Systematic Derivation of Bounds and Glue Constraints for Time-Series Constraints

Ekaterina Arafailova, Nicolas Beldiceanu, Mats Carlsson, Pierre Flener, María Andreína Francisco Rodríguez, Justin Pearson, Helmut Simonis

Systematic Derivation of Bounds and Glue Constraints for Time-Series Constraints*

Ekaterina Arafailova[1], Nicolas Beldiceanu[1], Mats Carlsson[2], Pierre Flener[3], María Andreína Francisco Rodríguez[3], Justin Pearson[3], and Helmut Simonis[4]

[1] TASC (CNRS/INRIA), Mines Nantes, FR – 44307 Nantes, France
{Ekaterina.Arafailova,Nicolas.Beldiceanu}@mines-nantes.fr
[2] SICS, P.O. Box 1263, SE – 164 29 Kista, Sweden
Mats.Carlsson@sics.se
[3] Uppsala University, Dept of Information Technology, SE – 751 05 Uppsala, Sweden
{Pierre.Flener,Maria.Andreina.Francisco,Justin.Pearson}@it.uu.se
[4] Insight Centre for Data Analytics, University College Cork, Ireland
Helmut.Simonis@insight-centre.org

Abstract. Integer time series are often subject to constraints on the aggregation of the integer features of all occurrences of some pattern within the series. For example, the number of inflexions may be constrained, or the sum of the peak maxima, or the minimum of the peak widths. It is currently unknown how to maintain domain consistency efficiently on such constraints. We propose parametric ways of systematically deriving glue constraints, which are a particular kind of implied constraints, as well as aggregation bounds that can be added to the decomposition of time-series constraints [5]. We evaluate the beneficial propagation impact of the derived implied constraints and bounds, both alone and together.

1 Introduction

A *time series* is here a sequence of integers, corresponding to measurements taken over a time interval. Time series are common in many application areas, such as the output of electric power stations over multiple days [8], or the manpower required in a call-centre [3].

We showed in [5] that many constraints $\gamma(\langle X_1, \ldots, X_n \rangle, N)$ on an unknown time series $X = \langle X_1, \ldots, X_n \rangle$ of given length n can be specified by a triple $\langle \sigma, f, g \rangle$, where σ is a regular expression over the alphabet $\Sigma = \{`<', `=', `>'\}$ (we assume the reader is familiar with regular expressions and automata [12]), while $f \in \{\texttt{max}, \texttt{min}, \texttt{one}, \texttt{surface}, \texttt{width}\}$ is called a *feature*, and $g \in \{\texttt{Max}, \texttt{Min}, \texttt{Sum}\}$

* We thank the anonymous referees for their helpful comments. The authors in Nantes are supported by the EU H2020 programme under grant 640954 for project GRACeFUL and by the Gaspard-Monge programme. The authors in Uppsala are supported by the Swedish Research Council (VR) under grants 2011-6133 and 2012-4908. The last author is supported by Science Foundation Ireland (SFI) under grant SFI/10/IN.1/I3032; the Insight Centre for Data Analytics is supported by SFI under grant SFI/12/RC/2289.

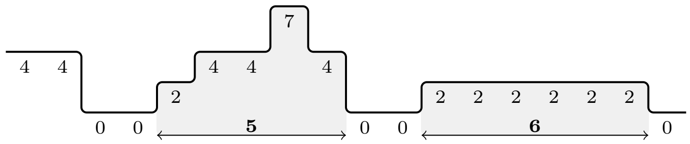


Fig. 1: MIN_WIDTH_PEAK$(5, \langle 4,4,0,0,2,4,4,7,4,0,0,2,2,2,2,2,2,0\rangle)$

is called an *aggregator*. Let the sequence $S = \langle S_1, \ldots, S_{n-1}\rangle$, called the *signature* and containing *signature variables*, be linked to X via the *signature constraints* $(X_i < X_{i+1} \Leftrightarrow S_i = \text{‘}<\text{’}) \wedge (X_i = X_{i+1} \Leftrightarrow S_i = \text{‘}=\text{’}) \wedge (X_i > X_{i+1} \Leftrightarrow S_i = \text{‘}>\text{’})$ for all $i \in [1, n-1]$. A σ*-pattern* is a sub-series of X that *corresponds to* a maximal occurrence of σ within S. Integer variable N is constrained to be the aggregation, computed using g, of the list of values of feature f for all σ-patterns in X. A set of 20 regular expressions is considered. We name a time-series constraint specified by $\langle \sigma, f, g\rangle$ as g_f_σ.

Example 1. The time series $X = \langle 4,4,0,0,2,4,4,7,4,0,0,2,2,2,2,2,2,0\rangle$ has the signature $S =$ ‘=>=<<=<>>=<=====>’. Consider the regular expression `Peak` = ‘<(<|=)*(>|=)*>’: a `Peak`-pattern, called a *peak*, within a time series corresponds, except for its first and last elements, to a maximal occurrence of `Peak` in the signature, and the `width` feature value of a peak is its number of elements. The time series X contains two peaks, namely $\langle 2,4,4,7,4\rangle$ and $\langle 2,2,2,2,2,2\rangle$, visible the way X is plotted in Figure 1, of widths 5 and 6 respectively, hence the minimal-width peak, obtained by using the aggregator `Min`, has width $N = 5$: the underlying constraint is named MIN_WIDTH_PEAK. □

After recalling in Section 2 further required background material on time-series constraints $g\_f\_\sigma(\langle X_1, \ldots, X_n\rangle, N)$, the contributions of this paper are ways of systematically deriving parametric implied constraints and bounds:

- We show in Section 3 how to derive systematically implied constraints, parametrised by aggregator g and feature f, for any regular expression σ.
- We give in Section 4 a methodology for systematically deriving bounds, parametrised by σ, on the variable N, for any pair of g and f, and then we demonstrate our methodology on the case when $g =$ `Max` and $f =$ `min`.
- We evaluate in Section 5 the beneficial propagation impact of the derived implied constraints and bounds, both alone and together.

In Section 6, we conclude and discuss other related work. The implied constraints and bounds for all time-series constraints are in [2].

2 Background: Automata for Time-Series Constraints

In [5], we showed how to synthesise a deterministic finite automaton, enriched with accumulators [7], from any triple $\langle \sigma, f, g\rangle$ that specifies a time-series constraint. We now discuss the required background concepts using an example, namely the regular expression `Peak` = ‘<(<|=)*(>|=)*>’ of Example 1.

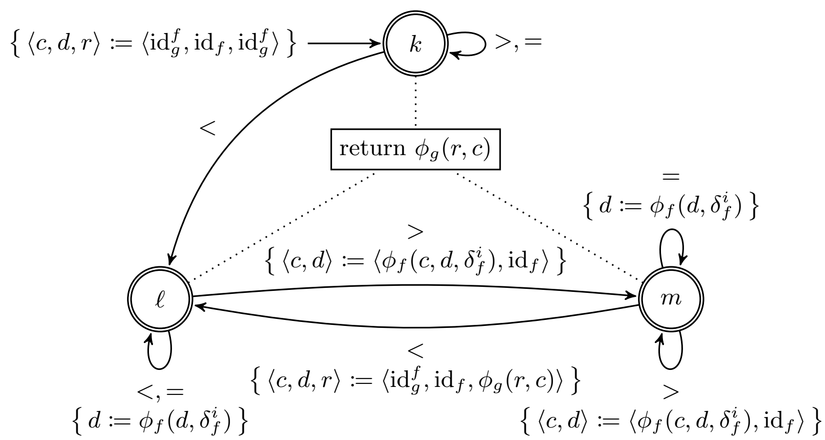


Fig. 2: Synthesised automaton for any $g_f_$PEAK constraint

The synthesised automaton for any $g_f_$PEAK constraint is in Figure 2. It returns the aggregation, using g, of the values of feature f for all `Peak`-patterns corresponding to the occurrences of `Peak` within an input word over the alphabet $\Sigma = \{\text{`}<\text{'}, \text{`}=\text{'}, \text{`}>\text{'}\}$. The start state is $k$, annotated within braces by the initialisation of three accumulators: at any moment, accumulator $c$ stores the feature value of the *current* `Peak`-pattern while $d$ stores the feature value of a *potential* part of a `Peak`-pattern, and $r$ stores the aggregated *result* for the feature values of the already encountered `Peak`-patterns. A transition is depicted by an arrow between two states and is annotated by a consumed alphabet symbol and, within braces, an accumulator update. The constants and operators appearing in the accumulator initialisation and updates are listed in Table 1; the binary operators ϕ_f and ϕ_g are used with arbitrary arity throughout this paper, in order to reduce the amount of parentheses. All states are accepting, an accepting state being marked by a double circle. Hence this automaton accepts the language Σ^*, but accepted words may be distinguished by the value of the returned expression, given within a box linked to all states. Note that the size of this automaton does *not* depend on the length of the input word.

In [7], we showed how to use an automaton with accumulators in order to decompose a constraint such as $g_f_$PEAK$(\langle X_1, \dots, X_n \rangle, N)$ into signature constraints, linking $\langle X_1, \dots, X_n \rangle$ to introduced signature variables $\langle S_1, \dots, S_{n-1} \rangle$, as well as arithmetic and TABLE constraints, linking $\langle S_1, \dots, S_{n-1} \rangle$ and N to introduced state variables Q_i and tuples $\langle C_i, D_i, R_i \rangle$ of accumulator variables, respectively denoting the automaton state and accumulator values $\langle c, d, r \rangle$ after consuming S_i. It is still unknown how to maintain domain consistency efficiently in general on this decomposition (see [7] for an analysis), hence implied constraints can help achieve more propagation, as we already showed in [6,11].

Feature f	id_f	$\min_f$	$\max_f$	ϕ_f	δ_f^i
`one`	1	1	1	1	1
`width`	0	0	$+\infty$	$+$	1
`surface`	0	$-\infty$	$+\infty$	$+$	X_i
`max`	$-\infty$	$-\infty$	$+\infty$	max	X_i
`min`	$+\infty$	$-\infty$	$+\infty$	min	X_i

Aggregator g	ϕ_g	id_g^f
`Max`	max	$\min_f$
`Min`	min	$\max_f$
`Sum`	$+$	0

Table 1: (Left) Features: identity, minimum, and maximum values; operators ϕ_f and δ_f^i recursively define the feature value v_u of a time series $\langle X_\ell, \dots, X_u\rangle$ by $v_\ell = \phi_f(\mathrm{id}_f, \delta_f^\ell)$ and $v_i = \phi_f(v_{i-1}, \delta_f^i)$ for $i > \ell$, where δ_f^i is the contribution of X_i to v_u. (Right) Aggregators: operators & identity values relative to a feature f.

3 Glue Constraints for Time-Series Constraints

In [6] we derived an implied constraint, called a *glue constraint*, that can be added to the decomposition of a constraint specified by an automaton with accumulators: the derivation was *ad hoc* in most cases. In this paper, we introduce *parametric glue constraints* and show that they can be derived *automatically* for time-series constraints, which we introduced a year later in [5].

Example 2. We can explain the key insight using Example 1. The reverse of its time series X is $X' = \langle 0,2,2,2,2,2,2,0,0,4,7,4,4,2,0,0,4,4\rangle$ and has the signature S^{mir} = ‘<=====>=<<>=>>=<=’, which we will call the *mirror* of the original signature S. The automaton of Figure 2 returns the same value whether it consumes a signature or its mirror: the peaks of X are the reverses of the peaks of X' and the aggregation of their feature values is the same because all the operators ϕ_f and ϕ_g are commutative. We have this property for 19 of the 20 regular expressions in [5]. The idea now is to derive an implied constraint, which we will call a *glue constraint*, between the three accumulator triples of such an automaton after it has consumed (*i*) a signature w, (*ii*) a prefix w_1 of w, and (*iii*) the mirror of the corresponding suffix w_2 of w. For instance, let us split S into the prefix P = ‘=>=<<=<’ and the suffix T = ‘>>=<=====>’, which has the mirror T^{mir} = ‘<=====>=<<’. If we instantiate the automaton $\mathcal{A}$ of Figure 2 for the MIN_WIDTH_PEAK constraint, that is with f = `width` and g = `Min`, then $\mathcal{A}$ has the accumulator triples $\langle c,d,r\rangle = \langle 6,0,5\rangle$ after consuming S, and $\langle c_1,d_1,r_1\rangle = \langle +\infty,3,+\infty\rangle$ after consuming P, and $\langle c_2,d_2,r_2\rangle = \langle +\infty,1,6\rangle$ after consuming T^{mir}. The value $\phi_g(r,c) = \min(5,6) = 5$ returned by $\mathcal{A}$ on S can also be computed using the formula $\phi_g(r_1, r_2, \phi_f(d_1, d_2, \delta_f^i))$, that is $\min(+\infty, 6, 3+1+1)$. That formula computes the minimum width of the following three peaks:

- the minimum-width peak corresponding to P, which actually has *no* occurrence of `Peak` = ‘<(<|=)*(>|=)*>’, hence $r_1 = \mathrm{id}_f = +\infty$;
- the minimum-width peak corresponding to T^{mir}, whose only occurrence of `Peak` gives width $r_2 = 6$;
- the peak that is *created* by concatenating the following two potential peaks:

- the potential occurrence of Peak at the end of P, giving width $d_1 = 3$;
- the potential occurrence of Peak at the end of T^{mir}, giving $d_2 = 1$; note that if we feed T rather than T^{mir} to $\mathcal{A}$, then $\langle c_2, d_2, r_2 \rangle = \langle 6, 0, +\infty \rangle$ and d_2 reflects information about the *end* of T, rather than its beginning, hence the created peak is missed;

but the contribution $\delta_f^i = 1$ (with $i = |P| + 1$) is required to compensate for the fact that $d_1 + d_2 = 4$ under-measures the width 5 of the created peak. □

We now formalise this insight, and add scenarios other than creation.

Definition 1 (mirror). *The* mirror *of a language L over $\Sigma = \{`<', `=', `>'\}$, denoted by L^{mir}, consists of the mirrors of all the words in L, where the* mirror *of a word or regular expression has the reverse order of its symbols and has all occurrences of the symbol '$<$' flipped into '$>$' and vice versa.*

We denote by $\mathcal{L}(\sigma)$ the regular language defined by a regular expression σ.

Definition 2 (state language). *Let q be a state of an automaton $\mathcal{A}$. The* language accepted *by q, denoted by $\mathcal{L}_q$, is the regular language accepted when q is made to be the only accepting state of $\mathcal{A}$.*

Example 3. Consider the automaton in Figure 2. We have $\mathcal{L}_k = \mathcal{L}((>|=)^*)$, $\mathcal{L}_\ell = \Sigma^*\mathcal{L}(<(<|=)^*)$, and $\mathcal{L}_m = \Sigma^*\mathcal{L}(\texttt{Peak})\mathcal{L}(=^*)$, where $\texttt{Peak} = `<(<|=)^*(>|=)^*>'$ is the regular expression for peaks. Standard algorithms of automata theory [12] can be used to compute state languages: we use the FAdo tool [1] to do so, as well as to check the language equalities stated in the following three examples. □

We concatenate two words by writing them side by side, with an implicit infix concatenation operator between them. The concatenation L_1L_2 of two languages L_1 and L_2 is the language of all words w_1w_2 where $w_1 \in L_1$ and $w_2 \in L_2$.

Definition 3 (extension). *We say that the concatenation L_1L_2* extends *a regular expression σ if and only if for any non-empty words $w_1 \in L_1$ and $w_2 \in L_2$ there exist a non-empty suffix s of w_1 and a non-empty prefix p of w_2 such that $sp \in \mathcal{L}(\sigma)$ and either s starts with the last occurrence of σ in w_1, where we say that L_1L_2* extends the last *σ in L_1, or p ends with the first occurrence of σ in w_2, where we say that L_1L_2* extends the first *σ in L_2, or both.*

Example 4. Consider the regular expression $\texttt{Peak} = `<(<|=)^*(>|=)^*>'$. Every word w_1 in $L_1 = \Sigma^*\mathcal{L}(\texttt{Peak})\mathcal{L}(=^*)$ has a suffix s in $\mathcal{L}(\texttt{Peak})\mathcal{L}(=^*)$. Every word w_2 in $L_2 = \mathcal{L}((>|=)^*>)\Sigma^*$ has a prefix p in $\mathcal{L}((>|=)^*>)$. The concatenation sp is in $\mathcal{L}(\texttt{Peak})\mathcal{L}(=^*)\mathcal{L}((>|=)^*>)$, which is a subset of $\mathcal{L}(\texttt{Peak})$, hence L_1L_2 extends the last Peak in L_1. Note that p cannot end with any occurrence of Peak, hence L_1L_2 does not extend any Peak in L_2. □

Definition 4 (creation). *We say that the concatenation L_1L_2* creates *a regular expression σ if and only if for any non-empty words $w_1 \in L_1$ and $w_2 \in L_2$, there exist a non-empty suffix s of w_1 and a non-empty prefix p of w_2 such that $sp \in \mathcal{L}(\sigma)$ but neither does s start with an occurrence of σ in w_1 nor does p end with an occurrence of σ in w_2.*

Example 5. Consider again the regular expression `Peak` = ‘<(<|=)*(>|=)*>’. Every word w_3 in $L_3 = \Sigma^* \mathcal{L}(<(<|=)^*)$, such as P of Example 2, has a suffix s in $\mathcal{L}(<(<|=)^*)$. Every word w_4 in $L_4 = \mathcal{L}((>|=)^*>)\Sigma^*$, such as T^{mir} of Example 2, has a prefix p in $\mathcal{L}((>|=)^*>)$. The concatenation sp is in $\mathcal{L}(<(<|=)^*(>|=)^*>)$, which is equal to $\mathcal{L}(\texttt{Peak})$. However, neither can s start with an occurrence of `Peak` nor can p end with an occurrence of `Peak`: hence $L_3 L_4$ does not extend `Peak`, but instead creates `Peak`. □

We now give the glue constraint for a time-series constraint specified by $\langle \sigma, f, g \rangle$: it is specific to regular expression σ but generic in f and g. Let an automaton $\mathcal{A}$ for σ reach state $\overrightarrow{Q}$ and accumulator values $\langle \overrightarrow{C}, \overrightarrow{D}, \overrightarrow{R} \rangle$ on a prefix of a word w, as well as state $\overleftarrow{Q}$ and accumulator values $\langle \overleftarrow{C}, \overleftarrow{D}, \overleftarrow{R} \rangle$ on the mirror of the corresponding suffix of w. The value N returned by $\mathcal{A}$ on the entire word w is constrained by $N = \phi_g(\overrightarrow{R}, \overleftarrow{R}, \Gamma)$, where Γ is called the *glue expression* and is defined as follows:

1. if $\mathcal{L}_{\overrightarrow{Q}} \mathcal{L}^{\text{mir}}_{\overleftarrow{Q}}$ extends σ, then:
   (a) if $\mathcal{L}_{\overrightarrow{Q}} \mathcal{L}^{\text{mir}}_{\overleftarrow{Q}}$ extends both the last σ in $\mathcal{L}_{\overrightarrow{Q}}$ and the first σ in $\mathcal{L}^{\text{mir}}_{\overleftarrow{Q}}$, then $\Gamma = \phi_f(\overrightarrow{C}, \overleftarrow{C}, \overrightarrow{D}, \overleftarrow{D}, \delta^i_f)$;
   (b) if $\mathcal{L}_{\overrightarrow{Q}} \mathcal{L}^{\text{mir}}_{\overleftarrow{Q}}$ extends only the last σ in $\mathcal{L}_{\overrightarrow{Q}}$, then $\Gamma = \phi_f(\overrightarrow{C}, \overrightarrow{D}, \overleftarrow{D}, \delta^i_f)$;
   (c) if $\mathcal{L}_{\overrightarrow{Q}} \mathcal{L}^{\text{mir}}_{\overleftarrow{Q}}$ extends only the first σ in $\mathcal{L}^{\text{mir}}_{\overleftarrow{Q}}$, then $\Gamma = \phi_f(\overleftarrow{C}, \overrightarrow{D}, \overleftarrow{D}, \delta^i_f)$;
2. if $\mathcal{L}_{\overrightarrow{Q}} \mathcal{L}^{\text{mir}}_{\overleftarrow{Q}}$ creates σ, then $\Gamma = \phi_f(\overrightarrow{D}, \overleftarrow{D}, \delta^i_f)$;
3. if $\mathcal{L}_{\overrightarrow{Q}} \mathcal{L}^{\text{mir}}_{\overleftarrow{Q}}$ neither creates nor extends σ, then $\Gamma = \phi_g(\overrightarrow{C}, \overleftarrow{C})$.

Note that these rules are exhaustive and mutually exclusive, because the final conditions of extension and creation are negations of each other.

Example 6. Consider the regular expression `Peak` = ‘<(<|=)*(>|=)*>’, the automaton $\mathcal{A}$ in Figure 2, and the languages in Example 3 for the states of $\mathcal{A}$.

- Consider $\overrightarrow{Q} = m$ and $\overleftarrow{Q} = \ell$: by Example 4, for $L_1 = \mathcal{L}_m$ and $L_2 = \mathcal{L}^{\text{mir}}_{\ell}$, we know that $\mathcal{L}_{\overrightarrow{Q}} \mathcal{L}^{\text{mir}}_{\overleftarrow{Q}}$ extends only the last `Peak` in $\mathcal{L}_m$, so rule 1b applies.
- Consider $\overrightarrow{Q} = \ell$ and $\overleftarrow{Q} = \ell$: by Example 5, for $L_3 = \mathcal{L}_\ell$ and $L_4 = \mathcal{L}^{\text{mir}}_{\ell}$, we know that $\mathcal{L}_\ell \mathcal{L}^{\text{mir}}_{\ell}$ creates `Peak`, so rule 2 applies.
- Consider $\overrightarrow{Q} = m$ and $\overleftarrow{Q} = m$: we have that $\mathcal{L}_m = \Sigma^* \mathcal{L}(\texttt{Peak}) \mathcal{L}(=^*)$ and $\mathcal{L}^{\text{mir}}_m = \mathcal{L}(=^*) \mathcal{L}(\texttt{Peak}) \Sigma^*$; note that there does not exist a non-empty suffix of any word in $\mathcal{L}_m$ that, concatenated with a non-empty prefix of any word in $\mathcal{L}^{\text{mir}}_m$, can form a word in $\mathcal{L}(\texttt{Peak})$, so rule 3 applies.

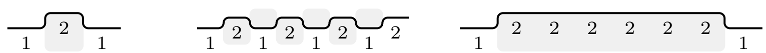
The other six pairs $\langle \overrightarrow{Q}, \overleftarrow{Q} \rangle$ of states are handled similarly. All nine glue expressions are presented in matrix form in Table 2. □

We derived glue constraints for the covered 19 regular expressions: they can be shown to be correct. We establish their propagation impact in Section 5.

In the next section, in order to exploit glue constraints better, we provide bounds on their main variables, namely the results of aggregating feature values on a time series, on a prefix thereof, and on the corresponding suffix thereof.

	k	ℓ	m
k	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_g(\overrightarrow{C},\overleftarrow{C})$
ℓ	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_f(\overrightarrow{D},\overleftarrow{D},\delta_f^i)$	$\phi_f(\overleftarrow{C},\overrightarrow{D},\overleftarrow{D},\delta_f^i)$
m	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_f(\overrightarrow{C},\overrightarrow{D},\overleftarrow{D},\delta_f^i)$	$\phi_g(\overrightarrow{C},\overleftarrow{C})$

Table 2: Glue expressions for any $g_f_$PEAK constraint. A row index refers to the state of the automaton $\mathcal{A}$ in Figure 2 reached for the prefix, and a column index refers to the state of $\mathcal{A}$ reached for the mirror of the corresponding suffix.



(a) (b) (c)

Fig. 3: (a): A time series with an inflexion of shortest width, namely one. (b): A time series with six inflexions. (c): A time series with one inflexion.

4 Bounds for Time-Series Constraints

We derive bounds on N for any time-series constraint $g\_f\_\sigma(\langle X_1,\dots,X_n\rangle, N)$ from a few general formulae and the structure of *ground* time series that give extreme values of N. The bounds are valid regardless of the domain choice, but their sharpness is guaranteed only if all the X_i are over the *same* interval domain $[a,b]$. A bound is *sharp* if it equals N for at least one ground time series.

For each regular expression, there exists a necessary condition, based on the domains and number of the X_i, for it to occur at least once within the signature.

Example 7. An `Inflexion`-pattern, called an *inflexion*, within a time series $X = \langle X_1,\dots,X_n\rangle$ corresponds, except for its first and last elements, to a maximal occurrence of the regular expression `Inflexion` = '(<(<|=)*>)|(>(>|=)*<)' in the signature of X. The necessary condition for having at least one inflexion in X is $b > a \wedge n \geq 3$, where $[a,b]$ is the smallest interval containing the union of the domains of the X_i. Figure 3a gives an example of inflexion. □

In Section 4.1, we describe a systematic methodology for deriving sharp bounds on N for any time-series constraint $g\_f\_\sigma(\langle X_1,\dots,X_n\rangle, N)$, under the assumptions that all the X_i have the same interval domain and, without loss of generality, that the underlying necessary condition holds. In Section 4.2, we illustrate the methodology on one family of constraints.

4.1 Methodology

For any time-series constraint $g\_f\_\sigma(\langle X_1,\dots,X_n\rangle, N)$, our aim is to derive formulae for lower and upper bounds on N, parametrised only by n and the domain bounds of the X_i. We define a time-series structure that depends only on g and f, in order to build an *optimal time series* for the upper (resp. lower) bound, defined as a ground time series where N is equal to that upper (resp. lower) bound. We use the following non-mutually-exclusive properties, which were derived manually, all occurrences of 'maximal' and 'minimal' being over all time series of length n over $[a, b]$:

- Property I holds if the number of σ-patterns is maximal.
- Property II^{up}(resp. II^{low}) holds if there is at least one σ-pattern whose length is maximal (resp. minimal).
- Property $\text{III}^{\text{up}}_{\text{max}}$ (resp. $\text{III}^{\text{low}}_{\text{max}}$) holds if there is at least one σ-pattern and the absolute difference between b (resp. a) and its maximum is minimal.
- Property $\text{III}^{\text{up}}_{\text{min}}$ (resp. $\text{III}^{\text{low}}_{\text{min}}$) holds if there is at least one σ-pattern and the absolute difference between b (resp. a) and its minimum is minimal.
- Property IV holds if there is no σ-pattern.
- Property $\text{V}^{\text{up}}_{\text{max}}$ (resp. $\text{V}^{\text{low}}_{\text{max}}$) holds if the time series is among those where the sum of the absolute differences between b (resp. a) and the maxima of the σ-patterns is minimal, and the number of σ-patterns is maximal.
- Property $\text{V}^{\text{up}}_{\text{min}}$ (resp. $\text{V}^{\text{low}}_{\text{min}}$) holds if the time series is among those where the sum of the absolute differences between b (resp. a) and the minima of the σ-patterns is minimal, and furthermore the number of σ-patterns is maximal.
- Property $\text{VI}^{\text{up}}_{\text{max}}$ (resp. $\text{VI}^{\text{low}}_{\text{max}}$) holds if the time series is among those where the number of σ-patterns is maximal, and the sum of the absolute differences between b (resp. a) and the maxima of the σ-patterns is minimal.
- Property $\text{VI}^{\text{up}}_{\text{min}}$ (resp. $\text{VI}^{\text{low}}_{\text{min}}$) holds if the time series is among those where the number of σ-patterns is maximal, and furthermore the sum of the absolute differences between b (resp. a) and the minima of the σ-patterns is minimal.
- Property VII^{up} (resp. VII^{low}) holds if there is at least one σ-pattern of maximal length among those with only non-negative (resp. non-positive) elements and the sum of the absolute differences between b (resp. a) and all elements of such a σ-pattern is minimal.
- Property VIII^{up} (resp. VIII^{low}) holds if there is at least one σ-pattern of minimal length and the sum of the absolute differences between b (resp. a) and all elements of such a σ-pattern is minimal.

Twelve constraints have a more involved optimal time-series structure that is not described in this paper for space reasons. The formulae for these twelve constraints take time linear in n to evaluate, whereas the formulae for the constraints covered by the given methodology take constant time to evaluate.

Table 3 gives for each feature/aggregator pair the set of properties of optimal time series. An *optimal time series for a property P* is a ground time series for $g\_f\_\sigma(\langle X_1, \ldots, X_n\rangle, N)$ where N takes the largest (resp. smallest) value for all ground time series possessing P. If there are several properties for an $\langle f, g\rangle$ pair, then we first need to identify an optimal time series for each of those properties. An optimal time series for some property is an optimal time series if it has the maximal (resp. minimal) value of N among the set of optimal time series for every property for $\langle f, g\rangle$.

bound	$g\backslash f$	`max`	`min`	`one`	`surface`	`width`
upper	`Max`	$\text{III}^{\text{up}}_{\text{max}}$	$\text{III}^{\text{up}}_{\text{min}}$		VII^{up};VIII^{up}	II^{up}
lower	`Min`	$\text{III}^{\text{low}}_{\text{max}}$	$\text{III}^{\text{low}}_{\text{min}}$		VII^{low};VIII^{low}	II^{low}
upper	`Sum`	IV;$\text{V}^{\text{up}}_{\text{max}}$;$\text{VI}^{\text{up}}_{\text{max}}$	IV;$\text{V}^{\text{up}}_{\text{min}}$;$\text{VI}^{\text{up}}_{\text{min}}$	I	IV;VII^{up};VIII^{up}	I;II^{up}
lower	`Sum`	IV;$\text{V}^{\text{low}}_{\text{max}}$;$\text{VI}^{\text{low}}_{\text{max}}$	IV;$\text{V}^{\text{low}}_{\text{min}}$;$\text{VI}^{\text{low}}_{\text{min}}$		IV;VII^{low};VIII^{low}	

Table 3: Properties of optimal time series, for feature f and aggregator g.

Example 8. Consider $n = 8$ time-series variables over the integer interval $[1, 2]$.

- Consider SUM_ONE_INFLEXION($\langle X_1, \dots, X_8\rangle, N$), which constrains N to be the number of inflexions in $\langle X_1, \dots, X_8\rangle$. For an upper bound on N, the time series in Figure 3b is optimal, with $N = 6$ inflexions, and has Property I.
- Consider MAX_MIN_INFLEXION($\langle X_1, \dots, X_8\rangle, N$), which constrains N to be the maximum of the minima of all inflexions in $\langle X_1, \dots, X_8\rangle$. For an upper bound on N, the time series in Figures 3b and 3c are optimal, both with $N = 2$, and have Property $\text{III}^{\text{up}}_{\text{min}}$ as both have inflexions whose minima have an absolute difference with $b = 2$ that is 0, hence their minima are b.
- Consider MAX_SURFACE_INFLEXION($\langle X_1, \dots, X_8\rangle, N$), which constrains N to be the maximum of the sums of the elements of all inflexions in $\langle X_1, \dots, X_8\rangle$. By Table 3, for an upper bound on N, there exists an optimal time series for Property VII^{up} or Property VIII^{up} or both. The time series in Figure 3c is optimal for Property VII^{up}, with $N = 12$: there is an inflexion of maximal length, namely 6, among those with only non-negative elements, and all elements of this inflexion have an absolute difference with $b = 2$ that is minimal, namely 0. The time series in Figure 3b is optimal for Property VIII^{up}, with $N = 2$: there is an inflexion of minimal length, namely 1, whose elements all have an absolute difference with $b = 2$ that is minimal, namely 0. Hence the upper bound is the maximum of these two values of N, that is 12. □

4.2 Bounds for Constraints that Only Have Property $\text{III}^{\text{up}}_{\text{min}}$

We consider constraints that only have Property $\text{III}^{\text{up}}_{\text{min}}$, that is with $g =$ `Max` and $f =$ `min` according to Table 3. This pair of feature/aggregator makes sense for 18 of the 20 regular expressions in [5]. Our goal is to derive an upper bound on the maximum of the minima of all σ-patterns in a time series, where σ is any of those regular expressions. According to Property $\text{III}^{\text{up}}_{\text{min}}$, in an optimal time series, there is at least one σ-pattern whose minimum is maximal: we use such a σ-pattern to derive this upper bound. For brevity, we do not derive a lower bound, because it is almost always possible to have no σ-patterns at all and the lower bound is then equal to the identity value of g, namely $-\infty$ by Table 1.

Example 9. We can explain the key ideas using Figure 3b. Consider `Inflexion` = '(<(<|=)*>)|(>(>|=)*<)' and time series over an integer interval $[a, b]$. Our goal is to maximise the maximum of the minima of all inflexions in the time series:

in other words, the difference between b and the minimum of some inflexion should be minimal. The time series $t = \langle 1,2,1,2,1,2,1,2\rangle$ in Figure 3b contains two types of inflexions: the first (resp. second) type corresponds to the signature ‘$<>$’ (resp. ‘$><$’); the inflexions are highlighted in grey. Assume the domain is $[-1,+2]$: the minima of the three ‘$<>$’-type inflexions equal the domain upper bound, namely $b = 2$, hence the difference with b is 0; the minima of the three ‘$><$’-type inflexions equal 1, that is $b-1$, hence the difference with b is 1. Hence the smallest difference between b and the minima of the inflexions of t equals 0. Regardless of the value of b, we can always construct a time series with some inflexion that contains b, provided the necessary condition of Example 7 holds. If we now consider the domain $[-1,+5]$, then every element of t can be increased by three, giving $t' = \langle 4,5,4,5,4,5,4,5\rangle$, which has the *same* signature as t. As for t, the minima of all ‘$<>$’-type inflexions equal the domain upper bound, namely $b = 5$, and the minima of all ‘$><$’-type inflexions equal 4, that is $b-1$. Hence the smallest difference between b and the minima of the inflexions of t' also equals 0. We have shown that the smallest difference between b and the minimum of every inflexion does not depend on b, due to the signature being ground. We need to compute the minimum, denoted by $\Delta_{\texttt{Inflexion}}$, of these smallest differences for *any* signature in $\mathcal{L}(\texttt{Inflexion})$. The sharp upper bound on N for the constraint MAX_MIN_INFLEXION($\langle X_1,\ldots,X_n\rangle, N$) equals $b - \Delta_{\texttt{Inflexion}}$. □

We now formalise these ideas.

Computing the Bounds. Consider a MAX_MIN_σ($\langle X_1,\ldots,X_n\rangle, N$) time-series constraint where all the X_i are over the *same* interval domain $[a,b]$. Without loss of generality, for determining an upper bound on N, it suffices to restrict our focus on time series containing just *one* σ-pattern, because the result of a `Max`-aggregation is *any* of its occurrences of the largest value, whereas smaller values are absorbed. Let T_ω denote the set of ground time series over $[a,b]$ whose signature is $\omega \in \mathcal{L}(\sigma)$. For any t in T_ω, let $t_{\downarrow\omega}$ denote the index set of the σ-pattern in t. We want to derive a formula that can be used to evaluate in *constant* time the upper bound $u = \max_{\omega\in\mathcal{L}(\sigma)} \max_{t\in T_\omega} \min_{i\in t_{\downarrow\omega}} t_i$, which is equal to the wanted upper bound on N under the stated focus restriction. Since u depends also on a and b, its direct computation would not take constant time, because every $|T_\omega|$ depends on a and b. In order to compute u in constant time, we reformulate it as an arithmetic expression on b and a parameter that *only* depends on σ, using the following transformations:

$$\begin{aligned} u = b - (b-u) &= b - (b - \max_{\omega\in\mathcal{L}(\sigma)} \max_{t\in T_\omega} \min_{i\in t_{\downarrow\omega}} t_i) \\ &= b - \min_{\omega\in\mathcal{L}(\sigma)} (b - \max_{t\in T_\omega} \min_{i\in t_{\downarrow\omega}} t_i) \\ &= b - \min_{\omega\in\mathcal{L}(\sigma)} \min_{t\in T_\omega} (b - \min_{i\in t_{\downarrow\omega}} t_i) \end{aligned} \tag{1}$$

The value of $\Delta_\omega = \min_{t\in T_\omega}(b - \min_{i\in t_{\downarrow\omega}} t_i)$, called the *shift of signature* ω, does not depend on a and b: every time series t in T_ω that gives this minimum *must* contain b, which can thus be replaced by $\max t$; otherwise, every element of t could be incremented by at least 1, as shown in Example 9, thus reducing the minimal value of $b - \min_{i \in t_{\downarrow\omega}} t_i$ and contradicting the optimal choice of t. Hence $\Delta_\sigma = \min_{\omega \in \mathcal{L}(\sigma)} \Delta_\omega$, called the *shift of regular expression* σ, does not depend on a and b either. The upper bound u on N then is $b - \Delta_\sigma$ by (1). In order to compute Δ_σ, we need to compute Δ_ω for each signature $\omega \in \mathcal{L}(\sigma)$.

We compute each Δ_ω as follows, for a ground signature $\omega = \langle S_1, \ldots, S_\ell \rangle$ linked to a time series $X = \langle X_1, \ldots, X_{\ell+1} \rangle$ by signature constraints. First, we rewrite Δ_ω as follows:

$$\Delta_\omega = \min_{t \in T_\omega} (b - \min_{i \in t_{\downarrow\omega}} t_i) = \min_{t \in T_\omega} \max_{i \in t_{\downarrow\omega}} (b - t_i) \tag{2}$$

Let Δ_i^t denote $b - t_i$. Note that $\Delta_i^t \geq 0$ because we assume $t_i \leq b$. Hence a time series that minimises the sum of the Δ_i^t also minimises each Δ_i^t, and thus the maximum of the Δ_i^t. So a tuple $\langle \Delta_1^t, \ldots, \Delta_{\ell+1}^t \rangle$ that is minimal for the sum $\sum_{i \in [1,\ell+1]} \Delta_i^t$ is also minimal for $\max_{i \in t_{\downarrow\omega}} \Delta_i^t$ and we can solve the following minimisation problem:

$$\begin{array}{llll}
\text{minimise} & \displaystyle\sum_{i=1}^{\ell+1} \Delta_i & & \\
\text{subject to} & \Delta_i \geq 0 & \forall i \in [1, \ell+1] & (3)\\
\text{if } S_i = \text{`}<\text{' then} & \Delta_i > \Delta_{i+1} & \forall i \in [1, \ell] & (4)\\
\text{if } S_i = \text{`}=\text{' then} & \Delta_i = \Delta_{i+1} & \forall i \in [1, \ell] & (5)\\
\text{if } S_i = \text{`}>\text{' then} & \Delta_i < \Delta_{i+1} & \forall i \in [1, \ell] & (6)\\
 & \Delta_i \in \mathbb{Z} & \forall i \in [1, \ell+1] &
\end{array}$$

The semantics of variable Δ_i, called the *shift of variable* X_i, with $i \in [1, \ell+1]$, is $b - X_i$. For example, if $S_i = $ '$>$', meaning $X_i > X_{i+1}$, then $b - X_i < b - X_{i+1}$, hence $\Delta_i < \Delta_{i+1}$. Depending on the value of each S_i, which is assumed ground, we post only one of the constraints (4), (5), or (6) for each pair $\langle \Delta_i, \Delta_{i+1} \rangle$. Note that Δ_i^t corresponds to $\Delta_i = b - X_i$ when $X_i = t_i$: hence constraint (3). Therefore, in an optimal solution $\Delta^* = \langle \Delta_1^*, \ldots, \Delta_{\ell+1}^* \rangle$, the value of Δ_i^* is the minimal shift of X_i. Hence Δ^* is also an optimal solution to the right-hand side of (2), and so we have $\Delta_\omega = \max_{i \in X_{\downarrow\omega}} \Delta_i^*$. Note that the optimal value of the optimisation problem itself is irrelevant.

Since Δ_ω does not depend on a and b, it can be computed once and for all for any signature ω. Hence it does not matter how much time the minimisation problems take to solve. We show further that the number of minimisation problems and their numbers of variables and constraints can be bounded by very small constants.

Example 10. Consider `Inflexion` = '(<(<|=)*>)|(>(>|=)*<)' and the signature $\omega = \langle S_1, S_2 \rangle = \langle$'$<$', '$>$'$\rangle \in \mathcal{L}($`Inflexion`$)$, linked to the time series

$\langle X_1, X_2, X_3\rangle$. We solve the following minimisation problem to compute Δ_ω:

$$\begin{array}{lll} \text{minimise} & \Delta_1 + \Delta_2 + \Delta_3 & \\ \text{subject to} & \Delta_i \geq 0 & \forall i \in [1,3] \\ & \Delta_1 > \Delta_2 & \\ & \Delta_2 < \Delta_3 & \\ & \Delta_i \in \mathbb{Z} & \forall i \in [1,3] \end{array}$$

The unique optimal solution is $\langle \Delta_1^*, \Delta_2^*, \Delta_3^*\rangle = \langle 1, 0, 1\rangle$. The inflexion that corresponds to $\langle S_1, S_2\rangle$ is $\langle X_2\rangle$, as exemplified in Figure 3a, thus $\Delta_\omega = \max_{i\in\{2\}} \Delta_i^* = \Delta_2^* = 0$: this inflexion contains a single element, which can be made to coincide with the domain upper bound. Figure 3a gives an example of such an inflexion within a time series of three variables with 2 as domain upper bound. □

We now state a condition when the computed upper bound is sharp.

Theorem 1. *Consider a time-series constraint* MAX_MIN_$\sigma(\langle X_1, \dots, X_n\rangle, N)$ *where all the* X_i *are over the same integer interval* $[a, b]$*. If at least one word* ω *in* $\mathcal{L}(\sigma)$ *with* $\Delta_\sigma = \Delta_\omega$ *may occur in the signature of* $\langle X_1, \dots, X_n\rangle$*, then the upper bound* $b - \Delta_\sigma$ *on* N *is sharp.*

Proof. Suppose there exists a word ω that satisfies the stated assumption. Hence there exists a ground time series with an occurrence of ω in its signature: the value of N on such a time series equals $b - \Delta_\omega$, so the bound $b - \Delta_\sigma$ on N is sharp because $\Delta_\sigma = \Delta_\omega$. □

For any regular expression σ in [5] and any time series X over some interval, the assumption of Theorem 1 holds if the necessary condition (such as in Example 7) for having at least one occurrence of σ in the signature of X is met.

Accelerating the Computation of the Shift of a Regular Expression. For some regular expressions, we do not need to minimise over the entire language $\mathcal{L}(\sigma)$ when computing $\Delta_\sigma = \min_{\omega\in\mathcal{L}(\sigma)} \Delta_\omega$. Consider the case when there exists a word ω in $\mathcal{L}(\sigma)^{\min}$, which is the set of the shortest words of $\mathcal{L}(\sigma)$, such that the following equality holds:

$$\Delta_\sigma = \Delta_\omega \tag{7}$$

We can then replace $\mathcal{L}(\sigma)$ with $\mathcal{L}(\sigma)^{\min}$ in the definition of Δ_σ. This is the case for all σ in [5], and, additionally, we have $\left|\mathcal{L}(\sigma)^{\min}\right| \leq 2$. Hence computing Δ_σ requires solving at most two optimisation problems over at most four variables.

Example 11. Since `Inflexion` = ‘(<(<|=)*>)|(>(>|=)*<)’ contains one disjunction at the highest level, every word in $\mathcal{L}$(`Inflexion`) belongs to either $L_1 = \mathcal{L}$(‘<(<|=)*>’) or $L_2 = \mathcal{L}$(‘>(>|=)*<’). Hence $\mathcal{L}(\texttt{Inflexion})^{\min}$ is the union of the two sets $L_1^{\min} = \{$‘<>’$\}$ and $L_2^{\min} = \{$‘><’$\}$. Consider the word ‘<<>’ in L_1 obtained from the word ‘<>’ in $L_1^{\min}$ by inserting just one ‘<’. In order to obtain the minimisation problem for computing $\Delta_{<<>}$, we modify the one of Example 10 for $\Delta_{<>} = 0$ by introducing the new variable Δ_4 and replacing the comparison constraints by the following ones:

$$\Delta_1 > \Delta_2 \ \wedge\ \Delta_2 > \Delta_3 \ \wedge\ \Delta_3 < \Delta_4$$

The unique optimal solution is $\langle 2, 1, 0, 1\rangle$, giving $\Delta_{<<>} = 1 > \Delta_{<>}$. Similarly, for the word ‘<=>’ obtained from ‘<>’ by inserting just one ‘=’, we have $\Delta_{<=>} = \Delta_{<>}$. Using these base cases, one can prove by induction that the shift of any word in L_1 longer than ‘<>’ is at least $\Delta_{<>}$. Applying the same reasoning for the language L_2, we obtain $\Delta_\omega \geq \Delta_{><} = 1$ for all words ω in L_2. Hence $\Delta_{\mathtt{Inflexion}} = \min(\Delta_{<>}, \Delta_{><}) = \min(0, 1) = 0$ and equality (7) holds, so we can replace $\mathcal{L}(\mathtt{Inflexion})$ by $\mathcal{L}(\mathtt{Inflexion})^{\min}$ in the definition of Δ_σ. □

5 Evaluation

We evaluate the impact of the methods introduced in the previous sections on both execution time and the number of backtracks (failures) for all the 200 time-series constraints for which the glue constraint exists.

In our first experiment, we consider a single $g\_f\_\sigma(\langle X_1, X_2, \ldots, X_n\rangle, N)$ constraint for which we first enumerate N and then either find solutions by assigning the X_i or prove infeasibility of the chosen N. For each constraint, we compare four variants of *Automaton*, which just states the constraint, using the automaton of [3]: *Glue* adds to *Automaton* the glue constraints of Section 3 for all prefixes and corresponding reversed suffixes, which can be done [6] by just posing *one* additional constraint, namely $g\_f\_\sigma^{\mathrm{mir}}(\langle X_n, \ldots, X_2, X_1\rangle, N)$; *Bounds* adds to *Automaton* the bound restrictions of Section 4; *Bounds+Glue* uses both the glue constraints and the bounds; and *Combined* adds to *Bounds+Glue* the bounds for each prefix and corresponding reversed suffix.

In Figure 4, we show results for two problems that are small enough to perform all computations for *Automaton* and all variants within a reasonable time. In the first problem (first row of plots), we use time series of length 10 over the domain $[1, 5]$, and find, for each value of N, the first solution or prove infeasibility. This would be typical for satisfaction or optimisation problems, where one has to detect infeasibility quickly. Our static search routine enumerates the time-series variables X_i from left to right, starting with the smallest value in the domain. In the case of the initial domains being of the same size, this heuristic typically works best. In the second problem (second row of plots), we consider time series of length 8 over the domain $[1, 5]$, and find all solutions for each value of N. This allows us to verify that no solutions are incorrectly eliminated by any of the variants, and provides a worst-case scenario exploring the complete search tree. Results for the backtrack count are on the left, results for the execution time on the right. We use log scales on both axes, replacing a zero value by one in order to allow plotting. All experiments were run with SICStus Prolog 4.2.3 on a 2011 MacBook Pro 2.2 GHz quadcore Intel Core i7-950 machine with 6 MB cache and 16 GB memory using a single core.

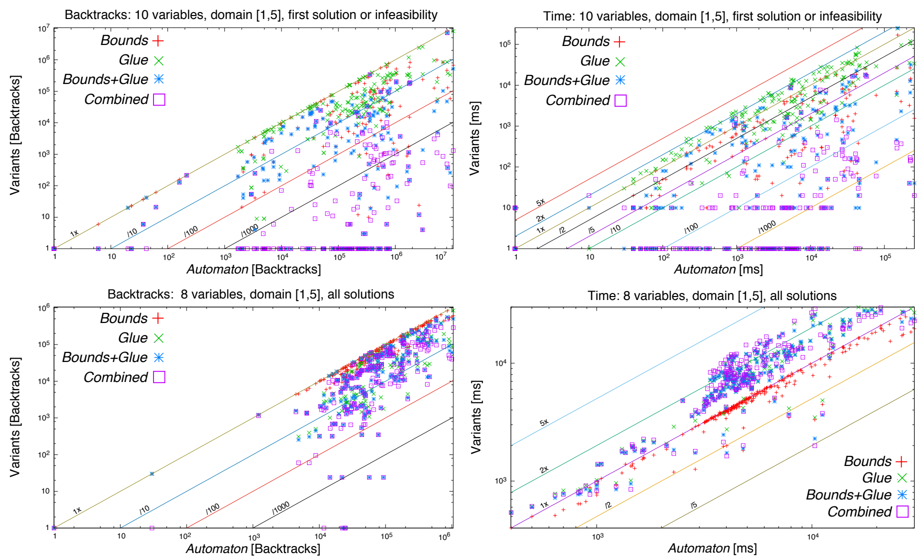


Fig. 4: Comparing backtrack count and runtime for *Automaton* and its variants for the first solution (length 10) and all solutions (length 8).

We see that *Bounds* and *Glue* on their own bring good reductions of the search space, but their combinations *Bounds*+*Glue* and *Combined* in many cases reduce the number of backtracks by more than three orders of magnitude. Indeed, for many constraints, finding the first solution requires no backtracks. On the other hand, there are a few constraints for which the number of backtracks is not reduced significantly. These are constraints for which values of N in the middle of the domain are infeasible, but this is not detected by any of our variants.

The time for finding the first solution or proving infeasibility is also significantly reduced by the combinations *Bounds*+*Glue* and *Combined*, even though the glue constraints require two time-series constraints. When finding all solutions, this overhead shows in the total time taken for the three variants using the glue constraints. The bounds on their own reduce the time for many constraints, but rarely by more than a factor of ten.

In our second experiment, shown in Figure 5, we want to see whether the *Combined* variant is scalable. For this, we increase the length of the time series from 10 to 120 over the domain $[1, 5]$. We enumerate all possible values of N and find a first solution or prove infeasibility. For each time-series constraint and value of N, we impose a timeout of 20 seconds, and we do not consider the constraint if there is a timeout on some value of N. We plot the percentage of all constraints for which the average runtime is less than or equal to the value on the horizontal axis. For small time values, there are some quantisation effects due to the SICStus time resolution of 10 milliseconds.

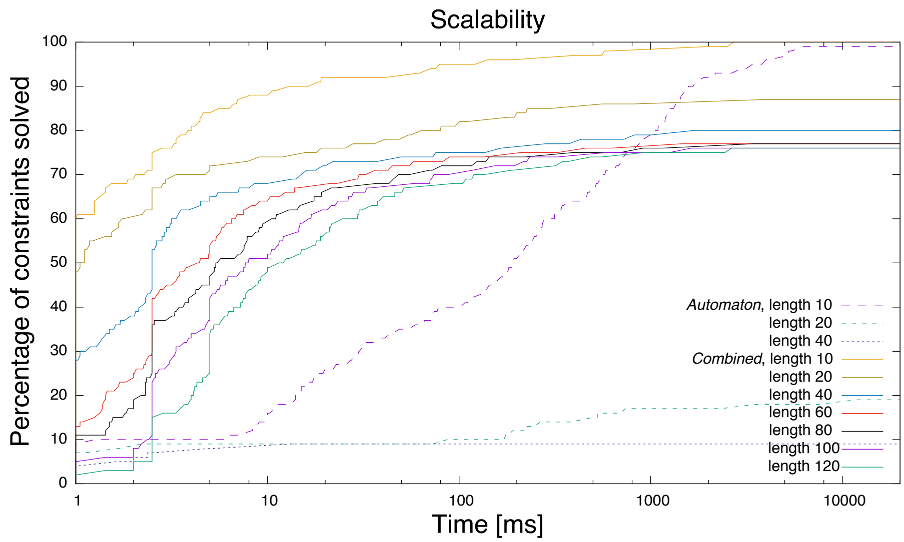


Fig. 5: Scalability results comparing time for *Automaton* and *Combined* on problems of increasing length.

For length 10, we find solutions for all values of N within the timeout, and our plots for *Automaton* (dashed) and *Combined* (solid) reach 100%, but the average time of *Combined* is much smaller. For *Automaton*, the percentage of constraints that are solved within the timeout drops to less than 20% for length 20, and less than 10% for length 40. For *Combined*, we solve over 75% of all constraints within the time limit, even for lengths 100 and 120.

The constraints that are not solved by *Combined* use the feature `surface` or the aggregator `Sum`. The worst performance is observed for constraints combining both `surface` and `Sum`. This is not surprising, as we know that achieving domain consistency for many of those constraints is NP-hard (encoding of *subset-sum*).

6 Conclusion

For the time-series constraints in [5], specified by a triple $\langle \sigma, f, g \rangle$, we showed in [3] how to generate simplified automata and linear implied constraints. Here, we further enhance the propagation of time-series constraints by a systematic generation of bounds and glue constraints. Rather than finding bounds and glue constraints for each time-series constraint independently, we introduce the concepts of *parametric bounds* and *parametric glue constraints*. Our approach differs from existing ones, which design dedicated propagation algorithms [14,4] and reformulations [9,10] for specific constraints, or propose generic approaches [15,13] that do not focus on the combinatorial aspect of a constraint.

References

1. Almeida, M., Moreira, N., Reis, R.: Enumeration and generation with a string automata representation. Theoretical Computer Science 387(2), 93–102 (2007), the *FAdo* tool is available at `http://fado.dcc.fc.up.pt/`
2. Arafailova, E., Beldiceanu, N., Carlsson, M., Douence, R., Flener, P., Francisco Rodríguez, M.A., Pearson, J., Simonis, H.: Global Constraint Catalogue, Volume II: Time-Series Constraints. Tech. rep., Computing Research Repository (forthcoming), `http://arxiv.org`
3. Arafailova, E., Beldiceanu, N., Douence, R., Flener, P., Francisco Rodríguez, M.A., Pearson, J., Simonis, H.: Time-series constraints: Improvements and application in CP and MIP contexts. In: Quimper, C.G. (ed.) CP-AI-OR 2016. LNCS, vol. 9676, pp. 18–34. Springer (2016)
4. Beldiceanu, N., Carlsson, M.: Sweep as a generic pruning technique applied to the non-overlapping rectangles constraints. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 377–391. Springer (2001)
5. Beldiceanu, N., Carlsson, M., Douence, R., Simonis, H.: Using finite transducers for describing and synthesising structural time-series constraints. Constraints 21(1), 22–40 (January 2016), journal fast track of CP 2015: summary on p. 723 of LNCS 9255, Springer, 2015
6. Beldiceanu, N., Carlsson, M., Flener, P., Francisco Rodríguez, M.A., Pearson, J.: Linking prefixes and suffixes for constraints encoded using automata with accumulators. In: O'Sullivan, B. (ed.) CP 2014. LNCS, vol. 8656, pp. 142–157. Springer (2014)
7. Beldiceanu, N., Carlsson, M., Petit, T.: Deriving filtering algorithms from constraint checkers. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 107–122. Springer (2004)
8. Beldiceanu, N., Ifrim, G., Lenoir, A., Simonis, H.: Describing and generating solutions for the EDF unit commitment problem with the ModelSeeker. In: Schulte, C. (ed.) CP 2013. LNCS, vol. 8124, pp. 733–748. Springer (2013)
9. Bessière, C., Hebrard, E., Hnich, B., Kiziltan, Z., Quimper, C.G., Walsh, T.: Reformulating global constraints: The *slide* and *regular* constraints. In: SARA 2007. LNAI, vol. 4612, pp. 80–92. Springer (2007)
10. Bessière, C., Katsirelos, G., Narodytska, N., Quimper, C.G., Walsh, T.: Decomposition of the NValue constraint. In: Cohen, D. (ed.) CP 2010. LNCS, vol. 6308, pp. 114–128. Springer (2010)
11. Francisco Rodríguez, M.A., Flener, P., Pearson, J.: Implied constraints for Automaton constraints. In: Gottlob, G., Sutcliffe, G., Voronkov, A. (eds.) GCAI 2015. EasyChair Proceedings in Computing, vol. 36, pp. 113–126 (2015)
12. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, 3rd edn. (2007)
13. Monette, J.N., Flener, P., Pearson, J.: Towards solver-independent propagators. In: Milano, M. (ed.) CP 2012. LNCS, vol. 7514, pp. 544–560. Springer (2012)
14. Régin, J.C.: A filtering algorithm for constraints of difference in CSPs. In: Hayes-Roth, B., Korf, R.E. (eds.) AAAI 1994. pp. 362–367. AAAI Press (1994)
15. Van Hentenryck, P., Saraswat, V., Deville, Y.: Design, implementation, and evaluation of the constraint language cc(FD). Tech. Rep. CS-93-02, Brown University, Providence, USA (January 1993), revised version in *Journal of Logic Programming* 37(1–3):293–316, 1998. Based on the unpublished manuscript *Constraint Processing in cc(FD)*, 1991.