Quantifier elimination over finite strings with monotone functions

Definitions: A *string* is a finite structure over the domain $\{1, \ldots, n\}$ with constants $n \geq 1$, totally ordered by a binary relation $<$, together with a fixed number of monadic predicates. The initial signature will be augmented by an infinite family of additional monotone (increasing) functions, i.e. those satisfying $x_1 < x_2 \Rightarrow f(x_1) \leq f(x_2)$.

Some are defined by using minimum and maximum operators applied to the parameterized set $\varphi(y) = \{x : \varphi(x, y)\}$ defined by a first-order formula φ, where whenever we take the minimum or maximum of the empty set, we return the maximum or minimum element of the domain respectively, i.e. $\min \{\} = n$ and $\max \{\} = 1$. These both have first-order definitions.

$$\min \{x : \varphi(x, y)\} = z \quad \Leftrightarrow \quad \forall x\, [\varphi(x, y) \rightarrow x \geq z] \wedge \exists x\, \varphi(x, y) \rightarrow \varphi(z, y) \wedge \nexists x \varphi(x, y) \rightarrow z = n$$
$$\max \{x : \varphi(x, y)\} = z \quad \Leftrightarrow \quad \forall x\, [\varphi(x, y) \rightarrow x \leq z] \wedge \exists x\, \varphi(x, y) \rightarrow \varphi(z, y) \wedge \nexists x \varphi(x, y) \rightarrow z = 1$$

E.g. from these, it is easy to define the successor and predecessor functions s and p respectively, by:

$s(y) = \min \{x : x > y\}$ That is, $s(y) = y + 1$ for $y < n$ (n otherwise).
$p(y) = \max \{x : x < y\}$ That is, $p(y) = y - 1$ for $y > 1$ (1 otherwise).

Notice that both are monotone, because successor tops out at n and predecessor bottoms out at 1.

Other monotone increasing functions will be derived by taking any function F and applying:

$$F_{max}(y) = \max \{x : F(x) \leq y\}$$
$$F_{min}(y) = \min \{x : F(x) \geq y\}$$

E.g. $s_{min}(y) = p(y)$ and $p_{max}(y) = s(y)$, $s_{max}(y) = p(y)$ for $y < n$ (n o.t.) and $p_{min}(y) = s(y)$ for $y > 1$ (1 o.t.).

Another application is the Skolem function returning the nearest element satisfying a formula $\varphi(x)$. Let γ_φ and λ_φ be the monotone (and idempotent) upward and downward Skolem functions that return respectively the greatest element $\leq y$ (1 otherwise) or least element $\geq y$ (n otherwise) satisfying φ. I.e.

$\gamma_\varphi = F_{max}$ (except for n) where $F(x) = x$ if $\varphi(x)$ (n otherwise)
$\lambda_\varphi = F_{min}$ (except for 1) where $F(x) = x$ if $\varphi(x)$ (1 otherwise)

We will see that the incorrect values for $\gamma_\varphi(n)$ or $\lambda_\varphi(1)$ are never used in our construction.

Theorem: Every first-order formula over strings is equivalent to a Boolean combination of quantifier-free formulas in a vocabulary expanded by linear-time computable function symbols.

Proof: Proceed by a quantifier elimination induction on the structure of formulas, where the only nontrivial case is application of an existential quantifier $\exists x\, \theta(x, \bar{y})$ to a formula in which negations have been pushed all the way down, replacing instances of $\nless$, $\ngtr$, and $\neq$ by the corresponding disjunctions over $<$, $>$, and $=$. The key observation is that every term involving x (or any other variable) must be of the form $F(x)$ where F is monotone, because all the functions are monadic and monotonicity is preserved under composition. In this case, F_{max} and F_{min} serve as quasi-inverses, allowing us to solve for x in any equality or inequality involving $F(x)$ by moving F to the other side, denoted by any term t (not containing x):

$$F(x) \leq t \quad \Leftrightarrow \quad x \leq F_{max}(t) \wedge F(1) \leq t$$
$$F(x) \geq t \quad \Leftrightarrow \quad x \geq F_{min}(t) \wedge F(n) \geq t$$

Use the transformations: $F(x) < t \Leftrightarrow F(x) \ngeq t$; $F(x) > t \Leftrightarrow F(x) \nleq t$; and $F(x) = t \Leftrightarrow t \leq F(x) \leq t$.

Next, put θ in disjunctive normal form and distribute the existential over disjunctions, factoring out any atomic formulas that don't involve x from within the scope of the quantifier to obtain $\exists y\ \psi(x, \bar{y})$ where ψ is a conjunction of atomic formulas, each containing x. Collect all atomic formulas which depend *only* on x (including any equations which involve x on both sides), and call their conjunction $\varphi(x)$. If there is an equality remaining it must be of the form $x = \tau(y)$ for y in $\bar{y}$, and consequently $\exists x\ \psi$ is equivalent to $\psi[x \leftarrow \tau(y)]$ in which all occurrences of x have been replaced by the term $\tau(y)$. Otherwise, all remaining formulas are inequalities of the form $x > \alpha_i(y)$ for $1 \leq i \leq k$ and $x < \beta_j(y)$ for $1 \leq j \leq l$ where each of the terms α_i and β_j involve some variable y among $\bar{y}$ (not necessarily the same one). Rewrite these as $\alpha_1, \ldots, \alpha_k < x < \beta_1, \ldots, \beta_l$ in order to see that they are equivalent to the disjunction over all i and j of the conjunction over all i' and j' of $\alpha_{i'} \leq \alpha_i < x < \beta_j \leq \beta_{j'}$. So x is sandwiched between two terms $\alpha_i = \alpha < x < \beta = \beta_j$ (let $\alpha = 1$ if $k = 0$ and $\beta = n$ if $l = 0$). To express $\exists x\ \psi$ we simply assert $\alpha < \gamma_\varphi(p(\beta))$ or equivalently $\lambda_\varphi(s(\alpha)) < \beta$ (this is where we need successor and predecessor, and note that $\gamma_\varphi(n)$ or $\lambda_\varphi(1)$ are never used).

To see each newly defined monotone function is computable in linear-time, it suffices to show by induction that F_{max} and F_{min} can be computed in linear-time, since clearly the values for predecessor and successor can be tabulated outright in linear-time. By induction hypothesis assume that a monotone F has been tabulated in linear-time. Starting from the beginning for F_{max} and the end for F_{min} respectively, it is a simple matter to assign all the values for them in a loop:

F_{max}: For $y = 1$, let $F_{max}(1) = \max\{x : F(x) = 1\}$, starting the search from $x = 1$ and ending when $F(x) > 1$. For $y = 2$ upto n compute $F_{max}(y) = \max\{x : F(x) \leq y\}$ starting from $x = F_{max}(y - 1)$ and ending when $F(x) > y$.

F_{min}: For $y = n$, let $F_{min}(n) = \min\{x : F(x) = n\}$, starting the search from $x = n$ and ending when $F(x) < n$. For $y = n - 1$ downto 1 compute $F_{min}(y) = \min\{x : F(x) \geq y\}$ starting from $x = F_{min}(y + 1)$ and ending when $F(x) < y$.

The total number of x values searched over all y values is linear because of monotonicity. [Therefore a minor modification might need to be made in computing the Skolem functions.]

Corollary: After a linear-time preprocessing stage; every first-order formula can be evaluated in constant-time given any particular assignment of its free variables.
Proof: Convert to a quantifier-free formula $\psi(\bar{y})$, and after computing tables for all the functions needed in linear-time, simply plug in values for the free variables and evaluate it in constant-time.

P.S. Extending this to modular counting quantifiers is even easier, since we only need to add monadic truth predicates.

S. Lindell 3/2012