)}$ which is defined as follows:

$$\mathbf{I}_1^{(n+1)} = \left\{ \mathbf{a} \;\middle|\; \exists a', c, \gamma \left(\begin{array}{c} \mathbf{a} \in \mathbf{I}_0^{(n+1)} \wedge \mathbf{a} = \langle a', a', c\rangle_{so}(\gamma) \wedge \\ a' + \gamma \leq n+1 \wedge c + \gamma \leq n+1 \end{array} \right) \right\}$$

This is the set of all Ssf-triples instances which can be evaluated given the new external stream position. These instances should be removed from the instance set. Thus, $\mathbf{I}'$ *is defined as follows:*

$$\mathbf{I}' = \mathbf{I}_0^{(n+1)} - \mathbf{I}_1^{(n+1)}$$

Using the above defined evaluator, we can define *chains* of evaluation steps. When concerning ourselves with the space complexity of evaluating *Ssf*-triples on a external stream quantifier, we need to provide bounds for the set of instances in relation to the size of the complete proper evaluation chain of a given *Ssf*-triple. We assume for now that the external stream quantifier is a finite interval $[\alpha, \beta]$. However, everyone of our results will be generalized to the interval $[\alpha, \infty)$.

Definition 5 (Evaluation Chain) *Given a Ssf-triple t, an interval* $[\alpha, \beta]$*,* $\delta \in [\alpha, \beta] \cup \{\rhd\}$ *and* $\gamma \in [\alpha, \beta]$ *such that* $\delta \leq \gamma$ *and a set of Ssf-triple instances of t, namely* $\mathbf{I}$*, an* evaluation chain *is a sequence of evaluator steps starting at* δ *with the set* $\mathbf{I}$ *and ending at* $\gamma \leq \beta$*.*

$$[\delta, t, \mathbf{I}] \xrightarrow{[\alpha,\beta]} \cdots \left[n, t, \mathbf{I}'\right] \xrightarrow{[\alpha,\beta]} \left[n+1, t, \mathbf{I}''\right] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} \left[\gamma, t, \mathbf{I}'''\right]$$

Definition 6 (Proper Evaluation Chain) *Given a Ssf-triple t, an interval* $[\alpha, \beta]$*, a* proper evaluation chain *is a sequence of evaluator steps starting at* $\rhd$ *and ending at* $\gamma \leq \beta$*.*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\beta]} \cdots [n, t, \mathbf{I}] \xrightarrow{[\alpha,\beta]} \left[n+1, t, \mathbf{I}'\right] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} \left[\gamma, t, \mathbf{I}''\right]$$

Definition 7 (Complete Proper Evaluation Chain) *Given a Ssf-triple t, an interval* $[\alpha, \beta]$*, a* complete proper evaluation chain *is a sequence of evaluator steps starting at* $\rhd$ *and ending at* β*.*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\beta]} \cdots [n, t, \mathbf{I}] \xrightarrow{[\alpha,\beta]} \left[n+1, t, \mathbf{I}'\right] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} \left[\beta, t, \mathbf{I}''\right]$$

The simplest question to start with is which *Ssf*-triples lead to a constant size instance set and how large is the constant. We answer this question by starting with the simplest *Ssf*-triple we can consider (which is non-trivial) and gradually increase the complexity of the considered triple till we cover all possible *Ssf*-triples.

4 Bounding the Space Complexity of *Ssf*-triples

In this section we apply the evaluation method define in Sec. 3 to specially constructed *Ssf*-triples in order to get space complexity bounds, at the end of the section we will provide a space complexity bound for general *Ssf*-triples.

Theorem 1 *Given an Ssf-triple t of the form $\langle 0, b, b\rangle_{ssf}$ and an interval $[\alpha, \beta]$ such that $\alpha \leq b < \beta$, then there exists a value $x \in [\alpha, \beta]$ such that given the complete proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} [x-1, t, \mathbf{I}_0] \xrightarrow{[\alpha,\beta]} [x, t, \mathbf{I}_1] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} [\beta, t, \mathbf{I}_{\beta-x}]$$

the following holds $|\mathbf{I}_0| \neq |\mathbf{I}_1| = \cdots = |\mathbf{I}_{\beta-x}|$.

Proof.

Let us assume that $x = \alpha + b - 1$ and show that the theorem holds for this value. It is obvious that the theorem cannot hold for a value smaller than or equal to $\alpha + b - 1$ because prior to this point instances are added to the instance set, but not removed, i.e. $\mathbf{I}_1^{(\gamma)} = \emptyset$ for $\gamma \in [\alpha, \alpha + b - 1]$. Thus, we are looking at the evaluation chain:

$$[\alpha + b - 1, t, \mathbf{I}_1] \xrightarrow{[\alpha,\beta]} [\alpha + b, t, \mathbf{I}_2] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} \left[\beta, t, \mathbf{I}_{\beta-(\alpha+b)}\right]$$

To calculate the difference in size between $\mathbf{I}_1$ and $\mathbf{I}_2$ we need to construct the various sets found in Def. 4 for position $\alpha + b$.

$$\mathbf{I}^{\alpha+b} \equiv \mathbf{I}_1 \cup \left\{\langle 0, b, b\rangle_{ssf}(\alpha + b)\right\}$$

$$\mathbf{I}_u^{(\alpha+b)} \equiv \bigcup_{i=0}^{b} \left\{\langle b - i, b, b\rangle_{ssf}(\alpha + i)\right\}$$

$$L_{(\alpha+b)} \equiv \left(\bigcup_{i=0}^{b} \left\{\langle b - i, b - i, b\rangle_{ssf}(\alpha + i)\right\}\right) \cup \left\{\langle b, b, b\rangle_{ssf}(\alpha + 1)\right\}$$

$$U_{(\alpha+b)} \equiv \left(\bigcup_{i=2}^{b} \left\{\langle b + 1 - i, b, b\rangle_{ssf}(\alpha + i)\right\}\right)$$

$$\mathbf{I}_0^{(\alpha+b)} = (\mathbf{I}^{(\alpha+b)} - \mathbf{I}_u^{(\alpha+b)}) \cup L_{(\alpha+b)} \cup U_{(\alpha+b)}$$

$$\mathbf{I}_1^{(\alpha+b)} = \left(\bigcup_{i=0}^{b} \left\{\langle i, i, b\rangle_{ssf}(\alpha)\right\}\right)$$

$$I_1 = \mathbf{I}_0^{(\alpha+b)} - \mathbf{I}_1^{(\alpha+b)}$$

We want to show now that I_1 and I_2 have the same size. We know the following:

$$|I_2| = |\mathbf{I}_0^{(\alpha+b)} - \mathbf{I}_1^{(\alpha+b)}| = |\mathbf{I}_0^{(\alpha+b)}| - (b + 1)$$

$$|\mathbf{I}_0^{(\alpha+b)}| = \left|\left(\mathbf{I}^{(\alpha+b)} - \mathbf{I}_u^{(\alpha+b)}\right) \cup L_{\alpha+b} \cup U_{\alpha+b}\right|$$

$$|\mathbf{I}^{(\alpha+b)}| = |\mathbf{I}_1| + 1$$

We also know that $\mathbf{I}_u^{(\alpha+b)}$ contains $b+1$ elements:

$$|\left(\mathbf{I}^{(\alpha+b)} - \mathbf{I}_u^{(\alpha+b)}\right)| = |\mathbf{I}_1| + 1 - (b+1)$$

Now getting back to $L_{\alpha+b} \cup U_{\alpha+b}$ which has a size of $(b+2)+(b-1)$ we get the following equation for the size of $\mathbf{I}_1$:

$$|\mathbf{I}_2| = ((|\mathbf{I}_1| + 1 - (b+1)) + (b+2) + (b-1)) - (b+1) = (|\mathbf{I}| + 1 - (b+1)) + b$$
$$= |\mathbf{I}_1|$$

And thus

$$|\mathbf{I}_2| = |\mathbf{I}_1|.$$

Now for the stepcase, let us assume that the theorem holds for $1 \leq i < \beta - (\alpha + b)$ and show that it also holds for $i+1$. This implies, using the complete proper evaluation chain from above, that $|\mathbf{I}_1| = |\mathbf{I}_2| = \cdots |\mathbf{I}_i|$ and we need to show that it is also the case that $|\mathbf{I}_i| = |\mathbf{I}_{i+1}|$, which can be shown by comparing the structure of the sets constructed by Def. 4. For example, the structure at position i of $\mathbf{I}_u^{(i)}$ must be

$$\mathbf{I}_u^{(i)} \equiv \bigcup_{j=0}^{b} \left\{ \langle b-j, b, b \rangle_{ssf} (i - b + j) \right\}$$

because if it where to contain an *Ssf*-triple instance for $i-(b+1)$ it would imply $\mathbf{I}_1^{(i-1)}$ was missing a *Ssf*-triple instance when constructed. To show that this would be the case, consider a triple of the form $\langle a', a', b \rangle_{ssf} (i - (b+1))$, let us consider when such a triple would by part of $\mathbf{I}_1^{(i-1)}$. This membership is based on the third component of the *Ssf*-triple plus the instance, which adds up to $i-1$. Thus, it would have to be a member of $\mathbf{I}_1^{(i-1)}$ and would have been removed earlier in the chain.

The same argument works for constructing $\mathbf{I}_u^{(i+1)}$ which has the following structure:

$$\mathbf{I}_u^{(i+1)} \equiv \bigcup_{j=0}^{b} \left\{ \langle b-j, b, b \rangle_{ssf} (i + 1 - b + j) \right\}$$

It is now obvious that there is a bijective mapping from $\mathbf{I}_u^{(i)}$ to $\mathbf{I}_u^{(i+1)}$, they have the same size $b+1$, and from both L_i and U_i to L_{i+1} and U_{i+1}, again because they are constructed in a similar fashion to $\mathbf{I}_u^{(i)}$ and $\mathbf{I}_u^{(i+1)}$, and have the same sizes. Also, because $|\mathbf{I}_i| = |\mathbf{I}_{i-1}|$ we know that $|\mathbf{I}^i| = |\mathbf{I}^{i+1}|$ and thus, $|\mathbf{I}_0^{(i)}| = |\mathbf{I}_0^{(i+1)}|$. Thus, the only thing left to show is that $|\mathbf{I}_1^{(i)}| = |\mathbf{I}_1^{(i+1)}|$. This is also obvious being that only instances $i+1-b$ can be in $\mathbf{I}_1^{(i+1)}$ and instances $i-b$ can be in $|\mathbf{I}_1^{(i)}|$, of which in both cases there are $b+1$ of them. Thus, $|\mathbf{I}_i| = |\mathbf{I}_{i+1}|$ and by induction we have shown the theorem to be true.

$\square$

Thm. 1 tells us that the class of *Ssf*-triples $\langle 0, b, b\rangle_{ssf}$ requires a bounded amount of space to evaluate on a complete proper evaluation chain. We can easily generalize this result to a proper evaluation chain with upper bound at infinity based on the induction argument used to prove Thm. 1.

Corollary 1 *Given an Ssf-triple t of the form $\langle 0, b, b\rangle_{ssf}$ and an interval $[\alpha, \infty]$, then there exists a value $x \in [\alpha, \infty]$ such that given the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} \cdots [x-1, t, \mathbf{I}_0] \xrightarrow{[\alpha,\infty]} [x, t, \mathbf{I}_1] \xrightarrow{[\alpha,\infty]} \cdots$$

the following holds $|\mathbf{I}_0| \neq |\mathbf{I}_1| = |\mathbf{I}_2| = \cdots$.

Though we have proven that the number of instances needed to evaluate an *Ssf*-triple of the form $\langle 0, b, b\rangle_{ssf}$ over the external stream is constant at some point, we do not know how big the constant is yet. This can easily be deduced by looking at the set $\mathbf{I}_u^{(\alpha+b)}$, and seeing how much has been unrolled up to that point.

Lemma 1 *Given an Ssf-triple t of the form $\langle 0, b, b\rangle_{ssf}$, an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha+1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq \frac{(b+1)*(b+2)}{2} - 1$$

for all $n \in [\alpha, \infty]$.

Proof.

Take the set $\mathbf{I}_u^{(\alpha+b)}$ and count how many elements have thus far been unrolled into the form $\langle a, a, b\rangle_{ssf}(n)$ or $\langle a', b, b\rangle_{ssf}(n)$ for $a' < b$. The result is the following summation:

$$\sum_{i=2}^{b+1} i = \frac{(b+1)*(b+2)}{2} - 1$$

$\square$

In other words, the space complexity concerning *Ssf*-triples of the form $\langle 0, b, b\rangle_{ssf}$ is $\Omega(b^2)$.

Now we have derived a baseline for the evaluation of *Ssf*-triples. Our next step is to extend the current result to more general classes of *Ssf*-triples until we cover all possible *Ssf*-triples.

Theorem 2 *Given an Ssf-triple t of the form $\langle 0, b, b+c\rangle_{ssf}$,where $\alpha + b + c \leq \beta$ and $c \in \mathbb{N}$, and an interval $[\alpha, \beta]$ such that $\alpha \leq b + c < \beta$, then there exists a value $x \in [\alpha, \beta]$ such that given the complete proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\beta]} \cdots [x-1, t, \mathbf{I}_0] \xrightarrow{[\alpha,\beta]} [x, t, \mathbf{I}_1] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} [\beta, t, \mathbf{I}_{\beta - x}]$$

the following holds $|\mathbf{I}_0| \neq |\mathbf{I}_1| = \cdots = |\mathbf{I}_{\beta-x}|$.

Proof.

For the basecase, we consider the case when $c = 0$ which is equivalent to Thm. 1 and thus holds. Let us assume the theorem holds for all $d < c + 1$ and show that the theorem holds for $c + 1$. We know, given the *Ssf*-triple $\langle 0, b, b+c\rangle_{ssf}$, using the same argumentation as Thm. 1 that there are $b + 1$ *Ssf*-triple instances with numeral α at position $\alpha + b + c$ in the evaluation chain. These $b + 1$ *Ssf*-triple instances will make up the set $\mathbf{I}_1^{(\alpha+b+c)}$ for the *Ssf*-triple $\langle 0, b, b+c\rangle_{ssf}$. And by the induction hypothesis, if we look at the evaluation chain

$$\left[\alpha + b + c, \langle 0, b, b+c\rangle_{ssf}, \mathbf{I}\right] \xrightarrow{[\alpha,\beta]} \left[\alpha + b + c + 1, \langle 0, b, b+c\rangle_{ssf}, \mathbf{I}_1\right]$$

we know that $|\mathbf{I}| = |\mathbf{I}_1|$. However, we instead looked at the evaluation chain

$$\left[\alpha + b + c, \langle 0, b, b+c+1\rangle_{ssf}, \mathbf{I}\right] \xrightarrow{[\alpha,\beta]} \left[\alpha + b + c + 1, \langle 0, b, b+c+1\rangle_{ssf}, \mathbf{I}_1\right]$$

the set $\mathbf{I}_1^{(\alpha+b+c)} \equiv \emptyset$ because the $b + 1$ *Ssf*-triple instances with numeral α now have a third component with $\alpha + b + c + 1$. Thus, for the *Ssf*-triple $\langle 0, b, b+c+1\rangle_{ssf}$, we instead need to look at the evaluation chain

$$\left[\alpha + b + c + 1, \langle 0, b, b+c+1\rangle_{ssf}, \mathbf{I}\right] \xrightarrow{[\alpha,\beta]} \left[\alpha + b + c + 2, \langle 0, b, b+c+1\rangle_{ssf}, \mathbf{I}_1\right]$$

being that in this case, $\mathbf{I}_1^{(\alpha+b+c+1)}$ is non-empty for the first time. By similar set analysis of the various parts of Def. 4 as in Thm 1, we get the same constant size result and thus the theorem holds.

$\square$

Corollary 2 *Given an Ssf-triple t of the form $\langle 0, b, b+c\rangle_{ssf}$, where $c \in \mathbb{N}$, and an interval $[\alpha, \infty]$, then there exists a value $x \in [\alpha, \infty]$ such that given the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} \cdots [x-1, t, \mathbf{I}_0] \xrightarrow{[\alpha,\infty]} [x, t, \mathbf{I}_1] \xrightarrow{[\alpha,\infty]} \cdots$$

the following holds $|\mathbf{I}_0| \neq |\mathbf{I}_1| = |\mathbf{I}_2| = \cdots$.

Theorem 3 *Given an Ssf-triple t of the form $\langle 0, b, b+c\rangle_{ssf}$, an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha+1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq \frac{(b+1)*(b+2)}{2} - 1 + c*(b+1)$$

for all $n \in [\alpha, \infty]$.

Proof.

We know by Lem. 1 that when $c = 0$, the bound is the following function

$$\frac{(b+1)*(b+2)}{2} - 1.$$

By carefully evaluating the stepcase of Thm. 2, we see that when we go from c to $c+1$ the number of elements which would have been removed from the instance set for c, but are no longer removed from the instance set for $c+1$ is $b+1$. Thus, we the following equation.

$$\frac{(b+1)*(b+2)}{2} - 1 + c*(b+1).$$

$\square$

In other words, the space complexity concerning *Ssf*-triples of the form $\langle 0, b, b+c\rangle_{ssf}$ is $\Theta(b^2 + bc)$.

The next two generalizations which need to be made are the cases when $-b \leq c < 0$ in the *Ssf*-triple $\langle 0, b, b+c\rangle_{ssf}$ and when $0 < a \leq b$.

Theorem 4 *Given an Ssf-triple t of the form $\langle 0, b, b-c\rangle_{ssf}$,where $0 < c \leq b$, and an interval $[\alpha, \beta]$ such that $\alpha \leq b < \beta$, then there exists a value $x \in [\alpha, \beta]$ such that given the complete proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\beta]} \cdots [x-1, t, \mathbf{I}_0] \xrightarrow{[\alpha,\beta]} [x, t, \mathbf{I}_1] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} [\beta, t, \mathbf{I}_{\beta-x}]$$

the following holds $|\mathbf{I}_0| \neq |\mathbf{I}_1| = \cdots = |\mathbf{I}_{\beta-x}|$.

Proof.

Unlike the previous cases, the first time $\mathbf{I}_1^{(n)} \equiv \emptyset$, where $n \in \mathbb{N}$ is less than the upper bound b, more specifically it is at $\alpha + b - c$. The set $\mathbf{I}_u^{(\alpha+b-c)}$ is different then in the previous cases because the *Ssf*-triple instance for α is not fully unrolled at this point.

$$\mathbf{I}_u^{(\alpha+b-c)} \equiv \bigcup_{j=0}^{b-c} \left\{ \langle b-c-j, b, b-c\rangle_{ssf}(\alpha+j) \right\}$$

We can compute the other sets of Def. 4 at this point assuming the evaluation chain (the $'$ are used because these sets are not the ones corresponding to the theorem statement),

$$\left[\alpha+b-c-1,t,\mathbf{I}'_0\right] \xrightarrow{[\alpha,\beta]} \left[\alpha+b-c,t,\mathbf{I}'_1\right]$$

as follows:

$$\mathbf{I}^{\alpha+b-c} \equiv \mathbf{I}'_0 \cup \left\{\langle 0,b,b-c\rangle_{ssf}(\alpha+b-c)\right\}$$

$$L_{(\alpha+b-c)} \equiv \left(\bigcup_{i=0}^{b-c}\left\{\langle b-c-i,b-c-i,b-c\rangle_{ssf}(\alpha+i)\right\}\right)$$

$$U_{(\alpha+b-c)} \equiv \left(\bigcup_{i=0}^{b-c}\left\{\langle b-c+1-i,b,b-c\rangle_{ssf}(\alpha+i)\right\}\right)$$

$$\mathbf{I}_0^{(\alpha+b-c)} = (\mathbf{I}^{(\alpha+b-c)} - \mathbf{I}_u^{(\alpha+b-c)}) \cup L_{(\alpha+b-c)} \cup U_{(\alpha+b-c)}$$

$$\mathbf{I}_1^{(\alpha+b-c)} = \left(\bigcup_{i=0}^{b-c}\left\{\langle i,i,b-c\rangle_{ssf}(\alpha)\right\}\right)$$

$$I'_1 = \mathbf{I}_0^{(\alpha+b-c)} - \mathbf{I}_1^{(\alpha+b-c)}$$

By doing the same arithmetic as in Thm. 1 we do not get that I'_0 and I'_1 have the same size, but rather that $|I'_1| = |I'_0|+1$. We know the following:

$$|I'_1| = |\mathbf{I}_0^{(\alpha+b-c)} - \mathbf{I}_1^{(\alpha+b-c)}| = |\mathbf{I}_0^{(\alpha+b-c)}| - (b-c+1)$$

$$|\mathbf{I}_0^{(\alpha+b-c)}| = \left|\left(\mathbf{I}^{(\alpha+b-c)} - \mathbf{I}_u^{(\alpha+b-c)}\right) \cup L_{\alpha+b-c} \cup U_{\alpha+b-c}\right|$$

$$|\mathbf{I}^{(\alpha+b-c)}| = |\mathbf{I}'_0|+1$$

$$|\left(\mathbf{I}^{(\alpha+b-c)} - \mathbf{I}_u^{(\alpha+b-c)}\right)| = |\mathbf{I}_0|+1-(b-c+1)$$

Now getting back to $L_{\alpha+b} \cup U_{\alpha+b}$ which has a size of $2*(b-c+1)$ we get the following equation for the size of $\mathbf{I}'_1$:

$$|\mathbf{I}'_1| = ((|\mathbf{I}_0|+1-(b-c+1)) + 2*(b-c+1)) - (b-c+1)) = |\mathbf{I}'_0|+1$$

Now that we have shown that $|\mathbf{I}'_1| = |\mathbf{I}'_0|+1$ we can consider at what point does it stabilize, i.e. are there instance sets such that $|\mathbf{I}'_{i+1}| = |\mathbf{I}'_i|$. Let us now consider the following evaluation chain:

$$\left[\alpha+b-c-1,t,\mathbf{I}'_0\right] \xrightarrow{[\alpha,\beta]} \left[\alpha+b-c,t,\mathbf{I}'_1\right] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]}$$

$$\left[\alpha + b - 1, t, \mathbf{I}'_{c-1}\right] \xrightarrow{[\alpha,\beta]} \left[\alpha + b, t, \mathbf{I}'_c\right]$$

If we consider the set $\mathbf{I}_u^{(\alpha+b)}$ we see a similar pattern as the one found in Thm. 1:

$$\mathbf{I}_u^{(\alpha+b)} \equiv \bigcup_{j=0}^{b} \left\{ \langle b-j, b, b-c\rangle_{ssf} (\alpha + j) \right\}$$

However, this time it is not the third component which is holding up the computation, rather it is the fact that we have not reached the appropriate message on the stream to even consider the *Ssf*-triple.

$$L_{(\alpha+b)} \equiv \left(\bigcup_{i=0}^{b} \left\{ \langle b-i, b-i, b-c\rangle_{ssf} (\alpha + i) \right\} \right)$$

$$U_{(\alpha+b)} \equiv \left(\bigcup_{i=1}^{b} \left\{ \langle b+1-i, b, b-c\rangle_{ssf} (\alpha + i) \right\} \right)$$

$$\mathbf{I}_0^{(\alpha+b)} = (\mathbf{I}^{(\alpha+b)} - \mathbf{I}_u^{(\alpha+b)}) \cup L_{(\alpha+b)} \cup U_{(\alpha+b)}$$

$$\mathbf{I}_1^{(\alpha+b)} = \left(\bigcup_{i=0}^{b} \left\{ \langle b-i, b-i, b-c\rangle_{ssf} (\alpha + i) \right\} \right)$$

$$I'_c = \mathbf{I}_0^{(\alpha+b)} - \mathbf{I}_1^{(\alpha+b)}$$

If one is to go through all the calculations, one would get

$$|\mathbf{I}'_c| = ((|\mathbf{I}'_{c-1}| + 1 - (b+1)) + (b+1) + b) - (b+1)) = (|\mathbf{I}'_{c-1}| + 1 - (b+1)) + b$$
$$= |\mathbf{I}'_{c-1}|$$

and thus, $|\mathbf{I}'_c| = |\mathbf{I}'_{c-1}|$. We now know that $\mathbf{I}'_c = \mathbf{I}_2$ and $\mathbf{I}'_{c-1} = \mathbf{I}_1$ By a similar inductive argument as Thm 1 & 2, one can show that it stays constant. However, we do not know anything about the positions in the stream between $\alpha + b - c$ and $\alpha + b - 1$, i.e. are they constant as well or are they increasing. We have to do an additional induction on the evaluation chain:

$$\left[\alpha + b - c - 1, t, \mathbf{I}'_0\right] \xrightarrow{[\alpha,\beta]} \left[\alpha + b - c, t, \mathbf{I}'_1\right] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} [\alpha + b - 1, t, \mathbf{I}_1]$$

To show that over this interval the instance set sizes are strictly monotonically increasing. As a basecase, we already know that $|\mathbf{I}'_1| = |\mathbf{I}'_0| + 1$, let us assume as induction hypothesis that for $1 \le w < c - 1$, the instance set size is increasing, we show that it is also increasing for $w < w + 1 \le c - 1$. Again we compute the sets of Def. 4:

$$\mathbf{I}_u^{(\alpha+b-c+(w+1))} \equiv \bigcup_{j=0}^{b-c+(w+1)} \left\{ \langle b-c+(w+1)-j, b, b-c\rangle_{ssf} (\alpha + j) \right\}$$

$$\mathbf{I}^{\alpha+b-c+(w+1)} \equiv \mathbf{I}_w \cup \left\{ \langle 0, b, b-c\rangle_{ssf} \left(\alpha + b - c + (w+1)\right) \right\}$$

$$L_{(\alpha+b-c+(w+1))} \equiv$$

$$\left(\bigcup_{i=0}^{b-c+(w+1)} \left\{ \langle b-c+(w+1)-i, b-c+(w+1)-i, b-c\rangle_{ssf} (\alpha+i) \right\} \right)$$

$$U_{(\alpha+b-c+(w+1))} \equiv$$

$$\left(\bigcup_{i=0}^{b-c+(w+1)} \left\{ \langle b-c+(w+1)+1-i, b, b-c\rangle_{ssf} (\alpha+i) \right\} \right)$$

$$\mathbf{I}_0^{(\alpha+b-c+(w+1))} = (\mathbf{I}^{(\alpha+b-c+(w+1))} - \mathbf{I}_u^{(\alpha+b-c+(w+1))}) \cup$$

$$L_{(\alpha+b-c+(w+1))} \cup U_{(\alpha+b-c+(w+1))}$$

$$\mathbf{I}_1^{(\alpha+b-c+(w+1))} = \left(\bigcup_{i=0}^{b-c+(w+1)} \left\{ \langle i, i, b-c\rangle_{ssf} (\alpha) \right\} \right)$$

$$I_{w+1} = \mathbf{I}_0^{(\alpha+b-c+(w+1))} - \mathbf{I}_1^{(\alpha+b-c+(w+1))}$$

We know the following:

$$|I_{w+1}| = |\mathbf{I}_0^{(\alpha+b-c+(w+1))} - \mathbf{I}_1^{(\alpha+b-c+(w+1))}| = |\mathbf{I}_0^{(\alpha+b-c+(w+1))}| - (b-c+(w+1)+1)$$

$$|\mathbf{I}_0^{(\alpha+b-c+(w+1))}| = \left| \mathbf{I}^{(\alpha+b-c+(w+1))} \cup L_{\alpha+b-c+(w+1)} \cup U_{\alpha+b-c+(w+1)} \right|$$

$$|\mathbf{I}^{(\alpha+b-c+(w+1))}| = |\mathbf{I}_w| + 1$$

$$|(\mathbf{I}^{(\alpha+b-c+(w+1))} - \mathbf{I}_u^{(\alpha+b-c+(w+1))})| = |\mathbf{I}_w| + 1 - (b-c+(w+1)+1)$$

Now getting back to $L_{\alpha+b} \cup U_{\alpha+b}$ which has a size of $2 * (b-c+(w+1)+1)$ we get the following equation for the size of $\mathbf{I}_2$:

$$|\mathbf{I}_{w+1}| = ((|\mathbf{I}_w| + 1 - (b-c+(w+1)+1)) + 2*(b-c+(w+1)+1)) - (b-c+(w+1)+1)) =$$

$$|\mathbf{I}_w| + 1$$

Now that we have shown that $|\mathbf{I}_{w+1}| = |\mathbf{I}_w| + 1$ and by induction we have shown that the interval is strictly monotonically increasing.

□

Corollary 3 *Given an Ssf-triple t of the form $\langle 0, b, b-c\rangle_{ssf}$,where $0 < c \leq b$, an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} \cdots [x-1, t, \mathbf{I}_0] \xrightarrow{[\alpha,\infty]} [x, t, \mathbf{I}_1] \xrightarrow{[\alpha,\infty]} \cdots$$

the following holds $|\mathbf{I}_0| \neq |\mathbf{I}_1| = |\mathbf{I}_2| = \cdots$.

Theorem 5 *Given an Ssf-triple t of the form $\langle 0, b, b-c\rangle_{ssf}$,where $0 < c \leq b$, and an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha+1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq \frac{(b-c)*(b-c+1)}{2} + (c-1)$$

for all $n \in [\alpha, \infty]$.

Proof.

We know by Thm. 4, that the size of the instance set at position $\alpha + b$ is $|\mathbf{I}'_0| + c$ where $\mathbf{I}'_0$ is the instance set at $\alpha + b - c - 1$. Thus, all we need to compute is the size of the instance set at $\alpha + b - c - 1$. This is computed the same way as Lem. 1 except the input values are smaller. Thus we get the following:

$$\frac{(b-c)*(b-c+1)}{2} + (c-1).$$

□

In other words, the space complexity concerning *Ssf*-triples of the form $\langle 0, b, b-c\rangle_{ssf}$ is $\Theta(b^2 + c^2 - 2\cdot bc + c)$.

The last part of the *Ssf*-triple that we haven't yet discussed in terms of the effect it has on the size of the inference set is the first component, mainly concerning the simplicity of dealing with it. When the other two components are left fixed, increasing the first component does two things, firstly it shrinks the internal quantifiers interval. Secondly, it pushes the unrolling of an *Ssf*-triple instance to some future message. To be more specific, imagine an *Ssf*-triple $\langle a, b, c\rangle_{ssf}$, there is some set $\mathbf{I}$ of instances which are unrolled when $\langle a, b, c\rangle_{ssf}$ instantiate with γ. If we where to increase the value of a to $a + 1$ the set $\mathbf{I}$ will not exist until $\gamma + 1$. We now turn these thoughts into the following theorem.

Theorem 6 *Given an Ssf-triple t of the form $\langle a, b, c\rangle_{ssf}$, and an interval $[\alpha, \beta]$ such that $\alpha \leq a \leq b < \beta$ and $c \in [\alpha, \beta)$, then there exists a value $x \in [\alpha, \beta]$ such that given the complete proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} [x-1, t, \mathbf{I}_0] \xrightarrow{[\alpha,\beta]} [x, t, \mathbf{I}_1] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} [\beta, t, \mathbf{I}_{\beta-x}]$$

the following holds $|\mathbf{I}_0| \neq |\mathbf{I}_1| = \cdots = |\mathbf{I}_{\beta-x}|$.

Proof.

To prove this theorem we can ignore the second and third component and try to reduced the case of $\langle a,b,c\rangle_{ssf}$ to the case of $\langle 0,b-a,c\rangle_{ssf}$. Firstly, if $0 < a$, to unroll an instance of the *Ssf*-triple $\langle a,b,c\rangle_{ssf}(\gamma)$, we need to reduce the message on the external stream $\gamma + a$. This means every instance added to the instance set generated during evaluation of the interval $[\gamma, \gamma + a)$ cannot be unrolled. This implies that the instance set at $\gamma + (a-1)$ will have an additional instances which could not be unrolled. It should be obvious, without further argument, that at $\gamma + (a)$ evaluation of these a instances continues as it would in the case when $a = 0$ and the start of the evaluation was $\gamma + (a)$. Also, to note that the larger the value of a in relation to b, the smaller the interval of the internal quantifier. Now let us consider the following two evaluation chains where $t = \langle a,b,c\rangle_{ssf}$ and $t' = \langle 0,b-a,c\rangle_{ssf}$:

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} [\gamma + (a-1), t, \mathbf{I}_0] \xrightarrow{[\alpha,\beta]} [\gamma + a, t, \mathbf{I}_1] \xrightarrow{[\alpha,\beta]}$$
$$\cdots \xrightarrow{[\alpha,\beta]} [\min\{\gamma + b, \gamma + c\} - 1, t, \mathbf{I}_w] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} [\beta, t, \mathbf{I}_{\beta - x}]$$

$$\left[\rhd, t', \emptyset\right] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} \left[\min\{b-a, c\} - 1, t', \mathbf{I}'_0\right] \xrightarrow{[\alpha,\beta]}$$
$$\left[\min\{b-a, c\} + 1, t', \mathbf{I}'_1\right] \xrightarrow{[\alpha,\beta]} \cdots \xrightarrow{[\alpha,\beta]} \left[\beta, t', \mathbf{I}'_{\beta - x}\right]$$

An equivalence can be constructed between the sets given the analysis above, namely that $|\mathbf{I}_w| = a + |\mathbf{I}'_0|$, the position right before any terms are removed from the instance set. We know that an evaluation of t' starting with the instance set $\mathbf{I}'_0$ is eventually constant, thus, we know that $\mathbf{I}_w$ is eventually constant as well.

□

Theorem 7 *Given an Ssf-triple t of the form $\langle a,b,c\rangle_{ssf}$,where $0 \le a \le b$, $0 \le c$, an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha + 1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \le f(a,b,c)$$

for all $n \in [\alpha, \infty]$,where

$$f(a,b,c) = \left\{ \begin{array}{l|c} a + (I-1) & c = b - d \ \&\ I - d \le 0 \\ a + \dfrac{(I-d)(I-d+1)}{2} + (d-1) & c = b - d \ \&\ 0 \le I - d \\ a + \dfrac{I*(I+1)}{2} + d*I - 1 & c = b + d \end{array} \right.$$

and $I = (b-a) + 1$.

In other words, the space complexity concerning *Ssf*-triples of the form $\langle a, b, c\rangle_{ssf}$ where $a \leq b$ and $0 \leq c$ is $\Theta(f(a, b, c))$.

Thm.7 gives us a constant space bound for sentences of the simplified stream fragment. This fragment is still missing some of the key features found in the arithmetic of the core language, namely, it is missing subtraction and ∞. The next step will be to add negative values to the first, second, and third component of the triples. This fragment will be referred to as the *Negative Stream Fragment* (*Nsf*). Adding negative values to the third component is quite simple. The real issue is how to handle negative values in the first and second component being that the interval we are evaluating the quantifier over will be split into a negative and positive part. For example, when $a < 0$, the internal quantifier steam gets split into two pieces where one or both of the values a and/or b are negative, i.e. essentially, $[\gamma - a, \gamma]$ and $[\gamma, \gamma + b]$.

5 Space Complexity of the Negative Stream Fragment

In the previous section we worked with the simplified stream fragment which only allows for positive values in the quantifier interval. Thus, a bound such as $x - 4$ where x is the external stream variable would not be allowed. In this section we will consider the effects on space complexity of allowing the internal quantifier bounds to have negative constants.

We can define *Nsf-triples* in a similar way as *Ssf*-triples, but allowing negative numbers in the three components, i.e. $\langle a, b, c\rangle_{Nsf}$ where $a \leq b$ and $a, b, c \in \mathbb{Z}$. Instances of *Nsf*-triples are defined in a similar way as instances of *Ssf*-triples.

Definition 8 (*Nsf* formulae) *Let G be a sentence constructed using VB_g, and let F by a formula (without quantification) with two holes ($F[\bullet, \star]$) , one for a formula ($\bullet$) and one for a variable symbol ($\star$), and let F' (without quantification) be a formula with three holes ($F'[\bullet, \star_1, \star_2]$), one for a formula ($\bullet$) and two for variable symbols ($\star_1, \star_2$), then $G \in S(Nsf)$ if it has the following form:*

$$mon(\overline{n}) : F[\forall y_{[\overline{n} \pm a, \overline{n} \pm b]} : F'[(\forall z_{[\overline{n} \pm c, \overline{n} \pm c]} : @z) , \overline{n}, y], \overline{n}]$$

where the bounds of the quantifier $(\forall z_{[\overline{n} \pm c, \overline{n} \pm c]} : @z)$ are either $[\overline{n} + c, \overline{n} + c]$ or $[\overline{n} - c, \overline{n} - c]$

Concerning the evaluation structure, everywhere where $\mathbb{N}$ was used, we replace it with $\mathbb{Z}$ and everywhere where *Ssf*-triples was used, we replace it with *Nsf*-triples. Otherwise, the definition will be exactly the same.

The simplest consideration to make is the case when c is negative and a, and b are positive. The reason this is the simplest consideration is that the formula we constructed in Thm. 7 is the same. This can be seen by looking at the first component. If c is negative, the first component is the one used, however, it does not rely on c. Thus, we can update Thm. 7 to the following:

Theorem 8 *Given an Nsf-triple t of the form $\langle a, b, c\rangle_{Nsf}$,where $0 \leq a \leq b$, $c \in \mathbb{Z}$, an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha+1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq f(a, b, c)$$

for all $n \in [\alpha, \infty]$, and

$$\lim_{n\to\infty} |\mathbf{I}_n| = f(a, b, c)$$

where,

$$f(a,b,c) = \begin{cases} a + (I-1) & c = b - d \ \& \ I - d \leq 0 \\ a + \dfrac{(I-d)(I-d+1)}{2} + (d-1) & c = b - d \ \& \ 0 \leq I - d \\ a + \dfrac{I*(I+1)}{2} + d * I - 1 & c = b + d \end{cases}$$

and $I = (b - a) + 1$.

From now on when we refer to the function $f(a, b, c)$ we are referring to the function of Thm. 8. The most interesting behaviour resulting from the addition of negative numbers is the result of allowing $a \leq 0$ and $0 \leq b$. The reason being that this splits the internal quantifier interval into two parts $[a, 0)$ and $[0, b]$. Depending on the position of c relative to a, we can unroll and evaluate many instances faster than for positive a. This splitting ability leads to the following result.

Theorem 9 *Given an Nsf-triple t of the form $\langle a, b, c\rangle_{Nsf}$,where $0 \leq b$, and $c, a \leq 0$, an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha+1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq f(0, b, 0)$$

for all $n \in [\alpha, \infty]$, and

$$\lim_{n\to\infty} |\mathbf{I}_n| = f(0, b, 0).$$

Proof.

When $c = 0$, no matter the position on the external stream, the *Nsf*-triple $\langle 0, b, 0\rangle_{Nsf}$ can be evaluated. The same holds for triples of the form $\langle a, b, 0\rangle_{Nsf}$ where $a \leq 0$ because any triple of the form $\langle d, d, 0\rangle_{Nsf}$, where $a \leq d \leq 0$, can be evaluated at any position on the external stream. When $c < 0$, the same obviously holds.

□

When we allow $0 \leq c$ we need to also consider the build up of the negative interval.

Theorem 10 *Given an Nsf-triple t of the form $\langle a, b, c\rangle_{Nsf}$,where $0 \leq b, c$, and $a < 0$, an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha+1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq f(0, b, c) + (|a| * c)$$

for all $n \in [\alpha, \infty]$, and

$$\lim_{n\to\infty} |\mathbf{I}_n| = f(0, b, c) + (|a| * c).$$

Proof.

Simply we add the size of the interval $[a, 0)$ a number of times equivalent to how long we have to wait to evaluate the messages.

□

Next we can consider what happens when everything is negative:

Theorem 11 *Given an Nsf-triple t of the form $\langle a, b, c\rangle_{Nsf}$,where $a, b, c < 0$, an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha+1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq 0$$

for all $n \in [\alpha, \infty]$, and

$$\lim_{n\to\infty} |\mathbf{I}_n| = 0.$$

Proof.

Everything can be evaluated right away and nothing is kept in memory.

□

Theorem 12 *Given an Nsf-triple t of the form $\langle a, b, c\rangle_{Nsf}$,where $b < 0$, $c > 0$ and $a < 0$, an interval $[\alpha, \infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha+1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq c * ((b - a) + 1)$$

for all $n \in [\alpha, \infty]$, and

$$\lim_{n\to\infty} |\mathbf{I}_n| = c * ((b - a) + 1).$$

Proof.

Simply we add the size of the interval $[a, b]$ a number of times equivalent to how long we have to wait to evaluate the messages

$\square$

We can now put all the parts together and get a space complexity for any *Nsf*-triple.

Theorem 13 *Given an Nsf-triple t of the form* $\langle a, b, c\rangle_{Nsf}$ *,where* $a, b, c \in \mathbb{Z}$*, an interval* $[\alpha, \infty]$*, and the proper evaluation chain:*

$$[\triangleright, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha + 1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq g(a, b, c)$$

for all $n \in [\alpha, \infty]$*, and*

$$\lim_{n\to\infty} |\mathbf{I}_n| = g(a, b, c)$$

.

Where we define $g(a, b, c)$ *as*

$$g(a,b,c) = \begin{cases} f(a,b,c) & 0 \leq a, b \;\&\; c \in \mathbb{Z} \\ f(0,b,0) & a, c < 0 \;\&\; 0 \leq b \\ f(0,b,c) + |a| * c & a < 0 \;\&\; 0 \leq b, c \\ c * I & a, b < 0 \;\&\; c > 0 \\ 0 & \mathit{otherwise} \end{cases}$$

and $I = (b - a) + 1$*.*

6 Space Complexity of the Infinite Stream Fragment

In the case of allowing infinity in the bounds, it really only makes sense if the internal quantifier's upper bound is infinity. When the lower bound is infinity the quantifier can never evaluate, and when the single point quantifier has two infinite bounds it does not really make sense. Though, even when the upper bound on the external quantifier is the only location where infinity shows up, there is no finite bound on the size of the instance set being that every evaluation step adds one more instance to the instance set and at least one instance from the previous instance set will be present in the new set. Thus, Thm. 13 also holds for a grammar including infinite, of which we will refer to as *Infinite Stream Fragment* (*Isf*) and its triples as *Isf*-triples.

Definition 9 (*Isf* **formulae**) *Let G be a sentence constructed using VB_g, and let F by a formula (without quantification) with two holes ($F[\bullet,\star]$) , one for a formula ($\bullet$) and one for a variable symbol ($\star$), and let F' (without quantification) be a formula with three holes ($F'[\bullet,\star_1,\star_2]$), one for a formula ($\bullet$) and two for variable symbols ($\star_1,\star_2$), then $G \in S(Isf)$ if it has the following form:*

$$mon(\overline{n}) : F[\forall y_{[\overline{n}\pm a,\overline{n}\pm b]} : F'[\left(\forall z_{[\overline{n}\pm c,\overline{n}\pm c]} : @z\right), \overline{n}, y], \overline{n}]$$

where the bounds of the quantifier $\left(\forall z_{[\overline{n}\pm c,\overline{n}\pm c]} : @z\right)$ are either $[\overline{n}+c,\overline{n}+c]$ or $[\overline{n}-c,\overline{n}-c]$, and a,b, or c can be ∞. We consider $\overline{n}\pm\infty \equiv \infty$.

Theorem 14 *Given an Isf-triple t of the form $\langle a,b,c\rangle_{Isf}$,where $a,b,c \in \mathbb{Z}\cup\{\infty\}$, an interval $[\alpha,\infty]$, and the proper evaluation chain:*

$$[\rhd, t, \emptyset] \xrightarrow{[\alpha,\infty]} [\alpha, t, \mathbf{I}_\alpha] \xrightarrow{[\alpha,\infty]} [\alpha+1, t, \mathbf{I}_{\alpha+1}] \xrightarrow{[\alpha,\infty]} \cdots$$

then,

$$|\mathbf{I}_n| \leq g(a,b,c)$$

for all $n \in [\alpha,\infty]$, and

$$\lim_{n\to\infty} |\mathbf{I}_n| = g(a,b,c),$$

where we define $g(a,b,c)$ as follows:

$$g(a,b,c) = \left\{ \begin{array}{l|c} f(a,b,c) & 0 \leq a,b < \infty \ \&\ c \in \mathbb{Z} \\ f(0,b,0) & a,c < 0 \ \&\ 0 \leq b < \infty \\ f(0,b,c) + |a| * c & a < 0 \ \&\ 0 \leq b,c < \infty \\ c * I & a,b < 0 \ \&\ 0 < c < \infty \\ \infty & a,c \in \mathbb{Z} \ \&b = \infty \\ \infty & c = \infty \\ 0 & otherwise \end{array} \right.$$

and $I = (b-a)+1$.

Now that we have derived a function for the infinite stream fragment, we can state a very important property which we will use to bound the space complexity of the core language. Essentially, the function g is monotonically increasing given an increase in the size of the interval of the internal quantifier.

Theorem 15 *Given the function g of Thm. 14 and $a, b, c \in \mathbb{Z} \cup \{\infty\}$, when $g(a,b,c) \neq \infty$ then the following properties hold:*

$$1: \; g(a,b,c) \leq g(a,b+1,c)$$

$$2: \; g(a+1,b,c) \leq g(a,b,c)$$

$$3: g(a,b,c) \leq g(a,b,c+1)$$

Proof.

We will show that property 1 holds; the proofs of the other properties proceed in a similar way. We assume that neither b nor c are equivalent to infinity being that these cases are trivial. When $a < 0$ we have that $g(a,b,c) = f(0,b,0)$ or $g(a,b,c) = f(0,b,c)$. In the first case we have to show that $f(0,b,0) \leq f(0,b+1,0)$ we can rewrite this inequality using the definition of f as follows:

$$\frac{(b+1-b)(b+1-b+1)}{2}{}_{+(b-1)\leq}\frac{(b+2-(b+1))(b+2-(b+1)+1)}{2}{}_{+(b+1-1)}$$

$$b \leq b+1$$

This computation also considers the case when $c < 0$, we just remove

$$\frac{(b+2-(b+1))(b+2-(b+1)+1)}{2}$$

from the calculation. To show that $f(0,b,c) \leq f(0,b+1,c)$ there are two components of f which can be used:

$$a + \frac{(I-d)(I-d+1)}{2} + (d-1)$$

$$a + \frac{I*(I+1)}{2} + d*I - 1$$

where d is equivalent to $b-c$. When $c \leq b$ we have the following computation:

$$\frac{(b+1-(b-c))(b+1-(b-c)+1)}{2} + ((b-c)-1) \leq$$

$$\frac{(b+2-(b+1-c))(b+2-(b+1-c)+1)}{2} + ((b+1-c)-1)$$

$$\frac{(c+1)(c+2)}{2} + ((b-c)-1) \leq \frac{(c+1)(c+2)}{2} + (b-c)$$

This inequality obvious holds. When $c > b$ ($d = c - b$) we use the other component of f:

$$\frac{(b+1)*(b+2)}{2} + (c-b)*(b+1) - 1 \leq \frac{(b+2)*(b+3)}{2} + (c-b-1)*(b+2) - 1$$

$$\frac{(b+1)*(b+2)}{2} \leq \frac{(b+2)*(b+3)}{2} + (b+2) + (c-b)$$

$$0 \leq \frac{(b+2)}{2} + \frac{(b+3)}{2} + (b+2) + (c-b)$$

Thus, when $a < 0$ the statement $g(a,b,c) \leq g(a,b+1,c)$ *OR* $g(a,b,c) = \infty$ holds. When $a >= 0$ the same argument works because increasing b increases the size of the interval which is in the quadratic term. And thus the function increases.

□

7 Moving from ISF to VB_g

So far we have focused on formulae with a simplified structure, that is a single quantifier with an internal quantifier whose interval is singleton. This simplified structure allowed us to get a precise bound on the space complexity. However, the expressive power of this language is quite limited. Though, the structure we have dealt with so far can act as an invariant in recursive function bounding formulae with more complex structure. At any position in a formula F constructed using VB_g where a quantifier is used, there will be a position in its matrix which requires the position furthest in the future (i.e. the third component c) and it is possible to rewrite the formula F such that quantifier bounds only contain the external stream variable. Thus, what we need to do is transform the formula F into the correct form and then apply $g(a,b,c)$ to each quantifier inductively. This is what we do in this section.

Definition 10 *Given a sentence F constructed using VB_g, we say F is* regular *if the each quantifier has a distinct variable name for its bound variable.*

Definition 11 (Infinitary substitution) *A substitution σ^{∞} is said to be* infinitary *if given a term of the form $t[x]$ such that σ^{∞} maps x to ∞, then the following holds $t[x]\sigma^{\infty} = \infty$.*

Definition 12 (Infinitary addition) *We define the operator $+^{\infty}$ as standard addition over the natural numbers except when a term includes infinity ∞. In these cases the following holds:*

$$x +^{\infty} \infty = \infty +^{\infty} x = \infty +^{\infty} \infty = \infty$$

Definition 13 (Infinitary subtraction) *We define the operator $-^{\infty}$ as standard addition over the natural numbers except when a term includes infinity ∞. In these cases the following holds:*

$$x -^{\infty} \infty = \infty -^{\infty} x = \infty -^{\infty} \infty = \infty$$

Definition 14 (Infinitary Multiplication) *We define the operator $*^{\infty}$ as standard multiplication over the natural numbers except when a term includes ∞. In these cases the following holds:*

$$x *^{\infty} \infty = \infty *^{\infty} x = \infty *^{\infty} \infty = \infty$$

when $0 \neq x$, and

$$0 *^{\infty} \infty = \infty *^{\infty} 0 = 0$$

Definition 15 (Dominating Formula transformation) *Given a regular sentence F constructed using FF_g we construct the* dominating formula of *F, F_D, using the following transformation:*

$$\begin{aligned}
D(mon(X) : F, \emptyset, \emptyset) &\Longrightarrow mon(x) : D(F, \{x \leftarrow x\}, \{x \leftarrow x\}) \\
D(F \&\& G, \sigma_l^{\infty}, \sigma_h^{\infty}) &\Longrightarrow D(F, \sigma_l^{\infty}, \sigma_h^{\infty}) \&\& D(G, \sigma_l^{\infty}, \sigma_h^{\infty}) \\
D(F \wedge G, \sigma_l^{\infty}, \sigma_h^{\infty}) &\Longrightarrow D(F, \sigma_l^{\infty}, \sigma_h^{\infty}) \wedge D(G, \sigma_l^{\infty}, \sigma_h^{\infty}) \\
D({}^{\sim}F, \sigma_l^{\infty}, \sigma_h^{\infty}) &\Longrightarrow {}^{\sim}D(F, \sigma_l^{\infty}, \sigma_h^{\infty}) \\
D(\forall Y_{[b_1,b_2]} : F, \sigma_l^{\infty}, \sigma_h^{\infty}) &\Longrightarrow \forall Y_{[h_L(b_1), h_H(b_2)]} : D(F, \sigma_l^{\infty}\{Y \leftarrow h_L(b_1)\}, \sigma_2^{\infty}\{Y \leftarrow h_H(b_2)\}) \\
D(@X, \sigma_l^{\infty}, \sigma_h^{\infty}) &\Longrightarrow @X
\end{aligned}$$

where $h_L(b_1)$ is defined as,

$$h_L(b_1) = \min\{b_1\sigma_l^{\infty}, b_1\sigma_h^{\infty}\}$$

and where $h_H(b_2)$ is defined as,

$$h_H(b_2) = \max\{b_2\sigma_l^{\infty}, b_2\sigma_h^{\infty}\}$$

To make sure that F_D dominates F we need the h function to find the largest interval which can be constructed using the low and high instantiations of the variable. This will result in the largest interval of which, by Thm. 15, results in the largest evaluation of the function $g(a, b, c)$. It is not necessarily the case that the highest instantiation of a variable will be the one to result in the largest interval over which the quantifier will evaluate. Take for example, the interval $[y, x + 4]$. If y is evaluated over the interval $[x, x + 5]$, then choosing the highest instantiation value leads to an empty interval while choosing the lowest leads to the largest interval.

Definition 16 (bounding function) *Let F be a regular sentence constructed using VB_g and F_D be its dominating formula then we construct the bounding function $b(F_D)$ as follows:*

$$
\begin{aligned}
b(mon(x):F) &\Longrightarrow b(F,\{x \leftarrow 0\})\\
b(@y\&\&@z,\sigma) &\Longrightarrow 1\\
b(@y\&\&G,\sigma) &\Longrightarrow b(G,\sigma)\\
b(F\&\&@z,\sigma) &\Longrightarrow b(F,\sigma)\\
b(F\&\&G,\sigma) &\Longrightarrow b(F,\sigma) +_\infty b(G,\sigma)\\
b(@y \wedge @z,\sigma) &\Longrightarrow 1\\
b(@y \wedge G,\sigma) &\Longrightarrow b(G,\sigma)\\
b(F \wedge @z,\sigma) &\Longrightarrow b(F,\sigma)\\
b(F \wedge G,\sigma) &\Longrightarrow b(F,\sigma) +_\infty b(G,\sigma)\\
b(\,{}^\sim F,\sigma) &\Longrightarrow b(F,\sigma)\\
b(\forall y_{[x+a,x+b]} : F,\sigma) &\Longrightarrow g(a,b,w(F,\sigma\{y \leftarrow (x+b)\sigma\})) *^\infty b(F,\sigma\{y \leftarrow (x+b)\sigma\})\\
b(\forall y_{[x+a,\infty]} : F,\sigma) &\Longrightarrow \infty\\
b(\forall y_{[\infty,x+b]} : F,\sigma) &\Longrightarrow 0\\
b(\forall y_{[\infty,\infty]} : F,\sigma) &\Longrightarrow 0\\
b(@y,\sigma) &\Longrightarrow 1
\end{aligned}
$$

$$
\begin{aligned}
w(F\&\&G,\sigma^\infty) &\Longrightarrow \max\{w(F,x,\sigma^\infty), w(G,x,\sigma^\infty)\}\\
w(F \wedge G,\sigma^\infty) &\Longrightarrow \max\{w(F,x,\sigma^\infty), w(G,x,\sigma^\infty)\}\\
w(\,{}^\sim F,\sigma^\infty) &\Longrightarrow w(F,x,\sigma^\infty)\\
w(\forall Y_{[x+a,x+b]} : F,\sigma^\infty) &\Longrightarrow w(G,\sigma^\infty\{y \leftarrow (x+b)\sigma^\infty\})\\
w(\forall Y_{[x+a,\infty]} : F,\sigma^\infty) &\Longrightarrow w(G,\sigma^\infty\{y \leftarrow \infty\})\\
w(\forall Y_{[\infty,x+b]} : F,\sigma^\infty) &\Longrightarrow 0\\
w(\forall Y_{[\infty,\infty]} : F,\sigma^\infty) &\Longrightarrow 0\\
w(@y,\sigma^\infty) &\Longrightarrow y\sigma^\infty
\end{aligned}
$$

Theorem 16 *Given a regular formula F constructed using VB_g, then the upper bound space complexity of evaluating F is $O(b(D(F,\emptyset,\emptyset)))$.*

Proof.

We know by the work of Sec. 6, when the quantifier depth is one the theorem holds. We can assume that if we at most m formulae of quantifier depth one chained together by propositional connectives, then the theorem holds, we show for $m+1$. This is trivially two because the previous analysis ignores propositional connectives. Let as assume it holds for all formula with maximum quantifier depth n and show it holds for depth $n+1$. If we unroll the outermost quantifier the resulting instance has a reduced quantifier depth and by the previous induction we can handle each quantifier of depth n individually, thus, the theorem holds. To note it is no longer a Θ bound being that we convert F to its dominating formula.

$\square$

Corollary 4 *Given a regular formula F constructed using VB_g which does not include ∞, the upper bound space complexity of evaluating F is $O((a_I)^{2n})$ where n is the quantifier depth and a_I is the largest quantifier interval in the dominating formula of F.*

8 A Better Bound for VB$_g$

Instead of calculating the dominating formula, we could attempt to calculate the bounds of the quantifiers directly from the bounds provided in the formula. The major difference between the previous method and calculating directly from the formula is that we need to consider the entire interval of the quantifier and not just the upper bound. The important calculation for this function is $g(a, b, 0)$ being that it only counts the number of full intervals of the quantifier. We pass down the substitution of the variable $y \leftarrow a$. It is easy to generate a bounding formula, but the combinatorial properties are much more complex and it is harder to compute a complexity bound. Also, it is quite obvious to see that this formula bounds the formula.

Definition 17 (bounding function) *Let F be a regular sentence constructed using VB_g, then we construct the bounding function $b(F)$ as follows:*

$$
\begin{aligned}
b(mon(x) : F, \emptyset) &\Longrightarrow b(F, \{x \leftarrow x\}) \\
b(@y\&\&@z, \sigma) &\Longrightarrow 1 \\
b(F\&\&@y, \sigma) &\Longrightarrow b(F, \sigma) \\
b(@y\&\&G, \sigma) &\Longrightarrow b(G, \sigma) \\
b(F\&\&G, \sigma) &\Longrightarrow b(F, \sigma) +_\infty b(G, \sigma) \\
b(@y \wedge @z, \sigma) &\Longrightarrow 1 \\
b(F \wedge @y, \sigma) &\Longrightarrow b(F, \sigma) \\
b(@y \wedge G, \sigma) &\Longrightarrow b(G, \sigma) \\
b(F \wedge G, \sigma) &\Longrightarrow b(F, \sigma) +_\infty b(G, \sigma) \\
b(\,^\sim F, \sigma) &\Longrightarrow b(F, \sigma) \\
b(\forall y_{[y_1+a, y_2+b]} : F, \sigma) &\Longrightarrow g((y_1+a)\sigma\left\{x \leftarrow 0\right\}, (y_1+b)\sigma\left\{x \leftarrow 0\right\}, 0) *^\infty \\
&\quad\;\; \left(\sum_{i=(y_1+a)\sigma}^{(y_2+b)\sigma} b(F, \sigma\left\{y \leftarrow i\right\}) \right) \\
b(\forall y_{[y_1+a, \infty]} : F, \sigma) &\Longrightarrow \infty \\
b(\forall y_{[\infty, y_1+b]} : F, \sigma) &\Longrightarrow 0 \\
b(\forall y_{[\infty, \infty]} : F, \sigma) &\Longrightarrow 0 \\
b(@y, \sigma) &\Longrightarrow 1
\end{aligned}
$$

Theorem 17 *Given a regular formula F constructed using VB_g, then the upper bound space complexity of evaluating F is $O(b(F, \emptyset))$.*

Proof.

Given the nice properties of $g(a, b, 0)$ we know that we are generating the maximum number of branches, The rest follows from Thm. 16.

□

Let us refer to the bounding function of Sec. 7 as b_1 and the bounding function of this as b_2. We have not shown yet that b_1 always results in a value which is at least greater than b_2. Unlike the previous bounding function, we only consider how much memory is required to unroll the quantifier and not on how much memory is required to evaluate the formula completely, the latter was the goal of the work in Sec. 7. After considering how much memory it takes to unroll the quantifier, we perform one recursive call on the matrix of the quantifier for each value in the quantifiers interval and pass the substitution down substituting the value into the quantifiers variable. The amount of memory needed to unroll the quantifier is always less than the amount of memory needed to evaluate it. Also, rather than consider the largest interval, we consider the smaller intervals which means less memory consumption.

Example 2 *Let us now compare the bounding function provided in this section with the function from the previous section. We will consider the memory consumption of the following formula F:*

$$mon(x) : \forall y_{[x+1,x+5]} : \big(\big(\forall z_{[y,x+3]} :^{\sim} @z \wedge \wedge @y\big) \,\&\&$$
$$\big(\forall w_{[x-1,y+2]} : \big(^{\sim}@y \wedge \wedge \big(\forall m_{[y,w]} :^{\sim} @x\&\&@m\big)\big)\big)\big)$$

The dominating function of F, F_D, is the following:

$$mon(x) : \forall y_{[x+1,x+5]} : \big(\big(\forall z_{[x+1,x+3]} :^{\sim} @z \wedge \wedge @y\big) \,\&\&$$
$$\big(\forall w_{[x-1,x+7]} : \big(^{\sim}@y \wedge \wedge \big(\forall m_{[x+1,x+7]} :^{\sim} @x\&\&@m\big)\big)\big)\big)$$

Using the bounding function of Sec. 7, we can compute the following value for memory consumption:

$$g(1,5,7) * (g(1,3,5) + g(-1,7,7) * g(1,7,7)) = 25 * (12 + 34 * 28) = 24100$$

Now we will compute the bound using the function defined in this section. This is a bit more difficult given that there is a lot of branching. To simply the computation we will draw it as a tree where the arrows represent multiplication and the paths are summed together. By (a,b) we mean $g(a,b,0)$.

$$\begin{tikzcd}
&&&&&&& (1,5) \arrow[dllllll] \arrow[dlll] \arrow[d] \arrow[drrr] \arrow[drrrrr] &&&&&&& \\
& + \arrow[dl] \arrow[d] &&& + \arrow[dl] \arrow[d] &&& + \arrow[dl] \arrow[d] &&& + \arrow[dl] \arrow[d] && + \arrow[d] \arrow[dr] && \\
(1,3) & (-1,3) \arrow[dl] \arrow[d] \arrow[dr] && (2,3) & (-1,4) \arrow[dl] \arrow[d] \arrow[dr] && (3,3) & (-1,5) \arrow[dl] \arrow[d] \arrow[dr] && 0 & (-1,6) \arrow[dl] \arrow[d] \arrow[dr] && 0 & (-1,7) \arrow[dl] \arrow[d] \arrow[dr] & \\
(1,1) & (1,2) & (1,3) & (2,2) & (2,3) & (2,4) & (3,3) & (3,4) & (3,5) & (4,4) & (4,5) & (4,6) & (5,5) & (5,6) & (5,7)
\end{tikzcd}$$

Now we can add up the individual branches to get the memory consumption for this formula. This is represented by the following formula.

$$g(1,5,0) * \left(\sum_{i=1}^{3} (g(i,3,0)) + \sum_{i=0}^{4} \left(g(-1,3+i,0) * \left(\sum_{j=i+1}^{i+3} g(i+1,j,0) \right) \right) \right) =$$

$$5 * \left(9 + \sum_{i=0}^{4} \left((i+3) * (i+1) + (i+3) * \left(\sum_{j=i+2}^{i+3} g(i+1, j, 0)\right)\right)\right) =$$

$$5 * \left(9 + \sum_{i=0}^{4} ((i+3) * (i+1) + (i+3) * (i+2) + (i+3) * (i+3))\right) =$$

$$5*(9 + (3+6+9) + (8+12+16) + (15+20+25) + (24+30+36) + (35+42+49)) =$$

$$5 * (9 + 18 + 36 + 60 + 90 + 126) =$$

$$5 * (339) = 1695$$

Thus, as one can see the bound provided by the function in this section is a vast improvement over the previous bound, i.e. $24100 \gg 1695$.

9 Bounding FF$_g$

We have not yet discussed FF_g because up until now, we have not allowed constants in the bounds of the quantifier, i.e. both the lower and upper bound must have a variable or be infinity. When we allow constants in the grammar we have the follow cases to deal with $[a, b]$, $[x \pm a, b]$, $[a, x \pm b]$, and $[a, \infty]$. Also it is to be noted that allowing constants effects the already considered work. In the case of the function $g(a, b, c)$ if instead of $x + c$ we allowed just c, this would be equivalent to setting $c = 0$ because at some point every thing will instantly evaluate. Thus, we would just write $g(a, b, 0)$ for a constant c.

The $[x \pm a, b]$ case is trivial, being that at some point $x \pm a > b$ and then the quantifier will have zero instance. By Thm. 15, the interval is decreasing, and thus, the evaluation of the $g(a, b, c)$ will decrease too, all the way to zero. However we also have to consider what is inside the matrix of a quantifier with the interval $[x \pm a, b]$. We have to remember that even though the number of instances it generates at each step converges to zeros, it still generates some instances. The number of generated instances is easily computed to be an upper bounded of $O((b-a+1)^2)$. This is important because if the matrix contains a quantifier requiring infinite space than we need to keep track of it. However, unlike quantifiers which have an infinite upper bound without constants, these quantifiers in the matrix of a $[x \pm a, b]$ might no longer require infinite space and in some cases act more like a stream variable rather than a bounded variable. This occurs because only a finite number of instances of these quantifiers are generated. Being that everything has been reduced to x or infinity by the dominating function transformation, we know that such quantifiers can change the calculation of the g function by increasing the 3$^{\text{rd}}$ component. But, by Thm. 15, this is still an issue and leads to infinity anyway.

By a similar argument, it is easy to see that $[a, \infty]$ is also trivial because the interval is constantly increase and thus, the overall memory needed will be ∞. However, in the case $[a, b]$, the amount of memory needed is zero only if the variables in the matrix of the quantifier are independent of the stream variable. This means that the largest value in the matrix of the quantifier is constant, i.e. c is constant. Otherwise, by Thm. 15, the value c will increase and thus a quantifier with the interval $[a, b]$ will require space the same way as a negative interval requires it.

The last case to handle is $[a, x \pm b]$. When c is constant at some point we can ignore everything which is below the current position and this case becomes equivalent to $[x, x \pm b]$. However, when c is not constant, more specifically greater than zero, such a transformation is not possible and it is essentially infinity.

Using this outline above we can construct a bounding function for FF_g, though we need to add a few more complex expressions to be explained later.

Definition 18 (bounding function) *Let F be a regular sentence constructed using FF_g and F_D be its dominating formula then we construct the bounding function $b(F_D)$ as follows:*

$$
\begin{aligned}
b(mon(x) : F, x, \emptyset) &\Longrightarrow b(F, x, \emptyset) \\
b(@y\&\&@z, x, \sigma) &\Longrightarrow 1 \\
b(@y\&\&G, x, \sigma) &\Longrightarrow b(G, x, \sigma) \\
b(F\&\&@z, x, \sigma) &\Longrightarrow b(F, x, \sigma) \\
b(F\&\&G, x, \sigma) &\Longrightarrow b(F, x, \sigma) +_\infty b(G, x, \sigma) \\
b(@y \wedge @z, x, \sigma) &\Longrightarrow 1 \\
b(@y \wedge G, x, \sigma) &\Longrightarrow b(G, x, \sigma) \\
b(F \wedge @z, x, \sigma) &\Longrightarrow b(F, x, \sigma) \\
b(F \wedge G, x, \sigma) &\Longrightarrow b(F, x, \sigma) +_\infty b(G, x, \sigma) \\
b(\,{}^\sim F, x, \sigma) &\Longrightarrow b(F, x, \sigma) \\
b(\forall y_{[\infty,\infty]} : F, x, \sigma) &\Longrightarrow 0 \\
b(\forall y_{[\infty,b_2]} : F, x, \sigma) &\Longrightarrow 0 \\
b(\forall y_{[b_1,\infty]} : F, x, \sigma) &\Longrightarrow \infty \\
b(\forall y_{[a,b]} : F, x, \sigma) &\Longrightarrow (g(-a, -b, w(F, x, \sigma\{y \leftarrow b\})) + 1) *^\infty b(F, x, \sigma\{y \leftarrow b\}) \\
b(\forall y_{[x+a,b]} : F, x, \sigma) &\Longrightarrow \frac{(b-a+1)(b-a+2)}{2} *^\infty b(F, x, \sigma\{y \leftarrow b\}) \\
b(\forall y_{[a,x+b]} : F, x, \sigma) &\Longrightarrow (w(F, x, \sigma\{y \leftarrow x+b\}) *^\infty \infty) +^\infty \\
&\qquad g(0, b, 0) *^\infty b(F, x, \sigma\{y \leftarrow x+b\}) \\
b(\forall y_{[x+a,x+b]} : F, x, \sigma) &\Longrightarrow g(a, b, w(F, x, \sigma\{y \leftarrow x+b\})) *^\infty b(F, x, \sigma\{y \leftarrow x+b\}) \\
b(@y, \sigma) &\Longrightarrow 1
\end{aligned}
$$

$$
\begin{aligned}
w(F\&\&G, x, \sigma^{\infty}) &\Longrightarrow \max\left\{w(F, x, \sigma^{\infty}), w(G, x, \sigma^{\infty})\right\}\\
w(F \wedge G, x, \sigma^{\infty}) &\Longrightarrow \max\left\{w(F, x, \sigma^{\infty}), w(G, x, \sigma^{\infty})\right\}\\
w(\;{}^{\sim}F, x, \sigma^{\infty}) &\Longrightarrow w(F, x, \sigma^{\infty})\\
w(\forall y_{[\infty,\infty]} : F, x, \sigma^{\infty}) &\Longrightarrow 0\\
w(\forall y_{[\infty,b_2]} : F, x, \sigma^{\infty}) &\Longrightarrow 0\\
w(\forall y_{[b_1,\infty]} : F, x, \sigma^{\infty}) &\Longrightarrow w(G, x, \sigma^{\infty}\left\{y \leftarrow \infty\right\})\\
w(\forall Y_{[b_1,b_2]} : F, x, \sigma^{\infty}) &\Longrightarrow w(G, x, \sigma^{\infty}\left\{y \leftarrow b_2\right\})\\
w(@y, x, \sigma^{\infty}) &\Longrightarrow (y\sigma^{\infty}\left\{x \leftarrow y\sigma^{\infty}\right\} - y\sigma^{\infty}\left\{x \leftarrow 0\right\})\left\{x \leftarrow 0\right\}
\end{aligned}
$$

One of the expressions we were referring too is the following:

$$(y\sigma^{\infty}\left\{x \leftarrow y\sigma^{\infty}\right\} - y\sigma^{\infty}\left\{x \leftarrow 0\right\})\left\{x \leftarrow 0\right\}$$

in the definition of w. This expression is a way of differentiating between a c which is constant without variable, constant with variable, or infinite. if one computes this for various cases, when c is constant without variable we get 0, if it is constant with variable we get $0 \leq c$, and otherwise infinity. The other expression we were referring to is the evaluation $\forall y_{[a,x+b]} : F$ which is the following:

$$(w(F, x, \sigma\left\{y \leftarrow x + b\right\}) *^{\infty} \infty) +^{\infty} g(0, b, 0) *^{\infty} b(F, x, \sigma\left\{y \leftarrow x + b\right\})$$

Essentially it states that given a value of c greater than zero the result is infinity, otherwise we evaluate the quantifier given that c is zero.

Theorem 18 *Given a regular formula F constructed using FF_g, then the upper bound space complexity of evaluating F is $O(b(F, x, \emptyset))$.*

Proof.

Follows nicely from previous discussed results.

□

As we did with VB_g a more precise bound can be constructed.

Definition 19 (bounding function) *Let F be a regular sentence constructed using FF_g, then we can construct the bounding function $b(F)$ as follows:*

$$
\begin{aligned}
b(mon(x) : F, x, \emptyset) &\Longrightarrow b(F, x, \emptyset)\\
b(@y\&\&@z, x, \sigma) &\Longrightarrow 1\\
b(@y\&\&G, x, \sigma) &\Longrightarrow b(G, x, \sigma)\\
b(F\&\&@z, x, \sigma) &\Longrightarrow b(F, x, \sigma)\\
b(F\&\&G, x, \sigma) &\Longrightarrow b(F, x, \sigma) +_\infty b(G, x, \sigma)\\
b(@y \wedge @z, x, \sigma) &\Longrightarrow 1\\
b(@y \wedge G, x, \sigma) &\Longrightarrow b(G, x, \sigma)\\
b(F \wedge @z, x, \sigma) &\Longrightarrow b(F, x, \sigma)\\
b(F \wedge G, x, \sigma) &\Longrightarrow b(F, x, \sigma) +_\infty b(G, x, \sigma)\\
b(\,{}^\sim F, x, \sigma) &\Longrightarrow b(F, x, \sigma)\\
b(\forall y_{[\infty,\infty]} : F, x, \sigma) &\Longrightarrow 0\\
b(\forall y_{[\infty,b_2]} : F, x, \sigma) &\Longrightarrow 0\\
b(\forall y_{[b_1,\infty]} : F, x, \sigma) &\Longrightarrow \infty\\
b(\forall y_{[a,b]} : F, x, \sigma) &\Longrightarrow (g(-a, -b, w(F, x, \sigma\{y \leftarrow b\})) + 1)*^\infty\\
&\qquad \left(\sum_{i=a}^{b} b(F, x, \sigma\{y \leftarrow i\})\right)\\
b(\forall y_{[x+a,b]} : F, x, \sigma) &\Longrightarrow \frac{(b-a+1)(b-a+2)}{2} *^\infty b(F, x, \sigma\{y \leftarrow b\})\\
b(\forall y_{[a,y_1+b]} : F, x, \sigma) &\Longrightarrow (w(F, x, \sigma\{y \leftarrow (y_1+b)\{x \leftarrow 0\}\}) *^\infty \infty) +^\infty\\
&\qquad g(0, b, 0) *^\infty \left(\sum_{i=0}^{(y_1+b)\sigma\{x\leftarrow 0\}} b(F, x, \sigma\{y \leftarrow i\})\right)\\
b(\forall y_{[y_1+a,y_2+b]} : F, \sigma) &\Longrightarrow g((y_1+a)\sigma\{x \leftarrow 0\}, (y_1+b)\sigma\{x \leftarrow 0\}, 0)*^\infty\\
&\qquad \left(\sum_{i=(y_1+a)\sigma\{x\leftarrow 0\}}^{(y_2+b)\sigma\{x\leftarrow 0\}} b(F, x, \sigma\{y \leftarrow i\})\right)\\
b(@y, \sigma) &\Longrightarrow 1
\end{aligned}
$$

$$\begin{aligned}
w(F\&\&G, x, \sigma^{\infty}) &\Longrightarrow \max\{w(F, x, \sigma^{\infty}), w(G, x, \sigma^{\infty})\}\\
w(F \wedge G, x, \sigma^{\infty}) &\Longrightarrow \max\{w(F, x, \sigma^{\infty}), w(G, x, \sigma^{\infty})\}\\
w(\ {}^{\sim}F, x, \sigma^{\infty}) &\Longrightarrow w(F, x, \sigma^{\infty})\\
w(\forall y_{[\infty,\infty]} : F, x, \sigma^{\infty}) &\Longrightarrow 0\\
w(\forall y_{[\infty,b_2]} : F, x, \sigma^{\infty}) &\Longrightarrow 0\\
w(\forall y_{[b_1,\infty]} : F, x, \sigma^{\infty}) &\Longrightarrow w(G, x, \sigma^{\infty}\{y \leftarrow \infty\})\\
w(\forall Y_{[b_1,b_2]} : F, x, \sigma^{\infty}) &\Longrightarrow w(G, x, \sigma^{\infty}\{y \leftarrow b_2\})\\
w(@y, x, \sigma^{\infty}) &\Longrightarrow (y\sigma^{\infty}\{x \leftarrow y\sigma^{\infty}\} - y\sigma^{\infty}\{x \leftarrow 0\})\{x \leftarrow 0\}
\end{aligned}$$

10 Conclusion

What we have provided in this work is a space complexity analysis for VB_g, found in Sec. 7, a more precise space complexity analysis for VB_g, though a bit more complex, in Sec. 8, and a space complexity analysis for FF_g found in Sec. 9. There are only a few cases which take an infinite amount of space, namely when the upper bound of the quantifier is infinity and when the quantifier interval is $[a, x + b]$. Even when the matrix of the quantifier is instantly evaluated there is a build up of instants in memory in the case when the upper bound of the quantifier is infinity.

These infinite memory cases happen to be very important cases as well, because they are necessary to define finite unbounded temporal properties, such as *some point in the future*. We are planning to investigate methods to evaluate formulae which take infinite space such that we only allow for a constant growth rate of memory. This may be possible if we only consider fragments of LogicGuard's core language such as the two variable fragment. The two variable fragment of first order logic with an ordering predicate has nice properties which might be of importance to semantic evaluation as needed by LogicGuard. Even with the inclusion of infinity, we can work with a two variable logic if we consider the addition of free variable, i.e. infinity is replaced with a free variable. Essentially, we can remove infinity from the language and instead of infinity, we replace the symbols with free variables, however, only a fixed number of them. These free variables can be any position in the stream which is larger then every other stream variable, i.e. a type of finite but unbounded analysis. Along with a constant growth analysis, this would allow the the system to analyse the formula within a reasonable amount of space. If the formulae we are evaluating has an inductive structure, we can first show that it is satisfiable, and from the satisfiability result compare the current state of the stream to the possible models. This is a sort of backwards evaluation of the problem. However, this still has many problems because we do not know if the formula has only infinite models. We leave further discussion to future work.

Given that the core language of LogicGuard has been completely analysed, we plan to move on to evaluation of the full language of which has a few generalization over the quantifiers used in the core language. The idea is that one can use these results to provide users of Logicguard an automatic static space analysis at the run time of a monitor. Thus, the users will know what to expect in terms of space consumption.

Also, we would like to generalize this work to a time complexity analysis. As of yet, a reasonable metric for such an analysis has not been constructed. We leave the search for such a metric and the analysis it self to future work.

References

[1] Temur Kutsia and Wolfgang Schreiner. Verifying the Soundness of Resource Analysis for LogicGuard Monitors (Revised Version) . RISC Report Series 14-08, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Schloss Hagenberg, 4232 Hagenberg, Austria, 2014.

[2] Wolfgang Schreiner. Logicguard ii, 2015. http://www.risc.jku.at/projects/LogicGuard2/.

[3] Wolfgang Schreiner, Temur Kutsia, Michael Krieger, Ahmad Bashar, Helmut Otto, and Martin Rummerstorfer. Monitoring Network Traffic by Predicate Logic. Technical report, Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, September 2014.