A 0/1 encoding of the GACLex constraint for pairs of vectors*

Ian P. Gent[1], Patrick Prosser[2], and Barbara M. Smith[3]

[1] School of Computer Science, University of St Andrews, Scotland.
ipg@dcs.st-and.ac.uk
[2] Department of Computing Science, University of Glasgow, Scotland.
pat@dcs.gla.ac.uk
[3] School of Computing Science and Mathematics, University of Huddersfield, England. b.m.smith@hud.ac.uk

1 Introduction

Vectors A and B are in lexicographic order ($\leq_{lex}$) if $\sum_{i=1}^{n} A[i]^{n-i+1} \leq \sum_{i=1}^{n} B[i]^{n-i+1}$. That is, we consider vectors A and B as n digit numbers, where $A[1]$ and $B[1]$ are the most significant digits. In many problems we have 0/1 vectors where we wish to maintain lexicographic order, and we might do this in an attempt to break symmetries [7, 8, 3]. Therefore an efficient $\leq_{lex}$ constraint could prove useful.

Consider the following naive implementation of a $\leq_{lex}$ constraint. Assume that we have the two vectors A and B, both of length 4. We might post a constraint as follows:

$$A[1] < B[1] \vee (A[1] = B[1] \wedge (A[2] < B[2] \vee (A[2] = B[2] \wedge (A[3] < B[3] \vee (A[3] = B[3] \wedge (A[4] < B[4] \vee A[4] = B[4])))))$$

If $A[1]$ and $B[1]$ are instantiated and equal, $A[2]$ and $B[2]$ are instantiated and equal, and all values in the domain of $A[4]$ are greater than $B[[4]$ the constraint would force $A[3]$ to be less than $B[3]$. Note that if we wished to enforce $<_{lex}$ we would have removed the constraint $A[4] = B[4]$. The complexity of arc-consistency is $O(ed^r)$ where e is the number of constraints, d is the domain size, and r is the arity of the constraint [6]. Consequently the complexity for the above encoding of $\leq_{lex}$ is $O(d^n)$. We now present an encoding that is $O(n)$ when A and B are both 0/1 vectors of length n. We show how this encoding propagates, and we then prove that the encoding is correct.

2 The Encoding

We assume that we have two arrays of finite domained variables A and B, such that we want $A \leq_{lex} B$. We assume these are indexed from 1, most significant first. We introduce a new array of 0/1 variables α. We index α from 0 to n. The intended meaning of α is that

* This work was supported by EPSRC research grants GR/M90641, GR/R29666, and GR/R29673.

– if $\alpha[i] = 1$, then $A[j] = B[j]$ for all $1 \leq j \leq i$,
– if $\alpha[i+1] = 0$ but $\alpha[i] = 1$, then then $A[i+1] < B[i+1]$. (From the above case, we still have $A[j] = B[j]$ for $j \leq i$.)

The constraints in the problem are the following. The first is trivial but is included for pedagogical convenience.

$$\alpha[0] = 1 \tag{1}$$
$$(0 \leq i \leq n-1) \;\; \alpha[i] = 0 \implies \alpha[i+1] = 0 \tag{2}$$
$$(1 \leq i \leq n) \;\; \alpha[i] = 1 \implies A[i] = B[i] \tag{3}$$
$$(0 \leq i \leq n-1) \;\; \alpha[i] = 1 \;\&\; \alpha[i+1] = 0 \implies A[i+1] < B[i+1] \tag{4}$$
$$(0 \leq i \leq n-1) \;\; \alpha[i] = 1 \implies A[i+1] \leq B[i+1] \tag{5}$$

Note that the last constraint is implied by constraints (3) and (4), and the observation that either $\alpha[i+1] = 0$ or $\alpha[i+1] = 1$ and in either case $A[i+1] \leq B[i+1]$. However, because of the case analysis, arc consistency is not strong enough to make the deduction, hence the addition of constraint (5).

There are $4n+1$ constraints, involving at most one domain of size size d (since α contains 0/1 variables) and arity no more than 4. To establish arc consistency in this problem therefore takes time $O(nd^4)$. That is, assuming domains of size d are totally ordered, arc-consistency can be achieved for monotonic constraints in time $O(nd)$ [4]. However, we suspect this can be improved by the use of bounds consistency on A and B to $O(n)$. However, when A and B are both 0/1 vectors the complexity falls to $O(n)$.

If we want strict inequality between A and B, i.e. $A <_{lex} B$, we need only add the following constraint:

$$\alpha[n] = 0 \tag{6}$$

3 An Example of Propagation

Before presenting the proof, we provide an example of how propagation proceeds. Consider the following example of domains of A and B, where the underscore indicates that both 0 and 1 are in the domain.

A: 0 1 _ 0 1 ...
B: 0 1 _ 0 0 ...

We now show what happens to α in microscopic detail. Understanding this example should help considerably in understanding the proof to follow.

α: 1 _ _ _ _ _ ...
α: 1 1 _ _ _ _ ... (contrapositive of (4), because A[1]=B[1], but α[0]=1)
α: 1 1 1 _ _ _ ... (contrapositive of (4), because A[2]=B[2], but α[1]=1)

α: 1 1 1 _ _ 0 ... (contrapositive of (3), because A[5] $\neq$ B[5])
α: 1 1 1 _ 0 0 ... (contrapositive of (4), because A[5] $\geq$ B[5], but α[5]=0)
α: 1 1 1 0 0 0 ... (contrapositive of (4), because A[4] $\geq$ B[4], but α[4]=0)

Finally, we get propagation back into A and B.

A: 0 1 0 0 1 ... (direct application of (4), α[2]=1, α[3]=0)
B: 0 1 1 0 0 ... (direct application of (4), α[2]=1, α[3]=0)

A particular point to note is that the propagation which sets values of α is usually by the contrapositive of the constraints given above. Note that we must assume that constraints propagate in all directions, although this will normally be satisfied by a standard AC algorithm.

4 Correctness

We wish to prove that arc consistency is strong enough to establish GAC on the $\leq_{lex}$ constraint. To do this, we require certain properties of the treatment of numeric $<$ and $\leq$. Specifically, when the minimum of the domain of x is greater than (respectively $\geq$) than the maximum of the domain of y, we require that the falsity of $x \leq y$ (respectively $x < y$) is recognised and propagated accordingly. Further any values in the domain of x greater than ($\geq$) the maximum value in the domain of y should be removed if $x \leq y$ ($x < y$), and any values in the domain of y less than ($\leq$) the minimum value in the domain of x. This is a weak requirement, and certainly seems to be the case in ILOG Solver and in Choco [5], the two languages we have experimented with to date.

Theorem 1. *In an arc consistent state of this encoding in a system which respects the condition described above, the constraint $A \leq_{lex} B$ is generalised arc consistent (GAC). That is, any value in any of the domains of the variables of A or B occurs in tuples of values which satisfies the constraint $A \leq_{lex} B$.*

Proof. We first give a sketch of the proof. We are to prove that the constraints (1) to (5) are arc-consistent if and only if the vectors A and B are GAC with respect to the lex ordering constraint $\leq_{lex}$. Since we are to prove the biconditional, the proof is two parts. In the first part we prove the implication: if the constraints (1) to (5) are arc consistent then A and B are GAC with respect to $\leq_{lex}$. In the second part we prove that if A and B are GAC with respect to $\leq_{lex}$ then constraints (1) to (5) are arc consistent.

- **Part 1: If constraints (1) to (5) are arc consistent then vectors A and B are GAC with respect to $\leq_{lex}$**
  We first consider the possible patterns in the values of the vector α when it is arc-consistent with respect to the constraints (1) to (5), and we then prove that for each one of those patterns A and B are GAC with respect to $\leq_{lex}$.
  Consider the 0/1 vector α. An element $\alpha[i]$ can be instantiated, and have a value of 0 or 1, or uninstantiated and have a domain $\{0, 1\}$. Consider the following cases

1. $\alpha[i] = 0$ Consequently $\alpha[i+1] = 0$, for all i, as a result of constraint (2). Therefore constraints (3), (4), and (5) are satisfied, and A[j] is unconstrained with respect to B[j] for $j > i$.
2. $\alpha[i] = 1$ Consequently $\alpha[i-1] = 1$, i.e. all elements to the left of $\alpha[i]$ are also set to 1. This is due to the contrapositive of (2). Therefore due to (3) A[j] is constrained to be equal to B[j] for $1 \leq j \leq i$.
3. $\alpha[i] \in 0, 1$ Assume i is the smallest i with this property. Everything to the left of $\alpha[i]$ must be set to one, i.e. they cannot possibly be set to 0 otherwise $\alpha[i]$ would also have been set to 0 due to constraint (2). Therefore A[j] is constrained to be equal to B[j] for $1 \leq j < i$.

We use $\diamond$ to indicate an uninstantiated variable, i.e. $\alpha[i] \in \{0, 1\}$. We can only have patterns in α of the form $1^+ \diamond^* 0^*$, where X^+ means 1 or more occurrences of X and X^* means 0 or many occurrences. Of these sequences, all must be either 1^+0^* or $1^+\diamond^+0^*$. In both cases we can show that all values of A and B are GAC with respect to $\leq_{lex}$.

$\alpha = 1^+0^+$ This is the simpler case as no values of α are uninstantiated. We can distinguish three sub-cases of the pattern: (a) 10^+, (b) 111^*, and (c) (e) 11^+0^+. In pattern (a) constraint (4) enforces A[1] < B[1], and this guarantees $A <_{lex} B$. In pattern (b) constraint (3) enforces A[i] = B[i] for all i, guaranteeing $A =_{lex} B$. In pattern (c) constraint (2) enforces equality between the first elements of A and B, and at the transition point where $\alpha[i] = 1$ and $\alpha[i+1] = 0$ constraint (4) forces A[i+1] < B[i+1], guaranteeing $A <_{lex} B$.

$\alpha = 1^+\diamond^+0^*$ In this case some values of α are uninstantiated between the initial 1's and final 0's. Let i be the first index such that $\alpha[i]$ is uninstantiated. We first show that all values of $A[i], B[i]$ are GAC-consistent, and then of all later values $A[j], B[j]$. For $j < i$, we must have that $A[j] = B[j]$ from constraint (3). Note that $\alpha[i-1] = 1$, guaranteeing $A[i] \leq B[i]$ from constraint (5). Except possibly for the maximum remaining value, all values of $A[i]$ are thus GAC-consistent, because we simply set $B[i]$ to its maximum value, ensuring $A <_{lex} B$. Similarly all values of $B[i]$ are GAC-consistent except possibly the minimum value. So we now consider the case that v is the largest remaining value in the arc consistent domains of $A[i], B[i]$, but $A[i] = v$ is hypothetically inconsistent with $A \leq_{lex} B$. It would be consistent if we could set $A[i+1] < B[i+1]$. Similarly, we can get consistency through any number of values $A[i+1] = B[i+1], A[i+2] = B[i+2], \ldots$ either extending to the end of the tuples or followed immediately by a value $A[i+k] < B[i+k]$. The only way, therefore, to get inconsistency is to have some sequence of forced equalities followed by an index $i+k$ such that $A[i+k] > B[i+k]$. But the contrapositive of constraint (5) would give $\alpha[i+k] = 0$. The contrapositive of constraint (4) then gives $\alpha[i+k-1] = 0$. Constraint (4) will act in this way repeatedly, through the sequence of equalities following i. Eventually, we will have $A[i+1] = B[i+1]$ and $\alpha[i+1] = 0$. But this will give $\alpha[i] = 0$, contradicting the fact that $\alpha[i]$ is uninstantiated. So we have shown that the

hypothesis that $A[i] = v$ is inconsistent leads to a contradiction. A similar argument establishes that the minimum remaining value of $B[i]$ is GAC-consistent. Finally, note that $A[i] < B[i]$ must remain possible, as otherwise the converse of constraint (4) would set $\alpha[i] = 1$. Whatever values allow this make $A <_{lex} B$, ensuring the GAC-consistency of all values of $A[j]$, $B[j]$ for $j > i$.

- **Part 2: If vectors A and B are GAC with respect to $\leq_{lex}$ then the α variables are arc consistent with respect to the constraints (1) to (5)**
  We now show that there will be values of the array α which make it arc consistent when A and B are GAC with respect to $\leq_{lex}$.
  Set $\alpha[i] = 1$ for $i = 0$ and any initial consecutive sequence of values i such that $A[i] = B[i]$. Set $\alpha[j] = 0$ for the lowest number j such that $A[i] \neq B[j]$, and set $\alpha[k] = 0$ for all $k > j$. Set $\alpha[k] = 0$ for any value of k such that $A[k+1] \geq B[k+1]$ and $\alpha[k+1] = 0$ (and repeat this step if necessary.) Finally, leave any remaining variables in α uninstantiated. Under this definition none of the constraints (1)-(5) are unsatisfied or will set any uninstantiated variable to a value.

QED

5 Conclusion

We have presented an efficient encoding of the $\leq_{lex}$ constraint for 0/1 vectors. However, when domains are larger we expect that the complexity of the encoding will be dependent on the efficiency of the $<$ constraint. The encoding presented here was inspired by an unpublished algorithm proposed by Ian Miguel. His algorithm has a flavour of the Gale Shapley algorithm [1]. Consequently the encoding presented here has much in common with that in [2].

Acknowledgements

We would like to thank Warwick Harvey, Ian Miguel, and Toby Walsh.

References

1. D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, pages 9–15, 1962.
2. Ian P. Gent, Robert W. Irving, David Manlove, Patrick Prosser, and Barbara M. Smith. Constraint programming approach to the stable marriage problem. In *Proceedings of CP '01: the 7th International Conference on Principles and Practice of Constraint Programming. LNCS 2239*, pages 225–240, 2001.
3. Ian P. Gent and Barbara M. Smith. Symmetry breaking during search in constraint programming. In *Proceedings ECAI 2000*, pages 599–603, 2000.
4. P. Van Hentenryck, Y Deville, and C-M Teng. A generic arc-consistency algorithm and its applications. *Artificial Intelligence*, 57(2–3):291–322, 1992.

5. Francoise Laburthe. Choco: a constraint programming kernel for solving combinatori al optimization problems. http://www.choco-constraints.net/.
6. A.K. Mackworth and E.C. Freuder. The complexity of some polynomial consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–74, 1985.
7. P. Meseguer and C. Torras. Exploiting symmetries within constraint satisfaction search. *Artificial Intelligence*, 129:133–163, 2001.
8. Barbara M. Smith. Reducing symmetry in a combinatorial design problem. In *Proceedings of CP-AI-OR'01, 3rd International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 351–359, 2001.