

Complexity Classes and Theories for the Comparator Circuit Value Problem

Stephen A. Cook¹, Dai Tri Man Lê¹, and Yuli Ye¹

¹ Department of Computer Science, University of Toronto

Date: *October 11, 2011*

Abstract

Subramanian defined the complexity class CC as the set of problems log-space reducible to the comparator circuit value problem. He proved that several other problems are complete for CC , including the stable marriage problem, and finding the lexicographical first maximal matching in a bipartite graph. We suggest alternative definitions of CC based on different reducibilities and introduce a two-sorted theory VCC^* based on one of them. We sharpen and simplify Subramanian's completeness proofs for the above two problems and show how to formalize them in VCC^* .

Contents

1	Introduction	2
2	Preliminaries	3
2.1	Two-sorted vocabularies	3
2.2	Two-sorted theories	5
2.3	CCV and its complexity classes	7
2.4	The stable marriage problem	8
2.5	Notation	8
3	The new theory VCC^*	8
4	Lexicographical first maximal matching problem is CC-complete	10
4.1	$CCV \leq_m^{AC^0} 3VLFMM$	10
4.2	$VLFMM \leq_m^{AC^0} CCV$	12
4.3	$CCV \leq_m^{AC^0} 3LFMM$	13
4.4	$CCV \not\leq_m^{AC^0} CCV$	13
4.5	$LFMM \leq_m^{AC^0} CCV \not\leq_m^{AC^0} CCV$	14
5	The theory VCC^* contains VNL	15
6	The SM problem is CC-complete	19
6.1	$3LFMM$ is AC^0 -many-one-reducible to SM, MOSM and WOSM	19
6.2	MOSM and WOSM are AC^0 -many-one-reducible to CCV	20
6.2.1	THREE-VALUED CCV is CC -complete	21
6.2.2	A fixed-point method for solving stable marriage problems	22
7	Conclusion and future work	31

1 Introduction

Comparator networks were originally introduced as a method of sorting numbers (as in Batcher's even-odd merge sort [2]), but they are still interesting when the numbers are restricted to the Boolean values $\{0, 1\}$. A comparator gate has two inputs p, q and two outputs p', q' , where $p' = \min\{p, q\}$ and $q' = \max\{p, q\}$. In the Boolean case (which is the one we consider) $p' = p \wedge q$ and $q' = p \vee q$. A comparator circuit (i.e. network) is presented as a set of m horizontal lines in which the m inputs are presented at the left ends of the lines and the m outputs are presented at the right ends of the lines, and in between there is a sequence of comparator gates, each represented as a vertical arrow connecting some wire w_i with some wire w_j as shown in Fig. 1. These arrows divide each wire into segments, each of which gets a Boolean value. The values of wires w_i and w_j after the arrow are the comparator outputs of the values of wires w_i and w_j right before the arrow, with the tip of the arrow representing the maximum.

The comparator circuit value problem (CCV) is: given a comparator circuit with specified Boolean inputs, determine the output value of a designated wire. To turn this into a complexity class it seems natural to use a reducibility notion that is weak but fairly robust. Thus we define CC to consist of those problems (uniform) AC^0 many-one-reducible to CCV. However Subramanian [9] studied the complexity of CCV using a stronger notion of reducibility.

Thus his class, which we denote CC^{Subr} , consists of those problems log-space (many-one)-reducible to CCV. It turns out that a generalization of many-one AC^0 -reducibility which we will call *oracle AC^0 -reducibility* (called simply AC^0 reducibility in [3]), is also useful. Standard complexity classes such as AC^0 , L (log space), NL (nondeterministic log space), NC , and P are all closed under this AC^0 oracle reducibility. We denote the closure of CCV under this reducibility by CC^* .

We will show that

$$NL \subseteq CC \subseteq CC^{\text{Subr}} \subseteq CC^* \subseteq P \quad (1.1)$$

The last inclusion is obvious because CCV is a special case of the monotone circuit value problem, which is clearly in P . The inclusion $CC \subseteq CC^{\text{Subr}}$ follows because AC^0 functions are also log-space functions. The inclusion $CC^{\text{Subr}} \subseteq CC^*$ follows from the first inclusion, which in turn is a strengthening of a result in [6] (attributed to Feder) showing that $NL \subseteq CC^{\text{Subr}}$. Of course all three comparator classes coincide if it turns out that CC is closed under oracle AC^0 -reductions, but we do not know how to show this.

Note that comparator circuits are more restricted than monotone Boolean circuits because each comparator output has fan-out one. This leads to the open question of whether $CC^* \subsetneq P$. A second open question is whether the complexity classes CC^* and NC are incomparable. (Here NC is the class of problems computed by uniform circuit families of polynomial size and polylog depth, and satisfies $NL \subseteq NC \subseteq P$.) The answers could be different if we replaced CC^* by CC^{Subr} or CC , although $CC \subseteq NC$ iff $CC^* \subseteq NC$ because NC is closed under oracle AC^0 reductions.

The above classes associated with CCV are also interesting because they have several disparate complete problems. As shown in [6, 9] both the lexicographical first maximal matching problem (LFMM) and the stable marriage problem (SM) are complete for CC^{Subr} under log-space reductions¹. The SM problem is especially interesting: Introduced by Gale and Shapley in 1962 [4], it has

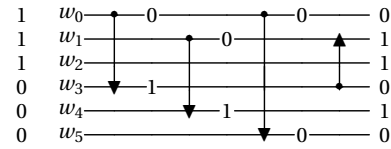


Figure 1

¹ The first author outlined a proof that LFMM is complete under NC^1 reductions in unpublished notes from 1983.

since been used to pair medical interns with hospital residencies jobs in the USA. SM can be stated as follows: Given n men and n women, each with a complete ranking according to preference of all n members of the opposite sex, find a complete matching of the men and women such that there are no two people of opposite sex who would both rather have each other than their current partners. Gale and Shapley proved that such a ‘stable’ matching always exists, although it may not be unique. Subramanian [9] showed that SM treated as a search problem (i.e. find any stable marriage) is complete for CC under log-space reducibility.

Strangely the CC classes have received very little attention since Subramanian’s papers [8, 9]. The present paper contributes to their complexity theory by sharpening these early results and simplifying their proofs. For example we prove that the three problems C_{CV}, LFMM, and SM are inter-reducible under many-one AC^0 -reductions as opposed to log-space reductions. Also we introduce a three-valued logic version of C_{CV} to facilitate its reduction to SM. Our paper contributes to the proof complexity of the classes by introducing a two-sorted formal theory VCC* which captures the class CC* and which can formalize the proofs of the above results.

Our theory VCC* is a two-sorted theory developed in the way described in [3, Chapter 9]. In general this method associates a theory VC with a suitable complexity class C in such a way that a function is in FC, the function class associated with C, iff it is provably total in VC. (A string-valued function is in FC iff it is polynomially bounded and its bit-graph is in C.) This poses a problem for us because the provably-total functions in a theory are always closed under composition, but it is quite possible that neither of the function classes FCC and FCC^{Subr} is closed under composition. That is why we define the class CC* to consist of the problems AC^0 -oracle-reducible (see Definition 3 below) to C_{CV}, rather than the problems AC^0 many-one reducible to C_{CV}, which comprise CC. It is easy to see that the functions in FCC* are closed under composition, and these are the functions that are provably total in our theory VCC*.

The above paragraph illustrates one way that studying proof complexity can contribute to main-stream complexity theory, namely by mandating the introduction of the more robust version CC* of CC and CC^{Subr}. Another way is by using the simple two-sorted syntax of our theories to demonstrate AC^0 reductions. Thus Theorem 1 below states that a simple syntactic class of formulas represents precisely the AC^0 relations. In general it is much easier to write down an appropriate such formula than to describe a uniform circuit family or alternating Turing machine program.

Once we describe our theory VCC* in Sections 2.1 and 2.2, the technical part of our proofs involve high-level descriptions of comparator circuits and algorithms. We do not say much about formalizing the proofs in VCC* since in most cases this part is relatively straightforward. However there are some cases (Theorems 15 and 27) where the proofs require a counting argument, and for these we use the fact (Corollary 8) that the counting theory VTC⁰ is contained in VCC*.

2 Preliminaries

2.1 Two-sorted vocabularies

We use two-sorted vocabularies for our theories as described by Cook and Nguyen [3]. Two-sorted languages have variables x, y, z, \dots ranging over \mathbb{N} and variables X, Y, Z, \dots ranging over finite subsets of \mathbb{N} , interpreted as bit strings. Two sorted vocabulary \mathcal{L}_A^2 includes the usual symbols 0, 1, +, ·, =, ≤ for arithmetic over \mathbb{N} , the length function $|X|$ for strings ($|X|$ is zero if X is empty, otherwise $1 + \max(X)$), the set membership relation \in , and string equality $=_2$ (subscript 2 is usually omitted). We will use the notation $X(t)$ for $t \in X$, and think of $X(t)$ as the t^{th} bit in the string X .

The number terms in the base language \mathcal{L}_A^2 are built from the constants 0, 1, variables x, y, z, \dots and length terms $|X|$ using + and ·. The only string terms are string variables, but when we

extend \mathcal{L}_A^2 by adding string-valued functions, other string terms will be built as usual. The atomic formulas are $t = u$, $X = Y$, $t \leq u$, $t \in X$ for any number terms x, y and string variables X, Y . Formulas are built from atomic formulas using \wedge, \vee, \neg and $\exists x, \exists X, \forall x, \forall X$. Bounded number quantifiers are defined as usual, and bounded string quantifier $\exists X \leq t, \varphi$ stands for $\exists X (|X| \leq t \wedge \varphi)$ and $\forall X \leq t, \varphi$ stands for $\forall X (|X| \leq t \rightarrow \varphi)$, where X does not appear in term t .

The class Σ_0^B consists of all \mathcal{L}_A^2 -formulas with no string quantifiers and only bounded number quantifiers. The class Σ_1^B consists of formulas of the form $\exists \vec{X} < \vec{t} \varphi$, where $\varphi \in \Sigma_0^B$ and the prefix of the bounded quantifiers might be empty. These classes are extended to Σ_i^B (and Π_i^B) for all $i \geq 0$, in the usual way. More generally we write $\Sigma_i^B(\mathcal{L})$ to denote the class of Σ_i^B -formulas which may have function and predicate symbols from $\mathcal{L} \cup \mathcal{L}_A^2$.

Two-sorted complexity classes contain relations $R(\vec{x}, \vec{X})$, where \vec{x} are number arguments and \vec{X} are string arguments. (In the sequel we refer to a relation $R(\vec{x}, \vec{X})$ as a *decision problem*: given (\vec{x}, \vec{X}) determine whether $R(\vec{x}, \vec{X})$ holds.) In defining complexity classes using machines or circuits, the number arguments are represented in unary notation and the string arguments are represented in binary. The string arguments are the main inputs, and the number arguments are auxiliary inputs that can be used to index the bits of strings. Using these input conventions, we define the two-sorted version of AC^0 to be the class of relations $R(\vec{x}, \vec{X})$ such that some alternating Turing machine accepts R in time $O(\log n)$ with a constant number of alternations. Then the descriptive complexity characterization of AC^0 gives rise to the following theorem [3, Chapter 4].

► **Theorem 1.** *A relation $R(\vec{x}, \vec{X})$ is in AC^0 iff it is represented by some Σ_0^B -formula $\varphi(\vec{x}, \vec{X})$.*

Given a class of relations C , we associate a class FC of string-valued functions $F(\vec{x}, \vec{X})$ and number functions $f(\vec{x}, \vec{X})$ with C as follows. We require these functions to be p -bounded, i.e., $|F(\vec{x}, \vec{X})|$ and $f(\vec{x}, \vec{X})$ are bounded by a polynomial in \vec{x} and $|\vec{X}|$. Then we define FC to consist of all p -bounded number functions whose graphs are in C and all p -bounded string functions whose bit graphs are in C . (Here the *bit graph* of $F(\vec{x}, \vec{X})$ is the relation $B_F(i, \vec{x}, \vec{X})$ which holds iff the i th bit of $F(\vec{x}, \vec{X})$ is 1.)

Most of the computational problems we consider here can be nicely expressed as decision problems (i.e. relations), but the stable marriage problem SM is an exception, because in general a given instance has more than one solution (i.e. there is more than one stable marriage). Thus SM is properly described as a search problem. Following [3, Section 8.5] we define a two-sorted *search problem* Q_R to be a multivalued function with graph $R(\vec{x}, \vec{X}, Z)$, so

$$Q_R(\vec{x}, \vec{X}) = \{Z \mid R(\vec{x}, \vec{X}, Z)\} \quad (2.1)$$

The search problem is *total* if the set $Q_R(\vec{x}, \vec{X})$ is non-empty for all \vec{x}, \vec{X} . The search problem is a *function problem* if $|Q_R(\vec{x}, \vec{X})| = 1$ for all \vec{x}, \vec{X} . A function $F(\vec{x}, \vec{X})$ solves Q_R if

$$F(\vec{x}, \vec{X}) \in Q_R(\vec{x}, \vec{X})$$

for all \vec{x}, \vec{X} .

Here we will be concerned only with total search problems.

► **Definition 2.** Let C be a complexity class. A relation $R_1(\vec{x}, \vec{X})$ is C -many-one-reducible to a relation $R_2(\vec{y}, \vec{Y})$ (written $R_1 \leq_m^C R_2$) if there are functions \vec{f}, \vec{F} in FC such that

$$R_1(\vec{x}, \vec{X}) \leftrightarrow R_2(\vec{f}(\vec{x}, \vec{X}), \vec{F}(\vec{x}, \vec{X})).$$

A search problem $Q_{R_1}(\vec{x}, \vec{X})$ is C -many-one-reducible to a search problem $Q_{R_2}(\vec{y}, \vec{Y})$ if there are functions G, \vec{f}, \vec{F} in FC such that

$$G(\vec{x}, \vec{X}, Z) \in Q_{R_1}(\vec{x}, \vec{X}) \text{ for all } Z \in Q_{R_2}(\vec{f}(\vec{x}, \vec{X}), \vec{F}(\vec{x}, \vec{X})).$$

Here we are mainly interested in the cases that C is either AC^0 or L (log space). We also need a generalization of many-one AC^0 -reducibility called simply AC^0 -reducibility in [3, Definition 9.1.1], which we will call oracle AC^0 -reducibility. Roughly speaking a function or relation is AC^0 -oracle-reducible to a collection \mathcal{L} of functions and relations if it can be computed by a uniform polynomial size constant depth family of circuits which have unbounded fan-in gates computing functions and relations from \mathcal{L} (i.e. ‘oracle gates’), in addition to Boolean gates. Formally:

► **Definition 3.** A string function F is AC^0 -oracle-reducible to a collection \mathcal{L} of relations and functions (written $F \leq_{\mathcal{L}}^{AC^0}$) if there is a sequence of string functions $F_1, \dots, F_n = F$ such that each F_i is p -bounded and its bit graph is represented by a $\Sigma_0^B(\mathcal{L}, F_1, \dots, F_{i-1})$ -formula.

A number function f is AC^0 -oracle-reducible to \mathcal{L} if $f = |F|$ for some string function F which is AC^0 -reducible to \mathcal{L} . A relation R is AC^0 -oracle-reducible to \mathcal{L} if its characteristic function is AC^0 -oracle-reducible to \mathcal{L} .

We note that standard small complexity classes including AC^0 , TC^0 , NC^1 , NL and P (as well as their corresponding function classes) are closed under oracle AC^0 -reductions.

2.2 Two-sorted theories

The theory V^0 for AC^0 is the basis for developing theories for small complexity classes within P in [3]. V^0 has the vocabulary \mathcal{L}_A^2 and is axiomatized by the set of *2-BASIC* axioms as given in Fig. 2, which express basic properties of symbols in \mathcal{L}_A^2 , together with the *comprehension* axiom schema

$$\Sigma_0^B\text{-COMP:} \quad \exists X \leq y \forall z < y (X(z) \leftrightarrow \varphi(z)),$$

where $\varphi \in \Sigma_0^B(\mathcal{L}_A^2)$ and X does not occur free in φ . Note that the axioms B1–B12 of *2-BASIC* are based on the *1-BASIC* axioms of theory $I\Delta_0$; the axioms L1 and L2 characterize $|X|$.

Although V^0 has no explicit induction axiom, nevertheless, using $\Sigma_0^B\text{-COMP}$ and the fact that $|X|$ produces the maximum element of the finite set X , the following schemes are provable in V^0 for every formula $\varphi \in \Sigma_0^B(\mathcal{L}_A^2)$

$$\begin{aligned} \Sigma_0^B\text{-IND:} & \quad [\varphi(0) \wedge \forall x (\varphi(x) \rightarrow \varphi(x+1))] \rightarrow \forall x \varphi(x), \\ \Sigma_0^B\text{-MIN:} & \quad \varphi(y) \rightarrow \exists x (\varphi(x) \wedge \neg \exists z < x \varphi(z)), \\ \Sigma_0^B\text{-MAX:} & \quad \varphi(0) \rightarrow \exists x \leq y [\varphi(x) \wedge \neg \exists z \leq y (z > x \wedge \varphi(z))]. \end{aligned}$$

In general, we say that a string function $F(\vec{x}, \vec{X})$ is Σ_1^B -definable (or provably total) in a two-sorted theory \mathcal{T} if there is a Σ_1^B formula $\varphi(\vec{x}, \vec{X}, Y)$ representing the graph $Y = F(\vec{x}, \vec{X})$ of F such that $\mathcal{T} \vdash \forall \vec{x} \forall \vec{X} \exists! Y \varphi(\vec{x}, \vec{X}, Y)$. Similarly for a number function $f(\vec{x}, \vec{X})$.

It was shown in [3, Chapter 5] that V^0 is finitely axiomatizable, and a function is provably total in V^0 iff it is in FAC^0 .

In [3, Chapter 9], Cook and Nguyen develop a general method for associating a theory VC with certain complexity classes $C \subseteq P$, where VC extends V^0 with an additional axiom asserting the existence of a solution to a complete problem for C . In order for this method to work, the class C must be closed under AC^0 -oracle-reducibility (Definition 3). The method shows how to define a universal conservative extension \overline{VC} of VC , where \overline{VC} has string function symbols for precisely the string functions of VC , and terms for precisely the number functions of VC . Further, \overline{VC} proves the $\Sigma_0^B(\mathcal{L})\text{-IND}$ and $\Sigma_0^B(\mathcal{L})\text{-MIN}$ schemes, where \mathcal{L} is the vocabulary of \overline{VC} . It follows from the Herbrand Theorem that the provably total functions of both VC and \overline{VC} are precisely those in FC .

Using this framework Cook and Nguyen define specific theories for several complexity classes and give examples of theorems formalizable in each theory. The theories of interest to us in this

-
- | | |
|--|---|
| B1. $x + 1 \neq 0$
B2. $x + 1 = y + 1 \rightarrow x = y$
B3. $x + 0 = x$
B4. $x + (y + 1) = (x + y) + 1$
B5. $x \cdot 0 = 0$
B6. $x \cdot (y + 1) = (x \cdot y) + x$
B7. $(x \leq y \wedge y \leq x) \rightarrow x = y$ | B8. $x \leq x + y$
B9. $0 \leq x$
B10. $x \leq y \vee y \leq x$
B11. $x \leq y \leftrightarrow x < y + 1$
B12. $x \neq 0 \rightarrow \exists y \leq x (y + 1 = x)$
L1. $X(y) \rightarrow y < X $
L2. $y + 1 = X \rightarrow X(y)$ |
|--|---|

$$\mathbf{SE.} (|X| = |Y| \wedge \forall i < |X| (X(i) = Y(i))) \rightarrow X = Y$$

■ **Figure 2** The 2-BASIC axioms

paper are VTC^0 , VNC^1 , VNL and VP for the complexity classes TC^0 , NC^1 , NL and P respectively. All of these theories have vocabulary \mathcal{L}_A^2 . Let $\langle x, y \rangle$ denote the *pairing function*, which is the \mathcal{L}_A^2 term $(x + y)(x + y + 1) + 2y$. The theory VTC^0 is axiomatized by the axioms of V^0 and the axiom:

$$\mathbf{NUMONES:} \quad \exists Z \leq 1 + \langle n, n \rangle, \delta_{\text{NUM}}(n, X, Z), \quad (2.2)$$

where the formula $\delta_{\text{NUM}}(n, X, Z)$ asserts that Z is a matrix consisting of n rows such that for every $y \leq n$, the y^{th} row of Z encodes the number of ones in the prefix of length y of the binary string X . Thus, the n^{th} row of Z essentially “counts” the number of ones in X . Because of this counting ability, VTC^0 can prove *the pigeonhole principle* $\text{PHP}(n, F)$ saying that if F maps a set of $n + 1$ elements to a set of n elements, then F is not an injection.

The theory VNC^1 is axiomatized by the axioms of V^0 and the axiom:

$$\mathbf{MFV:} \quad \exists Y \leq 2n + 1, \delta_{\text{MFV}}(n, F, I, Y), \quad (2.3)$$

where F and I encode a monotone Boolean formula with n literals and its input respectively, and the formula $\delta_{\text{MFV}}(n, G, I, Y)$ holds iff Y correctly encodes the evaluation of the formula encoded in F on input I . Recall that the *monotone Boolean formula value* problem is complete for NC^1 .

The theory VP is axiomatized by the axioms of V^0 and the axiom MCV , which is defined very similarly to MFV , but the *monotone circuit value* problem is used instead.

The theory VNL is axiomatized by the axioms of V^0 and the axiom:

$$\mathbf{CONN:} \quad \exists U \leq \langle n, n \rangle + 1, \delta_{\text{CONN}}(n, E, U), \quad (2.4)$$

where E encodes the edge relation of a directed graph G with n vertices v_0, \dots, v_{n-1} , and the formula $\delta_{\text{CONN}}(n, E, U)$ is intended to mean that U is a matrix of n rows, where each row has a Boolean value for each vertex in G , and $U(d, i)$ holds iff vertex v_i has distance at most d from v_0 . More directly, the formula $\delta_{\text{CONN}}(n, E, U)$ asserts

$$\begin{aligned} U(0, i) \text{ holds iff } i = 0, \text{ and } U(d + 1, i) \text{ holds iff either } U(d, i) \text{ holds or} \\ \text{there is } j \text{ such that } U(d, j) \text{ holds and there is an edge in } G \text{ from } v_j \text{ to } v_i. \end{aligned} \quad (2.5)$$

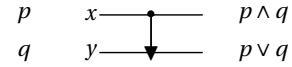
Similar to what is currently known about complexity classes, it was shown in [3, Chapter 9] that the following inclusions hold:

$$\text{V}^0 \subsetneq \text{VTC}^0 \subseteq \text{VNC}^1 \subseteq \text{VNL} \subseteq \text{VP}. \quad (2.6)$$

2.3 CCV and its complexity classes

A *comparator gate* is a function $C : \{0, 1\}^2 \rightarrow \{0, 1\}^2$, that takes an input pair (p, q) and outputs a pair $(p \wedge q, p \vee q)$. Intuitively, the first output in the pair is the smaller bit among the two input bits p, q , and the second output is the larger bit.

We will use the graphical notation on the right to denote a comparator gate, where x and y denote the names of the wires, and the direction of the arrow denotes the direction to which we move the larger bit as shown in the picture.



A *comparator circuit* is a directed acyclic graph consisting of: *input nodes* with in-degree zero and out-degree one, *output nodes* with in-degree one and out-degree zero, and *internal nodes* with in-degree two and out-degree two, where each internal node is labelled with a comparator gate. We also require each output computed by a comparator gate has fan-out exactly one. Under this definition, each comparator circuit can be seen as consisting of the wires that carry the bit values and are arranged in parallel, and each comparator gate connects exactly two wires as previously shown in Fig. 1.

The *comparator circuit value* problem (CCV) is the decision problem: Given a comparator circuit, an input assignment, and a designated wire, decide whether the circuit outputs one on that wire.

Formally CCV is a two-sorted relation $R(\vec{x}, \vec{X})$ like those discussed in Section 2. An instance of the decision problem is encoded in some reasonable way by the variables (\vec{x}, \vec{X}) . The exact encoding is not important. An example encoding is (n, m, i, I, G) where n, m and i are number variables and I and G are string variables. Here n is the number of wires in the comparator circuit, m is the number of comparator gates, i is the designated wire, I specifies the sequence of input values to the wires, and G specifies the sequence of m comparator gates in the circuit.

► **Definition 4.** The complexity class CC (resp. CC^{Subr} , CC^*) is the class of decision problems (i.e. relations) that are AC^0 many-one-reducible (resp. log space-reducible, AC^0 oracle-reducible) to CCV. A decision problem R is CC-complete (resp. CC^{Subr} -complete, CC^* -complete) if the respective class is the closure of R under the corresponding reducibility. We say that R is CC_{all} -complete if it is complete in all three senses.

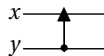
The next result is a straightforward consequence of (1.1) and the definitions involved.

► **Lemma 5.** *If a decision problem is CC-complete then it is CC_{all} -complete.*

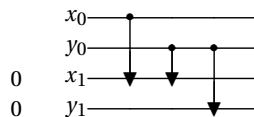
In the above definition of comparator circuit, each comparator gate can point in either direction, upward or downward (see Fig. 1). However, it is not hard to show the following.

► **Proposition 1.** *The CCV problem with the restriction that all comparator gates point in the same direction is CC-complete.*

Proof. Suppose we have a gate with the arrow pointing upward like following:



We will build a circuit that outputs the same values as those of x and y , but all the gates will now point downward.



It is not hard to see that the wires x_1 and y_1 in this new comparator circuit will output the same values with the wires x and y respectively in the original circuit. For the general case, we can simply make copies of all wires for each layer of the comparator circuit, where each copy of a wire will be used to carry value of a wire at a single layer of the circuit. Then apply the above construction to simulate the effect of each gates. Note that additional comparator gates are also needed to forward the values of the wires from one layer to another, in the same way that we use the gate $\langle y_0, y_1 \rangle$ to forward the value carried in wire y_0 to wire y_1 in the above construction. ◀

2.4 The stable marriage problem

An instance of the stable marriage problem (SM) is given by a number n (specifying the number of men and the number of women), together with a preference list for each man and each woman specifying a total ordering on all people of the opposite sex. The goal of SM is to produce a perfect matching between men and women, i.e., a bijection from the set of men to the set of women, such that the following *stability* condition is satisfied: there are no two people of the opposite sex who like each other more than their current partners. Such a stable solution always exists, but it may not be unique. Thus SM is a search problem (2.1), rather than a decision problem.

However there is always a unique *man-optimal* and a unique *woman-optimal* solution. In the man-optimal solution each man is matched with a woman whom he likes at least as well as any woman that he is matched with in any stable solution. Dually for the woman-optimal solution. Thus we define the *man-optimal stable marriage decision problem* (MOSM) as follows: given an instance of the stable marriage problem together with a designated man-woman pair, determine whether that pair is married in the man-optimal stable marriage. We define the *woman-optimal stable marriage decision problem* (WOSM) analogously.

We show here that the search version and the decision versions are computationally equivalent, and each is complete for CC with respect to the appropriate reducibility in Definition 2. Section 6.1 shows how to reduce the lexicographical first maximal matching problem (which is complete for CC) to the SM search problem, and Section 6.2 shows how to reduce both the MOSM and WOSM problems to CCV.

2.5 Notation

We write the notation “ $(T \vdash)$ ” in front of the statement of a theorem to indicate that the statement is formulated and proved within the theory T .

We often use two-dimensional matrices to encode binary relations, e.g. the edge relation of a graph, matching, etc. In this paper, it is more convenient for our purpose to index the entries of matrices starting from 0 instead of 1. In other words, if $X_{n \times n}$ is a two-dimensional matrix, then entries of X consist of all $X(i, j)$ for $0 \leq i, j < n$, and $X(0, 0)$ is the top leftmost entry of X .

3 The new theory VCC*

We encode a comparator circuit as a sequence of pairs $\langle i, j \rangle$, where each pair $\langle i, j \rangle$ encodes a comparator gate that swaps the values of the wires i and j iff the value on wire i is greater than the value of wire j . We also allow “dummy” gates of the form $\langle i, i \rangle$, which do nothing. We want to define a formula $\delta_{CCV}(m, n, X, Y, Z)$, where

- X encodes a comparator circuit with m wires and n gates as sequence of n pairs $\langle i, j \rangle$ with $i, j < m$, and we write $(X)^i$ to denote the i^{th} comparator gate of the circuit encoded by X .
- $Y(i)$ encodes the input value for the i^{th} wire as a truth value, and

- Z is an $(n+1) \times m$ matrix, where $Z(i, j)$ is the value of wire j at layer i , where each layer is simply a sequence of values carried by all the wires right after a comparator gate.

Although X encodes a circuit with only n gates, the matrix Z actually encodes $n+1$ layers since we use the first layer to encode the input values of the comparator circuit. The formula $\delta_{CCV}(m, n, X, Y, Z)$ holds iff Z encodes the correct values of the layers computed by the comparator circuit encoded by X with input Y , and thus is defined as the following Σ_0^B -formula:

$$\begin{aligned} & \forall i < m (Y(i) \leftrightarrow Z(0, i)) \wedge \forall i < n \forall x < m \forall y < m, \\ (X)^i = \langle x, y \rangle \rightarrow & \left[\begin{array}{l} Z(i+1, x) \leftrightarrow (Z(i, x) \wedge Z(i, y)) \\ \wedge \quad Z(i+1, y) \leftrightarrow (Z(i, x) \vee Z(i, y)) \\ \wedge \quad \forall j < m [(j \neq x \wedge j \neq y) \rightarrow (Z(i+1, j) \leftrightarrow Z(i, j))] \end{array} \right] \end{aligned} \quad (3.1)$$

► **Definition 6.** The theory VCC^* has vocabulary \mathcal{L}_A^2 and is axiomatized by the axioms of V^0 and the following axiom

$$CCV: \quad \exists Z \leq \langle m, n+1 \rangle + 1, \delta_{CCV}(m, n, X, Y, Z), \quad (3.2)$$

where $\delta_{CCV}(m, n, X, Y, Z)$ is defined as in (3.1).

There is a technical lemma required to show that VCC^* fits the framework described in [3, Chapter 9]. Define $F_{CCV}(m, n, X, Y)$ to be the Z satisfying $\delta_{CCV}(m, n, X, Y, Z)$ (with each $Z(i)$ set false when it is not determined). We need to show that the *aggregate* F_{CCV}^* of F_{CCV} is Σ_1^B -definable in VCC^* , where (roughly speaking) F_{CCV}^* is the string function that gathers the values of F_{CCV} for a polynomially long sequence of arguments. The nature of CC circuits makes this easy: The sequence of outputs for a sequence of circuits can be obtained from a single circuit which computes them all in parallel: the lines of the composite circuit comprise the union of the lines of each component circuit.

Thus the framework of [3, Chapter 9] does apply to VCC^* , and in particular the theory $\overline{VCC^*}$ is a universal conservative extension of VCC^* whose function symbols are precisely those in the function class FCC.

It is hard to work with VCC^* up to this point since we have not shown whether VCC^* can prove the definability of basic counting functions (as in VTC^0). However, we have the following theorem.

► **Theorem 7** ($VNC^1 \subseteq VCC^*$). *The theory VCC^* proves the axiom MFV defined in (2.3).*

Proof. Observe that each comparator gate can produce simultaneously an AND gate and an OR gate with the only restriction that each of these gates must have fan-out one. However, since all AND and OR gates of a monotone Boolean formula also have fan-out one, each instance of the Boolean formula value problem is a special case of CCV. ◀

Since $VTC^0 \subseteq VNC^1$ (see (2.6)) the next result is an immediate consequence of this theorem.

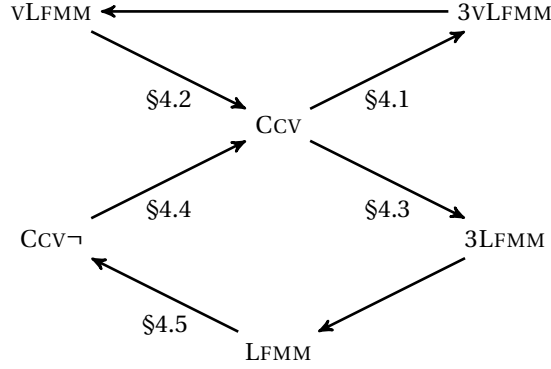
► **Corollary 8** ($VTC^0 \subseteq VCC^*$). *The theory VCC^* proves the axiom NUMONES defined in (2.2).*

This corollary is important since it allows us to use the counting ability of VTC^0 freely in VCC^* proofs. In particular using counting and induction on the layers of a comparator circuit, we can prove in VTC^0 the following fundamental property of comparator circuits.

► **Corollary 9.** *Given a comparator circuit computation, the theory VTC^0 (and hence VCC^*) proves that all layers of the computation have the same number of ones and zeros.*

► **Theorem 10** ($VCC^* \subseteq VP$). *The theory VP proves the axiom CCV defined in (3.2).*

Proof. This easily follows since CCV is a special case of the *monotone circuit value* problem. ◀



■ **Figure 3** Label of an arrow denotes the section in which the reduction is described. Arrows without labels denote trivial reductions.

4 Lexicographical first maximal matching problem is CC-complete

Let $G = (V, W, E)$ be a bipartite graph, where $V = \{v_i\}_{i=0}^{m-1}$, $W = \{w_i\}_{i=0}^{n-1}$ and $E \subseteq V \times W$. The *lexicographical first maximal matching* (lfm-matching) is the matching produced by successively matching each vertex v_0, \dots, v_{m-1} to the least vertex available in W .

Formally, let $E_{m \times n}$ be a matrix encoding the edge relation of a bipartite graph with m bottom nodes and n top nodes, where $E(i, j) = 1$ iff the bottom node v_i is adjacent to the top node w_j . Let L be a matrix of the same size as E with the following intended interpretation: $L(i, j) = 1$ iff the edge (v_i, w_j) is in the lfm-matching. We can define a Σ_0^B -formula $\delta_{\text{LFMM}}(m, n, X, L)$, which asserts that L properly encodes the lfm-matching of the bipartite graph represented by X as follows:

$$\forall i < m \forall j < n, L(i, j) \leftrightarrow \left[\begin{array}{l} E(i, j) \wedge \forall k < j \forall \ell < i (\neg L(i, k) \wedge \neg L(\ell, j)) \\ \wedge \forall k < j (\neg E(i, k) \vee \exists i' < i L(i', k)) \end{array} \right]. \quad (4.1)$$

In this section we will show that two decision problems concerning the lexicographical matching of a bipartite graph are CC-complete (under many-one AC^0 -reductions). The lexicographical first maximal matching problem (LFMM) is to decide if a designated edge belongs to the lfm-matching of a bipartite graph G . The vertex version of lexicographical first maximal matching problem (VLFMM) is to decide if a designated top node is matched in the lfm-matching of a bipartite graph G . LFMM is the usual way to define a decision problem for lexicographical first maximal matching as seen in [6, 9]; however, as shown in Sections 4.1 and 4.2, the VLFMM problem is even more closely related to the CCV problem.

We will show that the following two more restricted lexicographical matching problems are also CC-complete. We define 3LFMM to be the restriction of LFMM to bipartite graphs of degree at most three. We define 3VLFMM to be the restriction of VLFMM to bipartite graphs of degree at most three.

To show that the problems defined above are equivalent under many-one AC^0 -reductions, it turns out that we also need the following intermediate problem. A negation gate flips the value on a wire. For comparator circuits with negation gates, we allow negation gates to appear on any wire (see the left diagram of Fig. 7 below for an example). The comparator circuit value problem with negation gates (CCV \neg) is: given a comparator circuit with negation gates and input assignment, and a designated wire, decide if that wire outputs 1.

All reductions in this section are summarized in Fig. 3.

4.1 $CCV \leq_m^{AC^0} 3VLFMM$

By Proposition 1 it suffices to consider only instances of CCV in which all comparator gates point upward. We will show that these instances of CCV are AC^0 -many-one-reducible to instances of $3VLFMM$, which consist of bipartite graphs with *degree at most three*.

The key observation is that a comparator gate on the left below closely relates to an instance of $3VLFMM$ on the right. We use the top nodes p_0 and q_0 to represent the values p_0 and q_0 carried by the wires x and y respectively before the comparator gate, and the nodes p_1 and q_1 to represent the values of x and y after the comparator gate, where a top node is matched iff its respective value is one.



If nodes p_0 and q_0 are not previously matched, i.e. $p_0 = q_0 = 0$ in the comparator circuit, then edges $\langle x, p_0 \rangle$ and $\langle y, q_0 \rangle$ are added to the lfm-matching. So the nodes p_1 and q_1 are not matched. If p_0 is previously matched, but q_0 is not, then edges $\langle x, p_1 \rangle$ and $\langle y, q_0 \rangle$ are added to the lfm-matching. So the node p_1 will be matched but q_1 will remain unmatched. The other two cases are similar.

Thus, we can reduce a comparator circuit to the bipartite graph of an $3VLFMM$ instance by converting each comparator gate into a “gadget” described above. We will describe our method through an example, where we are given the comparator circuit in Fig. 4.

We divide the comparator circuit into vertical layers 0, 1, and 2 as shown in Fig. 4. Since the circuit has three wires a, b and c , for each layer i , we use six nodes, including three top nodes a_i, b_i and c_i representing the values of the wires a, b and c respectively, and three bottom nodes a'_i, b'_i, c'_i , which are auxiliary nodes used to simulate the effect of the comparator gate at layer i .

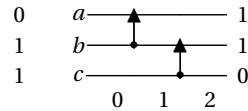
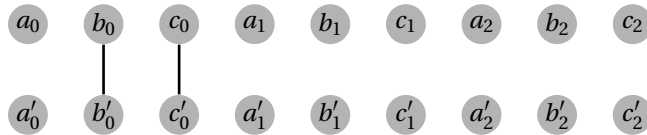
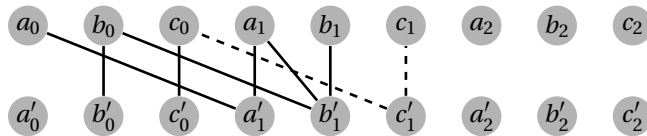


Figure 4

Layer 0: This is the input layer, so we add an edge $\{x_i, x'_i\}$ iff the wire x takes input value 1. In this example, since b and c are wires taking input 1, we need to add the edges $\{b_0, b'_0\}$ and $\{c_0, c'_0\}$.

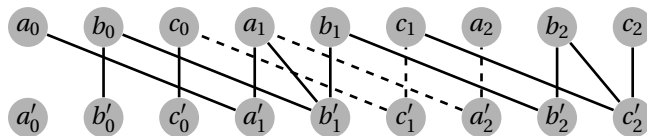


Layer 1: We then add the gadget simulating the comparator gate from wire b to wire a as follows.



Since the value of wire c does not change when going from layer 0 to layer 1, we can simply propagate the value of c_0 to c_1 using the pair of dashed edges in the picture.

Layer 2: We proceed very similarly to layer 1 to get the following bipartite graph.



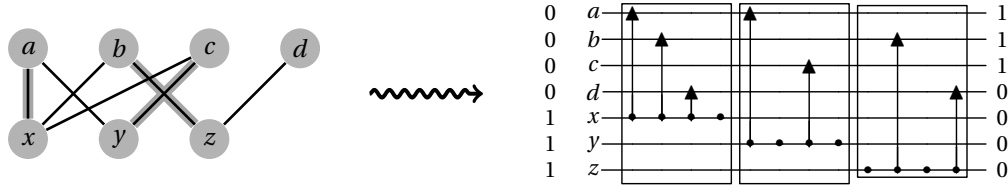
Finally, we can get the output values of the comparator circuit by looking at the “output” nodes a_2, b_2, c_2 of this bipartite graph. We can easily check that a_2 is the only node that remains unmatched, which corresponds exactly to the only zero produced by wire a of the comparator circuit in Fig. 4.

► Remark. The construction above is an AC^0 many-one reduction since each gate in the comparator circuit can be reduced to exactly one gadget in the bipartite graph that simulates the effect of the comparator gate. Note that since it can be tedious and unintuitive to work with AC^0 -circuits, it might seem hard to justify that our reduction is an AC^0 -function. However, thanks to Theorem 1, we do not have to work with AC^0 -circuits directly; instead, it is not hard to construct a Σ_0^B -formula that defines the above reduction.

The correctness of our construction can be proved in VCC^* by using Σ_0^B induction on the layers of the circuits and arguing that the matching information of the nodes in the bipartite graph can be correctly translated to the values carried by the wires at each layer.

4.2 $vLFMM \leq_m^{AC^0} CCV$

Consider an instance of $vLFMM$ consisting of a bipartite graph in Fig. 5. Recall that we find the lfm-matching by matching the bottom nodes x, y, \dots successively to the first available node on the top. Hence we can simulate the matching of the bottom nodes to the top nodes using comparator circuit on the right of Fig. 5, where we can think of the moving of a one, say from wire x to wire a , as the matching of node x to node a in the original bipartite graph. In this construction, a top node is matched iff its corresponding wire in the circuit outputs 1.



■ Figure 5

Note that we draw bullets without any arrows going out from them in the circuit to denote dummy gates, which do nothing. These dummy gates are introduced for the following technical reason. Since the bottom nodes might not have the same degree, the position of a comparator gate really depends on the number of edges that do not appear in the bipartite graph, which makes it harder to give a direct AC^0 -reduction. By using dummy gates, we can treat the graph as if it is a complete bipartite graph, where the missing edges are represented by dummy gates. This can easily be shown to be an AC^0 -reduction from $vLFMM$ to CCV , and its correctness can be carried out in VCC^* using Σ_0^B -induction on the layers of the circuit. This together with the reduction from Section 4.1 gives us the following theorem.

► **Theorem 11.** ($VCC^* \vdash$) *The problems CCV , $3vLFMM$ and $vLFMM$ are equivalent under many-one AC^0 -reductions.*

4.3 $CCV \leq_m^{AC^0} 3LFMM$

We start by applying the reduction $CCV \leq_m^{AC^0} 3vLFMM$ of Section 4.1 to get an instance of $3vLFMM$, and notice that the degrees of the top “output” nodes of the resulting bipartite graph, e.g. the nodes a_2, b_2, c_2 in the example of Section 4.1, have degree at most two. Now we show how to

reduce such instances of 3VLFMM (i.e. those whose designated top vertices have degree at most two) to 3LFMM. Consider the graph G with degree at most three and a designated top vertex b of degree two as shown on the left of Fig. 6. We construct a bipartite graph G' , which contains a copy of the graph G and one additional top node w_t and one additional bottom node w_b , and two edges $\{b, w_b\}$ and $\{w_t, w_b\}$, as shown in Fig. 6. Observe that the degree of the new graph G' is at most three.

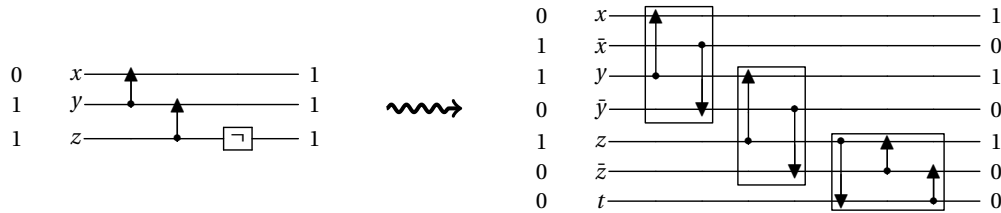


■ **Figure 6**

We treat the resulting bipartite graph G' and the edge $\{w_t, w_b\}$ as an instance of 3LFMM. It is not hard to see that the vertex b is matched in the lfm-matching of the original bipartite graph G iff the edge $\{w_t, w_b\}$ is in the lfm-matching of the new bipartite graph G' .

4.4 $CCV \neg \leq_m^{AC^0} CCV$

Recall that a comparator circuit value problem with negation gates ($CCV \neg$) is the task of deciding, on a comparator circuit with negation gates and an input assignment, if a designated wire outputs one. It should be clear that CCV is a special case of $CCV \neg$ and hence AC^0 many-one-reducible to $CCV \neg$. Here, we show the nontrivial direction that $CCV \neg \leq_m^{AC^0} CCV$.



■ **Figure 7** Successive gates on the left circuit corresponds to the successive boxes of gates on the right circuit.

This reduction is based on “double-rail” logic. Given an instance of $CCV \neg$ consisting of a comparator circuit with negation gates C with its input I and a designated wire s , we construct an instance of CCV consisting of a comparator circuit C' with its input I' and a designated wire s' as follows. For every wire w in I we put in two corresponding wires, w and \bar{w} , in C' . We define input I' of C' such that the input value of \bar{w} is the negation of the input value of w . We want to fix things so that the value carried by the wire \bar{w} at each layer is always the negation of the value carried by w . For any comparator gate $\langle y, x \rangle$ in C we put in both the gate $\langle y, x \rangle$ and a second gate $\langle \bar{x}, \bar{y} \rangle$ in C' immediately after $\langle y, x \rangle$. It is easy to check by De Morgan’s laws that the wires x and y in C' carry the corresponding values of x and y in C , and the wires \bar{x} and \bar{y} in C' carry the corresponding negations of x and y in C .

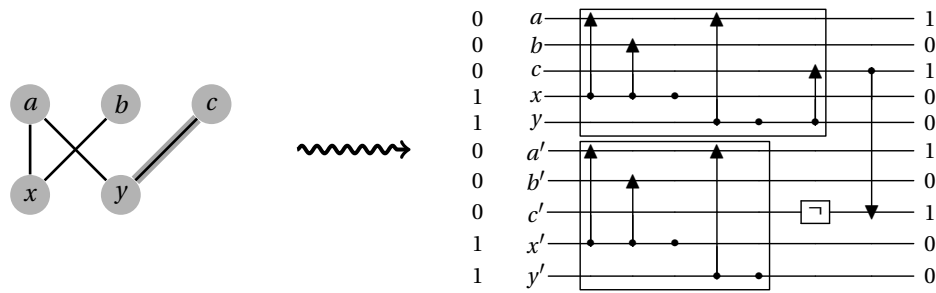
The circuit C' has one extra wire t with input value 0 to help in translating negation gates. For each negation gate on a wire, says z in the example from Fig. 7, we add and three comparator gates $\langle z, t \rangle$, $\langle \bar{z}, z \rangle$, $\langle t, \bar{z} \rangle$ as shown in the right circuit of Fig. 7. Thus t as a temporary “container” that we use to swap the values of carried by the wires z and \bar{z} . Note that the swapping of values of z and \bar{z}

in C' simulates the effect of a negation in C . Also note that after the swap takes place the value of t is restored to 0.

Finally note that the output value of the designated wire s in C is 1 iff the output value of the corresponding wire s in C' with input I' is 1. Thus we set the designated wire s' in I' to be s .

4.5 $\text{LFMM} \leq_m^{\text{AC}^0} \text{CCV}\neg$

Consider an instance of LFMM consisting of a bipartite graph on the left of Fig. 8, and a designated edge $\{y, c\}$. Without loss of generality, we can safely ignore all top vertices occurred after c , all bottom vertices occurred after y and all the edges associated with them, since they are not going to affect the outcome of the instance. Using the construction from Section 4.2, we can simulate the matching of the bottom nodes to the top nodes using comparator circuit in the upper box on the right of Fig. 8.



■ Figure 8

We keep another running copy of this simulation on the bottom, (see the wires labelled a', b', c', x', y' in Fig. 8). The only difference is that the comparator gate $\langle y', c' \rangle$ corresponding to the designated edge $\{y, c\}$ is not added. We finally add a negation gate on c' and a comparator gate $\langle c, c' \rangle$. We let the desired output of the CCV instance be the output of c , since c outputs 1 iff the edge $\{y, c\}$ is added to the lfm-matching. It is not hard to see that such construction can be generalized, and the output correctly computes if the designated edge is in the lfm-matching.

► **Theorem 12.** ($\text{VCC}^* \vdash$) *The problems CCV, CCV \neg , 3LFMM and LFMM are equivalent under many-one AC^0 -reductions.*

Combined with the results from Sections 4.1 and 4.2, we have the following corollary.

► **Corollary 13.** ($\text{VCC}^* \vdash$) *The problems CCV, 3VLFMM, VLFMM, CCV \neg , 3LFMM and LFMM are equivalent under many-one AC^0 -reductions.*

Since CCV \neg is complete for CC, we can use comparator circuits to decide the complement of the CCV problem: given a comparator circuit and an input assignment, does a designated wire output 0? Thus, we have the following corollary.

► **Corollary 14.** *CC is closed under complementation.*

5 The theory VCC^* contains VNL

Each instance of the REACHABILITY problem consists of a directed acyclic graph $G = (V, E)$, where $V = \{u_0, \dots, u_{n-1}\}$, and we want to decide if there is a path from u_0 to u_{n-1} . It is well-known that REACHABILITY is NL-complete. Since a directed graph can be converted into a layered graph with

an equivalent reachability problem, it suffices to give a comparator circuit construction that solves instances of REACHABILITY satisfying the following assumption:

$$\text{The graph } G \text{ only has directed edges of the form } (u_i, u_j), \text{ where } i < j. \quad (5.1)$$

We believe that our new construction for showing that $NL \subseteq CC$ is more intuitive than the one in [8, 6]. Moreover, we reduce REACHABILITY to CCV directly without going through some intermediate complete problem, and this was stated as an open problem in [8, Chapter 7.8.1].

We will demonstrate our construction through a simple example, where we have the directed graph in Fig. 9 satisfying the assumption (5.1). We will build a comparator circuit as in Fig. 10, where the wires v_0, \dots, v_4 represent the vertices u_0, \dots, u_4 of the preceding graph and the wires l_0, \dots, l_4 are used to feed 1-bits into the wire v_0 , and from there to the other wires v_i reachable from v_0 . We let every wire l_i take input 1 and every wire v_i take input 0.

We next show how to construct the gadgets in the boxes. For a graph with n vertices ($n = 5$ in our example), the k^{th} gadget is constructed as follows:

- 1: Introduce a comparator gate from wire l_k to wire v_0
- 2: **for** $i = 0, \dots, n - 1$ **do**
- 3: **for** $j = i + 1, \dots, n - 1$ **do**
- 4: Introduce a comparator gate from v_i to v_j if $(u_i, u_j) \in E$,
or a dummy gate on v_i otherwise.
- 5: **end for**
- 6: **end for**

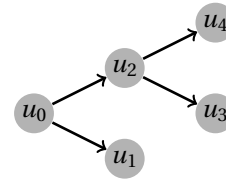


Figure 9

Note that the gadgets are identical except for the first comparator gate.

We only use the loop structure to clarify the order the gates are added. The construction can easily be done in AC^0 since the position of each gate can be calculated exactly, and thus all gates can be added independently from one another. Note that for a graph with n vertices, we have at most n vertices reachable from a single vertex, and thus we need n gadgets described above. In our example, there are at most 5 wires reachable from wire v_0 , and thus we utilize the gadget 5 times.

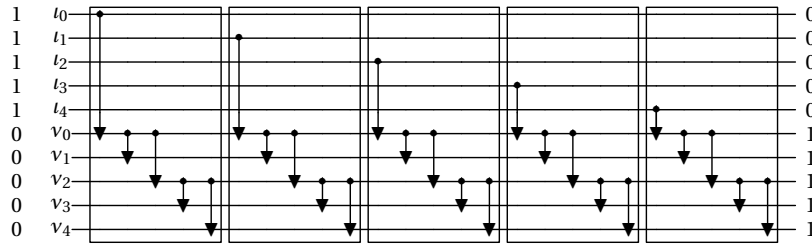


Figure 10 A comparator circuit that solves REACHABILITY. (The dummy gates are omitted.)

Intuitively, the construction works since each gadget from a box looks for the *lexicographical first maximal path* starting from v_0 (with respect to the *natural lexicographical ordering* induced by the vertex ordering v_0, \dots, v_n), and then the vertex at the end of the path will be marked (i.e. its wire will now carry 1) and thus excluded from the search of the gadgets that follow. For example, the gadget from the left-most dashed box in Fig. 10 will move a value 1 from wire l_0 to wire v_0 and from wire v_0 to wire v_1 . This essentially “marks” the wire v_1 since we cannot move this value 1 away from v_1 , and thus v_1 can no longer receive any new incoming 1. Hence, the gadget from the second box in Fig. 10 will repeat the process of finding the lex-first maximal path from v_0 to the remaining (unmarked) vertices. These searches end when all vertices reachable from v_0 are

marked. Note that this has the same effect as applying the *depth-first search* algorithm to find all the vertices reachable from v_0 .

► **Theorem 15** ($\text{VNL} \subseteq \text{VCC}^*$). *The theory VCC^* proves the axiom CONN defined in (2.4).*

Proof. Recall that if G is a directed graph on n vertices u_0, \dots, u_{n-1} with the edge relation E , then the formula $\delta_{\text{CONN}}(n, E, U)$ holds iff U is a matrix of n rows, where row d encodes the set of all vertices in G that are reachable from u_0 using a path of length at most d .

We start by converting G into a layered graph $G' = (V', E')$ which satisfies (5.1), where

$$\begin{aligned} V' &= \{u_i^\ell \mid 0 \leq i, \ell < n\}, \\ E' &= \{(u_i^\ell, u_j^{\ell+1}) \mid 0 \leq i, j, \ell < n \text{ and } (i = j \text{ or } (u_i, u_j) \in E)\}. \end{aligned} \quad (5.2)$$

Observe that a vertex u_i^ℓ is reachable from u_0^0 in G' iff u_i is reachable from u_0 by a path of length at most ℓ in G . Moreover, if we enumerate the vertices of G' by layers, then G' satisfies the condition (5.1). We now apply the above construction to G' to find all vertices in V' reachable from u_0^0 . Then we construct a matrix U witnessing the formula $\delta_{\text{CONN}}(n, E, U)$ by letting the row d encode the set of vertices u_i in V such that u_i^d is reachable from u_0^0 in G' .

We want to show that the comparator circuit constructed for G' using the above method produces the correct set of vertices u_i^d reachable from u_0^0 for every d . Although the correctness of the construction follows from the intuition that the circuit simulates the depth-first search algorithm, we cannot formalize this intuition directly since it would require VNL reasoning. Recall that up to this point, we only know that $\text{VTC}^0 \subseteq \text{VNC}^1 \subseteq \text{VCC}^*$ by Theorem 7. It turns out that using only the counting ability of VTC^0 , we can analyze the computation of the circuit in the above construction and argue that since we feed in as many 1-bits as the number of wires representing the vertices of G' , these 1-bits will eventually fill all the lower wires that are reachable from v_0^0 .

Now we begin the detailed proof. (To follow the proof it will be helpful to keep an eye on Fig. 10.) Let C be the comparator circuit constructed from the layered graph G' of G using the above construction, where C consists of the wires $\{v_i \mid 0 \leq i, \ell < n^2\}$ representing the vertices of G' and wires $\{t_0, \dots, t_{n^2}\}$ are used to feed ones to the wire v_0 . It is important to note that we will order the wires from high to low by layers of G' so that the sequence of wires

$$v_0, \dots, v_{n^2-1}$$

corresponds to the sequence of nodes

$$u_0^0, u_1^0, \dots, u_{n-1}^0, u_0^1, u_1^1, \dots, u_{n-1}^1, \dots, u_0^{n-1}, u_1^{n-1}, \dots, u_{n-1}^{n-1}.$$

Let $\gamma_0, \gamma_1, \dots, \gamma_{n^2-1}$ be the successive gadgets in the circuit C .

► **Lemma 16.** ($\text{VCC}^* \vdash$) *For each $k \leq n^2 - 1$, if wire v_0 has value 1 at the input of gadget γ_k then the value of each wire $v_i, 0 \leq i \leq n^2 - 1$, is the same at the output of γ_k as at the input of γ_k . If v_0 has value 0 at the input of γ_k then the above is true of γ_k with exactly one exception: some wire v_j is 0 at the input of γ_k and 1 at the output of γ_k .*

Proof. Note that for $k < n^2 - 1$ the output values of γ_k are the same as the input values of γ_{k+1} .

We proceed by induction on k . For $k = 0$ the input values of wires (v_0, \dots, v_{n^2-1}) are 0, but after the first gate (t_0, v_0) the value of wire v_0 becomes 1. Hence by Corollary 9 applied to γ_0 starting after that gate (or by induction on the depth of the gates in γ_0), the output values of wires (v_0, \dots, v_{n^2-1}) contain a single 1, and the rest are 0.

For the induction step suppose $k > 0$. Suppose first that the value of v_0 at the input of γ_k is 1, so the output value of v_0 for γ_{k-1} is 1. Then it follows from the induction hypothesis that the

tuple of values for wires (v_0, \dots, v_{n^2-1}) is either the same for the input and output of γ_{k-1} or wire v_0 is the only exception (it must be an exception if its input value is 0 because by assumption its output value is 1). Note that after the first gate (ι_0, v_0) in γ_{k-1} the value of wire v_0 is certainly 1 and so at this point in the gadget γ_{k-1} the tuple of values for wires (v_0, \dots, v_{n^2-1}) is the same as at this point in γ_k . Since gadgets γ_{k-1} and γ_k are identical after the first gate, the outputs for wires (v_0, \dots, v_{n^2-1}) are the same for the two gadgets. Thus the lemma follows for this case.

Now suppose that the value of v_0 at the input of γ_k (and hence the output of γ_{k-1}) is 0. Then by the induction hypothesis the value of v_0 at the input of γ_{k-1} is also 0 and the values of (v_0, \dots, v_{n^2-1}) at the inputs of γ_{k-1} and γ_k are identical except some wire v_j ($j \neq 0$) is 0 for γ_{k-1} and 1 for γ_k . After the first gate in each gadget the value of v_0 is 1, and since the gadgets are identical except for the first gate, it follows by induction on p that the value of each wire at position p in γ_{k-1} is the same as the value of that wire at position p in γ_k , except for each p one wire is 0 in γ_{k-1} and 1 in γ_k . The lemma follows when p is the output position. ◀

The next result follows easily from the preceding lemma by induction on k .

► **Lemma 17.** (VCC* \vdash) *If a wire v_i has value 1 at the output of some gadget γ_k , then it has value 1 at the outputs of all succeeding gadgets. If the tuple of output values for (v_0, \dots, v_{n^2-1}) is the same for two successive gadgets γ_k and γ_{k+1} then $v_0 = 1$ and the tuple remains the same for the outputs of all succeeding gadgets.*

The next result is the only place in the proof of Theorem 15 that makes essential use of the ability of VCC* to count, as in Corollary 9.

► **Lemma 18.** (VCC* \vdash) *Wire v_0 has value 1 at the output of the circuit C .*

Proof. By Lemma 17, if v_0 has value 0 at the output of C , then it has value 0 at the output of every gadget γ_k . Since there are n^2 ones at the input to C (one for every wire ι_i), by Corollary 9 there are n^2 ones at the output of C . But if v_0 remains 0 after every gadget, then by the construction of C we see that the final output of every wire ι_i is 0, and hence outputs of all n^2 wires v_i must be 1, including v_0 . ◀

► **Lemma 19.** (VCC* \vdash) *In the final gadget, no wire v_i changes its value after any comparator gate, except possibly v_0 changes from 0 to 1 after the initial gate feeds a 1 from ι_{n^2-1} .*

Proof. We use induction on i . For $i = 0$, by Lemma 18 the output of v_0 in the final gadget is 1. At the input to this gadget v_0 can be either 0 or 1, but after the first gate it is certainly 1. The value of v_0 cannot change from 1 to 0 in the gadget, because after the first gate there is no further gate leading down to v_0 which could bring down a 1 to change the value of v_0 from 0 to 1, but we know the final output value is 1.

For $i > 0$ it follows from Lemmas 16 and 18 that the value of wire v_i is the same at the input and output of the final gadget. We note that all gates leading down to v_i precede all gates leading away from v_i . The only way that v_i can change from 0 to 1 is at a gate bringing down a 1 from above, but that would change the value of the wire above, violating the induction hypothesis. The only way that v_i can change from 1 to 0 is at a gate leading away from v_0 , but then v_i cannot change back to 1, so the input and output cannot be the same. ◀

► **Lemma 20.** (VCC* \vdash) *If a wire v_i has value 1 at some position p in some gadget γ_k , then the output of v_i in the final gadget is 1.*

Proof. By Lemma 16 the tuple of input values for wires v_0, \dots, v_{n^2-1} is the same for gadgets γ_k and γ_{k+1} except possibly some input changes from 0 to 1. From this and the fact that the gadgets γ_k and γ_{k+1} are the same except for the first gate, it is easy to prove by induction on p that at position in p each wire has the same value in γ_{k+1} as in γ_k , except the value might be 0 in γ_k and 1 in γ_{k+1} .

Therefore by induction on k , if some wire v_i has value 1 in position p in some gadget then wire v_i has value 1 in position p in the final gadget. In this case, by Lemma 19 the output of v_i in the final gadget is 1. ◀

► **Lemma 21.** ($VCC^* \vdash$) *Let $j > 0$ and let y be the node in G' corresponding to wire v_j . Then the output of v_j in C is 1 iff there is $i < j$ such that the output of wire v_i in C is 1 and there is an edge from x to y , where x is the node in G' which corresponds to v_i .*

Proof. For the direction (\Leftarrow), suppose that the output of wire v_i is 1 and there is an edge from x to y in G' . Then there is a gate from v_i to v_j in the final gadget. Then the value of v_j must be 1 by Lemma 19, since otherwise this gate would change v_i from 1 to 0. Since the value of v_j cannot change, its value is 1 at the output.

For the direction (\Rightarrow) suppose the value of v_j is 1 at the output. Since the value of v_j at the input to C is 0, there is some gadget γ_k such that the value of v_j changes from 0 to 1. For this to happen there must be some comparator gate in γ_k from some wire v_i down to v_j with $i < j$ such that the value of v_i before the gate is 1, and there is an edge from the node x corresponding to v_i to node y . By Lemma 20 the value of v_i at the output of the final gadget is 1. ◀

To complete the proof of Theorem 15 recall the meaning of the array U given in (2.5), and the relation between the graph $G = (V, E)$ and the graph $G' = (V', E')$ given by (5.2). We define $U(d, i)$ (the truth value of node u_i of G in row d of the array U) to be true iff the the output of wire v_j in circuit C is 1, where v_j is the wire corresponding to node u_i^d in G' .

We prove that this definition of U satisfies the formula $\delta_{\text{CONN}}(n, E, U)$ (2.5) (which appears in the axiom (2.4) for VNL) by induction on d . The base case is $d = 0$. We have $U(0, 0)$ holds because the output of v_0 (the wire corresponding to node u_0^0) is 1 by Lemma 18. For $i > 0$, $U(0, i)$ is false because there is no edge in G' leading to node u_i^0 , and hence there is no comparator gate in any gadget in C leading down to the wire corresponding to u_i^0 , so that wire has output 0.

The induction step follows directly from our definition of U above, together with (2.5) and Lemma 21. ◀

As a of consequence of Theorem 15, we have the following result.

► **Theorem 22.** CC^* is closed under many-one NL-reductions, and hence $CC^{\text{Subr}} \subseteq CC^*$.

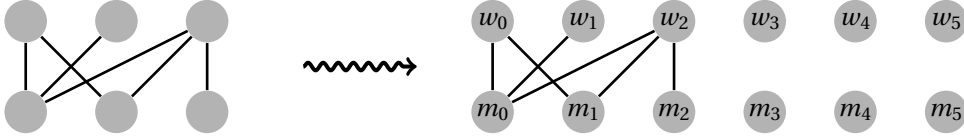
Proof. This follows from the following three facts: The function class FCC^* is closed under composition, $FNL \subseteq FCC$, and a decision problem is in CC^* iff its characteristic function is in FCC^* . ◀

6 The SM problem is CC -complete

6.1 $3LFMM$ is AC^0 -many-one-reducible to SM , $MoSM$ and $WoSM$

We start by showing that $3LFMM$ is AC^0 -many-one-reducible to SM in the second sense of Definition 2; i.e. we regard both $3LFMM$ and SM as search problems. (Of course the lfm-matching is the unique solution to $3LFMM$ formulated as a search problem,, but it is still a total search problem.)

Let $G = (V, W, E)$ be a bipartite graph from an instance of 3LFMM, where V is the set of bottom nodes, W is the set of top nodes, and E is the edge relation such that the degree of each node is at most three (see the example in the figure on the left below). Without loss of generality, we can assume that $|V| = |W| = n$. To reduce it to an instance of SM, we double the number of nodes in each partition, where the new nodes are enumerated after the original nodes and the original nodes are enumerated using the ordering of the original bipartite graph, as shown in the diagram on the right below. We also let the bottom nodes and top nodes represent the men and women respectively.



It remains to define a preference list for each person in this SM instance. The preference list of each man m_i , who represents a bottom node in the original graph, starts with all the woman w_j (at most three of them) adjacent to m_i in the order that these women are enumerated, followed by all the women w_n, \dots, w_{2n-1} ; the list ends with all women w_j not adjacent to m_i also in the order that they are enumerated. For example, the preference list of m_2 in our example is $w_2, w_3, w_4, w_5, w_0, w_1$. The preference list of each newly introduced man m_{n+i} simply consists of $w_0, \dots, w_{n-1}, w_n, \dots, w_{2n-1}$, i.e., in the order that the top nodes are listed. Preference lists for the women are defined dually.

Intuitively, the preference lists are constructed so that any stable marriage (not necessarily man-optimal) of the new SM instance must contain the lfm-matching of G . Furthermore, if a bottom node u from the original graph is not matched to any top node in the lfm-matching of G , then the man m_i representing u will marry some top node w_{n+j} , which is a dummy node that does not correspond to any node of G .

Formally, let \mathcal{S} be an instance of SM constructed from a bipartite graph $G = (V, W, E)$ using the above construction, where the set of men is $\{m_i\}_{i=0}^{2n-1}$ and the set of women is $\{w_i\}_{i=0}^{2n-1}$, and the preference lists are defined as above. For convenience, assume that the set of bottom nodes and top nodes of G are $V = \{m_i\}_{i=0}^{n-1}$ and $W = \{w_i\}_{i=0}^{n-1}$ respectively; the set of newly added bottom nodes and top nodes are $V' = \{m_i\}_{i=n}^{2n-1}$ and $W' = \{w_i\}_{i=n}^{2n-1}$ respectively. We will encode the edge relation of G by a Boolean matrix $E_{n \times n}$, where $E(i, j) = 1$ iff m_i is adjacent to w_j in G . Similarly, we encode the lfm-matching of G by Boolean matrix $L_{n \times n}$. We encode a stable marriage by a Boolean matrix $M_{2n \times 2n}$, and thus $M(i, j) = 1$ iff m_i marries w_j in M . We first prove the following lemma.

► **Lemma 23.** ($VCC^* \vdash$) *Given a stable marriage M , if $M(i, j) = 1$ for some $i, j < n$, then $E(i, j) = 1$.*

Proof. We prove by contradiction. Suppose $M(i, j) = 1$ for some $i, j < n$, but $E(i, j) = 0$, then since M is a perfect matching, by the pigeonhole principle $\text{PHP}(n-1, M)$, which is provable in VTC^0 (and hence in VCC^*), we cannot map the set of n men V' into the set of $n-1$ women W . Thus, there must exist some $p \geq n$ and $q \geq n$ such that $M(p, q) = 1$. Since m_i prefers w_q to w_j and w_j prefers m_q to m_i , M is not stable; hence a contradiction. ◀

► **Lemma 24.** ($VCC^* \vdash$) *Let M be a stable marriage of the SM instance \mathcal{S} reduced from the graph G , and let L be the lfm-matching of G . Then we have $L = (\{0, \dots, n-1\} \times \{0, \dots, n-1\}) \cap M$, where here L and M are treated as relations.*

Proof. First, we will show that L is contained in $(\{0, \dots, n-1\} \times \{0, \dots, n-1\}) \cap M$. Suppose otherwise, then there must exist a pair (i, j) , called "bad pair", $i, j < n$, such that $L(i, j) = 1$ but $M(i, k) = 1$ for some $k \neq j$. Using the Σ_0^B -MIN principle, we pick the "bad pair" (i, j) with minimum man index. There are two cases:

1. If $k < j$, then we cannot have $L(h, k) = 1$ for any $h < i$ for otherwise the pair (h, k) is a “bad pair” with a smaller man index than (i, j) , which is a contradiction. Therefore, $L(h, k) = 0$ for any $h < i$. By Lemma 23, we have $E(i, k) = 1$. Note that for any $\ell < k$, we have $L(i, \ell) = 0$, and furthermore, by the property of lfm-matching, for any $\ell < k$, either $E(i, \ell) = 0$ or there exists some $i' < i$ such that $L(i', \ell) = 1$. Therefore, $L(i, k) = 1$ by Eq. (4.1), which contradicts the fact that $L(i, j) = 1$.
2. Otherwise $k > j$, then we cannot have $M(h, j) = 1$ for any $h > i$ for otherwise m_i prefers w_j to w_k and w_j prefers m_i to m_h , which implies that M is not stable. Therefore, $M(h, j) = 1$ for some $h < i$. By Lemma 23, we have $E(h, j) = 1$. Note that for any $\ell < j$, $L(h, \ell) = 0$, for otherwise the pair (h, ℓ) is a “bad pair” with a smaller man index than (i, j) , which is a contradiction. Furthermore, by the property of lfm-matching, for any $\ell < j$, either $E(h, \ell) = 0$ or there exists some $i' < i$ such that $L(i', \ell) = 1$. Since for any $p < h$, we have $L(p, j) = 0$, by Eq. (4.1), we have $L(h, j) = 1$, which contradicts the fact that $L(i, j) = 1$.

Next, it remains to show that L cannot be strictly contained in $(\{0, \dots, n-1\} \times \{0, \dots, n-1\}) \cap M$. Suppose otherwise, let (i, j) , $i, j < n$, be a pair such that $M(i, j) = 1$ but for all $k < n$ and for all $\ell < n$, we have $L(i, k) = 0$ and $L(\ell, j) = 0$. By Lemma 23, we have $E(i, j) = 1$. Furthermore, by the property of lfm-matching, for any $\ell < j$, either $E(i, \ell) = 0$ or there exists some $i' < i$ such that $L(i', \ell) = 1$. By Eq. (4.1), we have $L(i, j) = 1$, which is a contradiction. ◀

Since Lemma 24 directly implies the correctness of the many-one AC^0 -reduction from 3LFMM to SM as search problems, any solution of a stable marriage instance constructed by the above reduction provides us all the information to decide if an edge is in the lfm-matching of the original 3LFMM instance. The key explanation is that every instance of stable marriage produced by the above reduction has a unique solution; thus the man-optimal solution coincides with the woman-optimal solution. Further Lemma 24 also shows that the decision version of 3LFMM is AC^0 -many-one-reducible to either of the decision problems MOSM and WOSM. Hence we have proven the following theorem.

► **Theorem 25.** $(VCC^* \vdash)$ 3LFMM is AC^0 -many-one-reducible to SM, MOSM and WOSM.

6.2 MOSM and WOSM are AC^0 -many-one-reducible to CCV

In this section, we formalize a reduction from SM to CCV due to Subramanian [8, 9]. Subramanian did not reduce SM to CCV directly, but to the *network stability problem* built from the less standard X gate, which takes two inputs p and q and produces two outputs $p' = p \wedge \neg q$ and $q' = \neg p \wedge q$. It is important to note that the “network” notion in Subramanian’s work denotes a generalization of circuits by allowing a connection from the output of a gate to the input of any gate including itself, and thus a network in his definition might contain cycles. An X-network is a network consisting only of X gates under the important restriction that each X gate has fan-out exactly one for each output it computes. The network stability problem for X gate (XNS) is then to decide if an X-network has a stable configuration, i.e., a way to assign Boolean values to the wires of the network so that the values are compatible with all the X gates of the network. Subramanian showed in his dissertation [8] that SM, XNS and CCV are all equivalent under log space reductions.

We do not work with XNS in this paper since networks are less intuitive and do not have a nice graphical representation as do comparator circuits. By utilizing Subramanian’s idea, we give a direct AC^0 -reduction from SM to CCV. For this goal, it turns out to be conceptually simpler to go through a new variant of CCV, where the comparator gates are three-valued instead of Boolean.

6.2.1 THREE-VALUED CCV is CC-complete

We define the THREE-VALUED CCV problem similarly to CCV, i.e., we want to decide, on a given input assignment, if a designated wire of a comparator circuit outputs one. The only difference is that each wire can now take either value 0, 1 or *, where a wire takes value * when its value is not known to be 0 or 1. The output values of the comparator gate on two input values p and q will be defined as follows.

$$p \wedge q = \begin{cases} 0 & \text{if } p = 0 \text{ or } q = 0 \\ 1 & \text{if } p = q = 1 \\ * & \text{otherwise.} \end{cases} \quad p \vee q = \begin{cases} 0 & \text{if } p = q = 0 \\ 1 & \text{if } p = 1 \text{ or } q = 1 \\ * & \text{otherwise.} \end{cases}$$

Every instance of CCV is also an instance of THREE-VALUED CCV. We will show that every instance of THREE-VALUED CCV is AC^0 -many-one-reducible to an instance of CCV by using a pair of Boolean wires to represent each three-valued wire and adding comparator gates appropriately to simulate three-valued comparator gates.

► **Theorem 26.** $(VCC^* \vdash)$ THREE-VALUED CCV and CCV are equivalent under many-one AC^0 -reductions.

Proof. Since each instance of CCV is a special case of THREE-VALUED CCV, it only remains to show that every instance of THREE-VALUED CCV is AC^0 -many-one-reducible to an instance of CCV.

First, we will describe a gadget built from standard comparator gates that simulates a three-valued comparator gate as follows. Each wire of an instance of THREE-VALUED CCV will be represented by a pair of wires in an instance of CCV. Each three-valued comparator gate on the left below, where $p, q, p \wedge q, p \vee q \in \{0, 1, *\}$, can be simulated by a gadget with two standard comparator gates on the right below.



The wires x and y are represented using the two pairs of wires $\langle x_1, x_2 \rangle$ and $\langle y_1, y_2 \rangle$, and three possible values 0, 1 and * will be encoded by $\langle 0, 0 \rangle$, $\langle 1, 1 \rangle$, and $\langle 0, 1 \rangle$ respectively. The fact that our gadget correctly simulates the three-valued comparator gate is shown in the following table.

p	q	$\langle p_1, p_2 \rangle$	$\langle q_1, q_2 \rangle$	$p \wedge q$	$p \vee q$	$\langle p_1 \wedge q_1, p_2 \wedge q_2 \rangle$	$\langle p_1 \vee q_1, p_2 \vee q_2 \rangle$
0	0	$\langle 0, 0 \rangle$	$\langle 0, 0 \rangle$	0	0	$\langle 0, 0 \rangle$	$\langle 0, 0 \rangle$
0	1	$\langle 0, 0 \rangle$	$\langle 1, 1 \rangle$	0	1	$\langle 0, 0 \rangle$	$\langle 1, 1 \rangle$
0	*	$\langle 0, 0 \rangle$	$\langle 0, 1 \rangle$	0	*	$\langle 0, 0 \rangle$	$\langle 0, 1 \rangle$
1	0	$\langle 1, 1 \rangle$	$\langle 0, 0 \rangle$	0	1	$\langle 0, 0 \rangle$	$\langle 1, 1 \rangle$
1	1	$\langle 1, 1 \rangle$	$\langle 1, 1 \rangle$	1	1	$\langle 1, 1 \rangle$	$\langle 1, 1 \rangle$
1	*	$\langle 1, 1 \rangle$	$\langle 0, 1 \rangle$	*	1	$\langle 0, 1 \rangle$	$\langle 1, 1 \rangle$
*	0	$\langle 0, 1 \rangle$	$\langle 0, 0 \rangle$	0	*	$\langle 0, 0 \rangle$	$\langle 0, 1 \rangle$
*	1	$\langle 0, 1 \rangle$	$\langle 1, 1 \rangle$	*	1	$\langle 0, 1 \rangle$	$\langle 1, 1 \rangle$
*	*	$\langle 0, 1 \rangle$	$\langle 0, 1 \rangle$	*	*	$\langle 0, 1 \rangle$	$\langle 0, 1 \rangle$

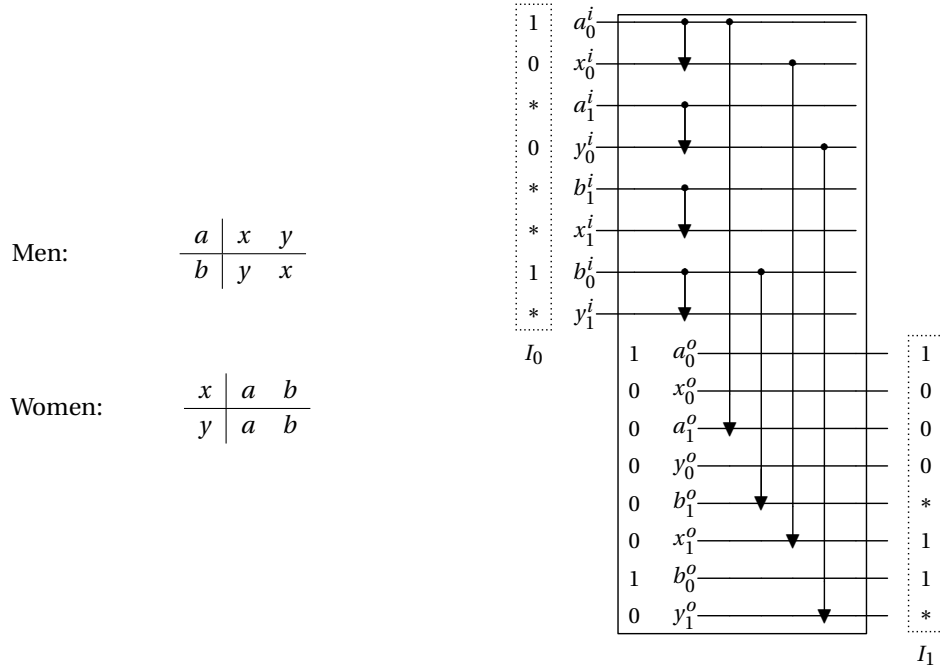
Using this gadget, we can reduce an instance of THREE-VALUED CCV to an instance of CCV by doubling the number of wires, and for every three-valued comparator gate of the THREE-VALUED CCV instance, we will add a gadget with two standard comparator gates simulating it.

The above construction shows how to reduce the question of whether a designated wire outputs 1 for a given instance of THREE-VALUED CCV to the question of whether a *pair* of wires of an

instance of CCV outputs $\langle 1, 1 \rangle$. However for an instance of CCV we are only allowed to decide whether a *single* designated wire outputs 1. This technical difficulty can be easily overcome since we can use an \wedge -gate (one of the two outputs of a comparator gate) to test whether a pair of wires outputs $\langle 1, 1 \rangle$, and outputs the result on a single designated wire. ◀

6.2.2 A fixed-point method for solving stable marriage problems

We formalize a method for solving SM using three-valued comparator circuits based on [8, 9]. Consider an instance \mathcal{S} of SM consisting of n men and n women and preference lists for each man and woman. From this instance we construct a three-valued comparator circuit $C_{\mathcal{S}}$. Fig. 11 illustrates $C_{\mathcal{S}}$ when \mathcal{S} consists of two men a, b and two women x, y with preference lists given by the matrices.



■ Figure 11

For each man m and woman w in \mathcal{S} and each pair j, k with $j, k < n$ we say $\text{Pair}(m_j, w_k)$ holds iff w is at the j^{th} position of m 's preference list and m is at the k^{th} position of w 's preference list. For each such pair there are two consecutive input wires of $C_{\mathcal{S}}$ labelled m_j^i and w_k^i respectively. (Here the superscript i stands for 'input'.) Hence there are n^2 pairs of input wires, making a total of $2n^2$ input wires.

In addition there are $2n^2$ other wires called 'output wires' labelled in the same order as above; two consecutive wires with labels m_j^o and w_k^o for each pair satisfying $\text{Pair}(m_j, w_k)$. These output wires have fixed input values: We let output wire m_0^o take input one for every man m , and let the rest of output wires have zero inputs.

The circuit $C_{\mathcal{S}}$ has the following comparator gates. For each pair (m_j^i, w_k^i) of consecutive inputs there is a gate from wire m_j^i to w_k^i . After these gates, for every person p , we add a gate from wire p_j^i to p_{j+1}^o for every $j < n - 1$. Note that the order of this last group of wires does not matter. (See Fig. 11.)

Given the instance \mathcal{S} of SM with n men and n women, define $\mathcal{M} : \{0, 1, *\}^{2n^2} \rightarrow \{0, 1, *\}^{2n^2}$ to

be the function computed by the preceding circuit construction, where the inputs of \mathcal{M} are those fed into the input wires, and the outputs of \mathcal{M} are those produced by the output wires. We will use the following notation. Any sequence $I \in \{0, 1, *\}^{2n^2}$ can be seen as an input of function \mathcal{M} , and thus we write $I(p_j^i)$ to denote the input value of wire p_j^i with respect to I . Similarly, if a sequence $J \in \{0, 1, *\}^{2n^2}$ is an output of \mathcal{M} , then we write $J(p_j^o)$ to denote the output value of wire p_j^o .

Let sequence $I_0 \in \{0, 1, *\}^{2n^2}$ be an input of \mathcal{M} defined as follows: $I_0(m_0^i) = 1$ for every man m , and $I_0(w_0^i) = 0$ for every woman w , and $I_0(p_j^i) = *$ for every person p and every j , $1 \leq j < n$. Note that the number of $*$'s in the sequence I_0 is

$$c(n) = 2n^2 - 2n. \quad (6.1)$$

Our version of Subramanian's method [8, 9] consists of computing

$$I_{c(n)} = \mathcal{M}^{c(n)}(I_0),$$

where \mathcal{M}^d simply denotes the d^{th} power of \mathcal{M} , i.e. the function we get by composing \mathcal{M} with itself d times. It turns out that $I_{c(n)}$ is a fixed point of \mathcal{M} , i.e. $I_{c(n)} = \mathcal{M}(I_{c(n)})$. To show this, we define a sequence I' to be an *extension* of a sequence I if $I(p) = I'(p)$ for every person p such that $I(p) \in \{0, 1\}$. In other words, all Boolean values in I are preserved in J , even though some $*$ -values in I might be changed to Boolean values in J . We can show that $\mathcal{M}(I)$ is an extension of I for every I which extends I_0 , and hence $\mathcal{M}^d(I_0)$ extends I_0 for all d . It follows that $\mathcal{M}^{c(n)}(I_0)$ is a fixed point because there are at most $c(n)$ $*$'s to convert to 0 or 1.

Now we can extract a stable marriage from the fixed point $I_{c(n)}$ by letting B be the sequence obtained by substituting zeros for all remaining $*$ -values in $I_{c(n)}$. Then B is also a fixed point of \mathcal{M} . A stable marriage can then be extracted from B by announcing the marriage of a man m and a woman w iff $\text{Pair}(m_j, w_k)$ and $B(m_j^o) = 1$ and $B(w_k^o) = 0$. Our goal is to formalize the correctness of this method.

In the example in Fig. 11, the computation of the fourth power of the function \mathcal{M} can easily be done by using the comparator circuit in Fig. 12. We show the outputs, produced as a result of the iteration, which comprise the fixed point $I_4 = \mathcal{M}^4(I_0)$ of \mathcal{M} . In this case the fixed point consists of Boolean values, where $(I_4(a_0^o), I_4(x_0^o)) = (1, 0)$ and $(I_4(b_0^o), I_4(y_1^o)) = (1, 0)$. Thus, woman x is married to man a , and woman y is married to man b , which is a stable marriage.

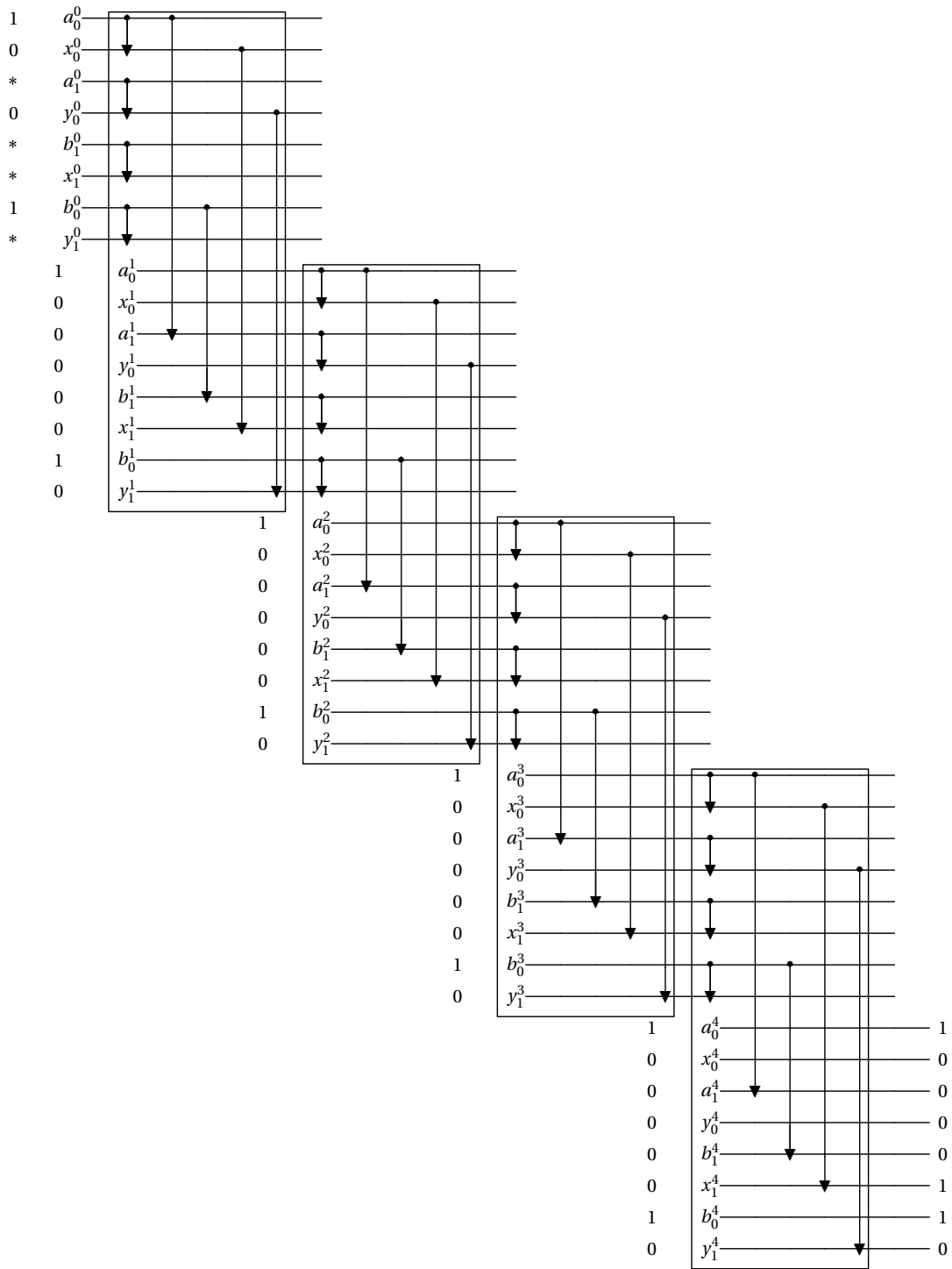
Henceforth, we will let I_ℓ denote the output of $\mathcal{M}^\ell(I_0)$. We want to show that the $c(n)^{\text{th}}$ power of $\mathcal{M} : \{0, 1, *\}^{2n^2} \rightarrow \{0, 1, *\}^{2n^2}$ on input I_0 defined above in fact produces a fixed point of \mathcal{M} .

► **Theorem 27.** ($\text{VCC}^* \vdash$) *The three-valued sequence $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$ is a fixed point of \mathcal{M} .*

To prove this theorem, we need a new definition. The proof of Theorem 27 follows from the important observation that every sequence I_q is extended by all sequences I_ℓ , $\ell \geq q$. And thus when going from I_ℓ to $I_{\ell+1}$, the only change that can be made is to change some $*$ -values in I_ℓ to Boolean values in $I_{\ell+1}$. But since we started out with at most $c(n)$ $*$ -values, we will reach a fixed point in at most $c(n)$ steps. Before proving Theorem 27, we need the following lemma, which says that any given sequence I_q is extended by all sequences I_ℓ , $\ell \geq q$. The lemma can be formulated as the following Σ_0^B statement.

► **Lemma 28.** ($\text{VCC}^* \vdash$) *For every $q \leq c(n) + 1$, for every person p , and for every $k < n$, if we have $I_q(p_k^i) = v \in \{0, 1\}$, then $I_\ell(p_k^i) = v$ for all ℓ satisfying $q \leq \ell \leq c(n) + 1$.*

Proof of Lemma 28. We prove by Σ_0^B induction on $q \leq c(n) + 1$. The base case ($q = 0$) is easy. With respect to I_0 the only wires having non- $*$ values are p_0^i for every person p . But the output



■ **Figure 12** The comparator circuit computing the fourth power of \mathcal{M} constructed from the example in Fig. 11. Since the output wires of the first three blocks serve as input wires to the next block, we use superscripts 1,2,3,4 for these wires on successive blocks, instead of the letter 'o' used in Fig. 11.

wires p_0^o corresponding to these input values are fed these same Boolean values as constant inputs, and these wires are not involved with any comparator gates. Thus, these values will be preserved for every I_ℓ with $q \leq \ell \leq c(n) + 1$.

For the induction step, we are given q such that $0 < q \leq c(n) + 1$, and assume that $I_q(p_k^i) = v \in \{0, 1\}$ for some person p and $k < n$, we want to show that $I_\ell(p_k^i) = v$ for all ℓ satisfying $q \leq \ell \leq c(n) + 1$. We will only argue for the case when p is a man m since the case when p is a woman can be argued similarly. We consider two cases. We may have $k = 0$, in which case, as argued in the base case, we have $I_\ell(m_0^i) = 1$ for all ℓ satisfying $q \leq \ell \leq c(n) + 1$. Otherwise, we have $k \geq 1$, then since $I_q(m_k^o) = v \in \{0, 1\}$, from how \mathcal{M} was constructed, the output wire p_k^o must have got its non-* value v from the wire m_{k-1}^i , which in turn must carry value v before transferring it to wire m_k^o . But then we observe that wire m_{k-1}^i is connected to some wire w_r^i by a comparator gate (i.e. $\text{Pair}(m_{k-1}, w_r)$ holds) before being connected by a comparator gate to m_k^o . Thus, from the definition of three-valued comparator gate, the value v produced on m_{k-1}^i by the gate $\langle m_{k-1}^i, w_r^i \rangle$ only depends on the non-* value(s) of either $I_{q-1}(m_{k-1}^i)$ or $I_{q-1}(w_r^i)$ or both. In any of these cases, by the induction hypothesis, these non-* values of I_{q-1} will be preserved in I_ℓ for all ℓ with $q-1 \leq \ell \leq c(n) + 1$. Hence, we will always get $I_\ell(m_k^o) = v$ for all ℓ satisfying $q \leq \ell \leq c(n) + 1$. ◀

Proof of Theorem 27. Suppose for a contradiction that for every $\ell \leq c(n)$, I_ℓ is not a fixed point. In other words, $I_{\ell+1} = \mathcal{M}(I_\ell) \neq I_\ell$ for all $\ell \leq c(n)$. It is important to note that, by Lemma 28, when going from I_ℓ to $I_{\ell+1}$, we know that $I_{\ell+1}$ extends I_ℓ . Thus, the only change that \mathcal{M} can make at each stage is to switch the *-values at some positions in I_ℓ to Boolean values in $I_{\ell+1}$. If $I_{c(n)}$ is not a fixed point, then by utilizing the counting in VTC^0 (Corollary 8), we can show that the number of *-values that are switched to Boolean values when going from I_0 to $I_{c(n)+1}$ is at least $c(n) + 1$. This is a contradiction, since we started out with only $c(n)$ *-values in I_0 , and no additional *-value was supplied during the iterations of \mathcal{M} . The final argument, i.e., the number of *-values never increases, can be proved more formally by Σ_0^B -induction on the layers of the comparator circuit computing $\mathcal{M}^{c(n)}(I_0)$. ◀

Although Theorem 27 gives us a fixed point of \mathcal{M} , this fixed point may still be three-valued and thus does not give us all the information needed to extract a stable marriage. However, every three-valued fixed point can easily be extended to a Boolean fixed point as follows. Given a three-valued sequence I , we let $I[* \rightarrow v]$ denote the sequence we get by substituting v for all the *-values in I .

► **Proposition 2.** ($\text{VCC}^* \vdash$) *If I is a three-valued fixed point of \mathcal{M} , then $I[* \rightarrow 0]$ and $I[* \rightarrow 1]$ are Boolean fixed points of \mathcal{M} .*

Proof. Suppose that I is a three-valued fixed point of \mathcal{M} . Then when the circuit $C_{\mathcal{G}}$ is presented with input I the output is also I . Without studying the detailed structure of the circuit but just observing that the gates compute monotone functions, by induction on the depth of a gate g in the circuit we can compare the values v and v' of g under the two inputs I and $I[* \rightarrow 0]$ as follows: $v' = v$ if $v \in \{0, 1\}$ and $v' = 0$ if $v = *$. In particular this is true of the output gates of the circuit. Since I is a fixed point, the output of $C_{\mathcal{G}}$ under input I is I , and so the output under input $I[* \rightarrow 0]$ is $I[* \rightarrow 0]$. Thus $I[* \rightarrow 0]$ is a fixed point of the circuit.

A similar argument works for $I[* \rightarrow 1]$ ◀

To show that the above method for solving SM using three-valued comparator circuits is correct, it remains to justify Subramanian's method for extracting a stable marriage from a Boolean fixed point. Define G to be an AC^0 -function, i.e., Σ_0^B -definable, which takes as input a Boolean fixed point B of \mathcal{M} , and returns a marriage M such that the pair of man m and woman w is in M iff when j, k are chosen such that $\text{Pair}(m_j, w_k)$ holds, then $B(m_j^o) = 1$ and $B(w_k^o) = 0$. It is worth

noting that since $B = \mathcal{M}(B)$, we have $B(p_k^i) = B(p_k^o)$ for every person p and every $k < n$; however, the superscripts o and i are useful for distinguishing between input and output values of the comparator circuit $C_{\mathcal{G}}$ computing \mathcal{M} . From the construction of G and the fixed-point property of \mathcal{M} , we have the following theorem.

► **Theorem 29.** ($\text{VCC}^* \vdash$) *If B is a Boolean fixed point of \mathcal{M} then $M = G(B)$ is a stable marriage.*

To prove this theorem, we first need to establish the next two lemmas that capture the basic properties of the comparator circuit computing \mathcal{M} .

► **Lemma 30.** ($\text{VCC}^* \vdash$) *Let B be any Boolean input to the circuit $C_{\mathcal{G}}$.*

1. *For every man m and every $k < n - 1$, if $B(m_k^i) = 1$ then $B(m_{k+1}^o) = 0$ iff $B(w_j^i) = 0$, where w_j^i is the wire that satisfies $\text{Pair}(m_k, w_j)$.*
2. *For every woman w and every $j < n - 1$, if $B(w_j^i) = 0$ then $B(w_{j+1}^o) = 1$ iff $B(m_k^i) = 1$, where m_k^i is the wire that satisfies $\text{Pair}(m_k, w_j)$.*

Proof. We will only prove Part 1 since Part 2 can be shown using a dual argument. For the (\Leftarrow) direction, we recall that since m_k^i was paired with w_j^i when constructing $C_{\mathcal{G}}$, we have a comparator gate going from m_k^i to w_j^i . Thus, since $B(m_k^i) = 1$ and $B(w_j^i) = 0$, after the comparator gate $\langle m_k^i, w_j^i \rangle$, the wire m_k^i now carries value zero. But since the output wire $B(m_{k+1}^o)$ will carry whatever value forwarded from the wire m_k^i , in this case, we have $B(m_{k+1}^o) = 0$.

For the (\Rightarrow) direction, from the construction of $C_{\mathcal{G}}$, the only way that we can change from $B(m_k^i) = 1$ to $B(m_{k+1}^o) = 0$ is by having a comparator gate $\langle m_k^i, w_j^i \rangle$ connecting m_k^i with some wire w_j^i , and $B(w_j^i) = 0$. ◀

► **Lemma 31.** ($\text{VCC}^* \vdash$) *Let B be any Boolean fixed point for \mathcal{M} .*

1. *For every man m and every $k < n$, if $B(m_k^i) = 1$ and $B(m_{k+1}^i) = 0$, then*

$$B(m_0^i) = \dots = B(m_k^i) = 1, \quad B(m_{k+1}^i) = \dots = B(m_n^i) = 0.$$

2. *For every woman w and every $j < n$, if $B(w_j^i) = 0$ and $B(m_{j+1}^o) = 1$, then*

$$B(w_0^i) = \dots = B(w_j^i) = 0, \quad B(w_{j+1}^i) = \dots = B(w_n^i) = 1.$$

Proof. We will only prove Part 1 since Part 2 can be proved by a dual argument. Assume that m is a man and $k < n$ is such that $B(m_k^i) = 1$ and $B(m_{k+1}^i) = 0$. We will use Σ_0^B -MIN to choose the least $k_0 \geq 0$ satisfying $B(m_{k_0}^i) = 1$ and $B(m_{k_0+1}^i) = 0$. We can then prove by Σ_0^B induction on ℓ , $k_0 + 1 \leq \ell < n$, that $B(m_\ell^i) = 0$. The base case when $\ell = k_0 + 1$ trivially holds. For the induction step, by the construction of $C_{\mathcal{G}}$, we observe that when $B(m_{\ell-1}^i) = 0$, then the wire $m_{\ell-1}^i$ will always carry value zero. But since m_ℓ^o will receive whatever value forwarded to it from $m_{\ell-1}^i$, we get $B(m_\ell^i) = B(m_\ell^o) = 0$. Thus, we have just shown that

$$B(m_0^i) = \dots = B(m_{k_0}^i) = 1, \quad B(m_{k_0+1}^i) = \dots = B(m_{n-1}^i) = 0.$$

But this implies that k_0 is the only subscript at which the elements of the sequence

$$B(m_0^i), B(m_2^i), \dots, B(m_{n-1}^i)$$

change their values from one to zero. Thus, we get $k = k_0$, and we are done. ◀

From the above two lemmas, we can show that using G we can extract from every Boolean fixed point of \mathcal{M} which extends I_0 a perfect matching.

► **Lemma 32.** ($\text{VCC}^* \vdash$) *If B is a Boolean fixed point of \mathcal{M} then $M = G(B)$ is a perfect matching between the men and women of \mathcal{F} .*

Proof. We will only prove that every man is married to a unique woman in M since the claim that every woman is married to a unique man can be shown similarly. Given a man m we want to show that he is married to a unique woman. Since B is a fixed point we know $B(m_0^i) = B(m_0^o) = 1$, and by Lemma 31 the elements of the sequence

$$B(m_0^i), B(m_1^i), \dots, B(m_{n-1}^i)$$

can only change their values from one to zero at most once. Thus by Lemma 30 and the definition of M , m can marry at most once. It remains for us to show that m indeed gets married. Suppose to the contrary that m remains single in M . Then

$$B(m_0^i) = B(m_1^i) = \dots = B(m_{n-1}^i) = 1.$$

For every woman w we can choose $k, j < n$ so that $\text{Pair}(m_k, w_j)$ holds, and so by Lemma 30 it follows that $B(w_j^i) = 1$. But $B(w_0^i) = B(w_0^o) = 0$, so the elements of the sequence

$$B(w_0^i), B(w_1^i), \dots, B(w_{n-1}^i)$$

must change their values from zero to one at least once, and by Lemma 31 they change their values exactly once. Thus by Lemma 30 and the definition of M , every woman is married to exactly one man. Since m was excluded, we have n women paired with at most $n - 1$ men, and this contradicts the pigeonhole principle $\text{PHP}(n - 1, M)$. ◀

Proof of Theorem 29. By Lemma 32, we know that M is a perfect matching. Thus it only remains to show that M satisfies the stability condition. Suppose not. Then there exist men a, b and women x, y such that x is married to a and y is married to b , but man a prefers y to x and woman y prefers a to b . Since x is married to a , by how M was constructed and Lemma 31, there are some $k < n$ and $p < n$ such that $\text{Pair}(a_k, x_p)$, and

$$B(a_0^i) = B(a_1^i) = \dots = B(a_k^i) = 1. \quad (6.2)$$

Similarly, since y is married to b , there are some $\ell < n$ and $q < n$ such that $\text{Pair}(b_\ell, y_q)$, and

$$B(y_0^i) = B(y_1^i) = \dots = B(y_q^i) = 0. \quad (6.3)$$

Now by the definition of Pair there must be some $s, t < n$ such that $\text{Pair}(a_s, y_t)$ holds. But since man a prefers y to x , and woman y prefers a to b , we have $s < k$ and $t < q$. Thus from (6.2) and (6.3), we get $B(a_s^i) = 1$ and $B(y_t^i) = 0$ respectively. Hence, y is also married to a , and this contradicts Lemma 32. ◀

Fix a stable marriage instance \mathcal{S} with n men and n women, and let \mathcal{M} be the function computed by the comparator circuit $C_{\mathcal{S}}$. Let Φ_{sm} denote the set of all stable marriages of \mathcal{S} , and let Φ_{fxp} denote the set of all Boolean fixed points of \mathcal{M} which extend the input I_0 defined from \mathcal{S} . Note that Φ_{sm} and Φ_{fxp} are exponentially large sets, so they are not really objects of our theories. In other words, we write $M \in \Phi_{\text{sm}}$ to denote that M satisfies a formula asserting the stable marriage property, and we write $I \in \Phi_{\text{fxp}}$ to denote that I satisfies a formula asserting that I is a fixed point of \mathcal{M} . It was proved in [9] that there is a one-to-one correspondence between Φ_{sm} and Φ_{fxp} , and that the matchings extracted from $I_{C(n)}[* \rightarrow 0]$ and $I_{C(n)}[* \rightarrow 1]$ are man-optimal and woman-optimal respectively. We now show how to formalize these results.

We define $F : \Phi_{\text{sm}} \rightarrow \Phi_{\text{fxp}}$ to be a function that takes as input a stable marriage M of \mathcal{S} , and outputs a sequence $I \in \{0, 1\}^{2n^2}$ defined as follows. For every man m and every woman w that are matched in M , if $j, k < n$ are subscripts such that $\text{Pair}(m_j, w_k)$ holds, then we assign

$$I(m_0^i) = \dots = I(m_j^i) = 1 \text{ and } I(m_{j+1}^i) = \dots = I(m_{n-1}^i) = 0 \quad (6.4)$$

$$I(w_0^i) = \dots = I(w_k^i) = 0 \text{ and } I(w_{k+1}^i) = \dots = I(w_{n-1}^i) = 1. \quad (6.5)$$

From this definition of F , we can prove the following lemma.

► **Lemma 33.** ($\text{VCC}^* \vdash$) *The function $F : \Phi_{\text{sm}} \rightarrow \Phi_{\text{fxp}}$ is a bijection.*

We first need to verify that the range of F is indeed contained in Φ_{fxp} .

► **Lemma 34.** ($\text{VCC}^* \vdash$) *If M is a stable marriage, then $I = F(M)$ is a fixed point of \mathcal{M} .*

Proof. We start by stating the following:

Claim: For every pair of wires (m_j^i, w_k^i) satisfying $\text{Pair}(m_j, w_k)$, we have $(I(m_j^i), I(w_k^i)) = (1, 0)$ iff man m is matched to woman w in M .

To see that I is a fixed point of \mathcal{M} it suffices to show that equations (6.4) and (6.5) hold with the superscripts i replaced by o . This follows from the Claim and Lemma 30 with $B = I$, together with the observation that for every man m and woman w , the circuit $C_{\mathcal{S}}$ assigns the outputs $m_0^o = 1$ and $w_0^o = 0$.

It remains to prove the Claim. The direction (\Leftarrow) follows immediately from the definition of I . To prove the direction (\Rightarrow), suppose that for some man m and woman v , $\text{Pair}(m_\ell, v_s)$ holds for some $\ell, s < n$ and $(I(m_\ell^i), I(v_s^i)) = (1, 0)$ but m is not matched to v . Then m is matched to some other woman w , and $\text{Pair}(m_j, w_k)$ holds for some $j, k < n$. Since $I(m_\ell^i) = 1$, it follows from (6.4) that $\ell < j$, so m prefers v to w . Since $I(v_s^i) = 0$, it follows from (6.5) applied to the woman v that v prefers m to the man that she is matched with. Therefore the marriage is not stable. ◀

Proof of Lemma 33. From Lemma 34, we know that the function F is properly defined. It also follows from how F was defined that two distinct stable marriages will get mapped to distinct fixed points of \mathcal{M} , and hence F is injective. It only remains to show that F is surjective. But then it is not hard to check that the function G defined before Theorem 29 is a left inverse of F . ◀

We next want to show that $G(I_{c(n)}[* \rightarrow 0])$ and $G(I_{c(n)}[* \rightarrow 1])$ are man-optimal and woman-optimal stable marriages of \mathcal{S} respectively. A technical difficulty is that it might be tricky to compare $G(I_{c(n)}[* \rightarrow 0])$ and $G(I_{c(n)}[* \rightarrow 1])$ with $G(J)$ for some arbitrary Boolean fixed point J of \mathcal{M} . However the following lemma shows that every Boolean fixed point of \mathcal{M} is an extension of $I_{c(n)}$, which means that it suffices to work with only Boolean fixed-point extensions of $I_{c(n)}$.

► **Lemma 35.** ($\text{VCC}^* \vdash$) *If J is a Boolean fixed point of \mathcal{M} , then J extends I_ℓ for every $\ell \leq c(n)$.*

Proof. We prove by Σ_0^B induction on $\ell \leq c(n)$. Base case ($\ell = 0$): we have $I_0(m_0^o) = 1$ for every man m and $I_0(w_0^o) = 0$ for every woman w . But from how \mathcal{M} was constructed, \mathcal{M} always outputs value one on wire m_0^o for every man m and zero on wire w_0^o for every woman w . Thus since J is a Boolean fixed point of \mathcal{M} , we also have $J(m_0^o) = 1$ for every man m and $J(w_0^o) = 0$ for every woman w , and hence J extends I_0 .

For the induction step, we are given ℓ such that $0 < \ell \leq c(n)$, and assume that J extends I_ℓ . We want to show that for every person p and $k < n$, if $I_\ell(p_k^i) = v \in \{0, 1\}$, then $J(p_k^i) = v$. We will only argue for the case when p is a man m since the case when p is a woman can be argued similarly. We consider two cases. We may have $k = 0$, then we can argue as in the base case. Otherwise, we

have $k \geq 1$, then since $I_\ell(m_k^o) = v \in \{0, 1\}$, from how \mathcal{M} was constructed, the output wire p_k^o must have got its non-* value v from the wire m_{k-1}^i , which in turn must have carried value v before transferring it to wire m_k^o . But then we observe that wire m_{k-1}^i is connected to some wire w_r^i by a comparator gate (i.e. $\text{Pair}(m_{k-1}, w_r)$ holds) before being connected by a comparator gate to m_k^o . Thus from the definition of three-valued comparator gate, the value v produced on m_{k-1}^i by the gate $\langle m_{k-1}^i, w_r^i \rangle$ only depends on the non-* value(s) of either $I_{m-1}(m_{k-1}^i)$ or $I_{q-1}(w_r^i)$ or both. In any of these cases, since J extends $I_{\ell-1}$ (by the induction hypothesis), these non-* values of I_{q-1} will also be contained J . But since $J = \mathcal{M}(J)$, we get $J(m_k^o) = v$. ◀

From Lemma 33 and Lemma 35, we can prove the following theorem.

► **Theorem 36.** ($\text{VCC}^* \vdash$) *Let M be a stable marriage of the SM instance \mathcal{I} . Let*

$$M_0 = G(I_{c(n)}[* \rightarrow 0])$$

$$M_1 = G(I_{c(n)}[* \rightarrow 1]).$$

Then M_0 and M_1 are stable marriages, and every man gets a partner in M_0 no worse than the one he gets in M , and every woman gets a partner in M_1 no worse than the one she gets in M . In other words, M_0 and M_1 are the man-optimal and woman-optimal solutions respectively.

Proof. We only prove that M_0 is man-optimal since the proof that M_1 is woman-optimal is similar. From Lemma 33 and Lemma 35, if we let $K = F(M)$, then K is a Boolean fixed point of \mathcal{M} extending the three-valued fixed point $I_{c(n)}$ and K uniquely determines M . Suppose for a contradiction that some man m gets a better partner in M than in M_0 . Let w and u be the women m marries in M and M_0 , and assume that $j, k, \ell, s < n$ are subscripts such that $\text{Pair}(m_j, w_k)$ and $\text{Pair}(m_\ell, u_s)$ hold. For brevity, let $O = I_{c(n)}[* \rightarrow 0]$. Then from how M and M_0 are constructed, we have $(K(m_j^i), K(w_k^i)) = (1, 0)$ and $(O(m_\ell^i), O(u_s^i)) = (1, 0)$. Note that we construct O by substituting zeros for all *-values in $I_{c(n)}$, so we must have $I_{c(n)}(m_\ell^i) = 1$ originally. By Lemma 31, we have $O(m_0^i) = \dots = O(m_\ell^i) = 1$. But since m prefers w to u , we also have $j < \ell$, and hence $O(m_j^i) = 1$. Since we cannot introduce additional ones to $I_{c(n)}$ to get O , we also have $I_{c(n)}(m_j^i) = 1$. We next show the following claim, which will imply a contradiction since m cannot marry both w and u in the stable marriage M_0 .

Claim: We must have $O(w_k^i) = 0$.

We cannot have $(I_{c(n)}(m_j^i), I_{c(n)}(w_k^i)) = (1, 0)$; otherwise, m has no choice but to marry w in both M and M_0 since both K and O are extensions of $I_{c(n)}$. This forces $I_{c(n)}(w_k^i) = *$. But then since we must substitute zeros for all *-values when producing O , we have $O(w_k^i) = 0$. ◀

► **Theorem 37.** ($\text{VCC}^* \vdash$) *MoSM and WoSM are AC^0 -many-one-reducible to $\text{CCV}\neg$.*

Proof. We will show only the reduction from MoSM to $\text{CCV}\neg$ since the reduction from WoSM to $\text{CCV}\neg$ works similarly.

Following the above construction, we can write a Σ_0^B -formula defining an AC^0 function that takes as input an instance of MoSM with preference lists for all the men and women, and produces a three-valued comparator circuit that computes the three-valued fixed point $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$, and then extracts the man-optimal stable marriage from $I_{c(n)}[* \rightarrow 0]$. Although the first step of computing $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$ can easily be done as shown in the example from Fig. 12, the second step of computing $I_{c(n)}[* \rightarrow 0]$ from the output $I_{c(n)}$ and extracting the stable marriage cannot be trivially done using a three-valued comparator circuit. However, we can apply the construction from the proof of Theorem 26 to simulate the three-valued computation of $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$ using an instance of $\text{CCV}\neg$, where we can then utilize the available negation gates, \wedge -gates and \vee -gates

to build the necessary gadget to decide if a designated pair of man and woman are married in the man-optimal stable marriage. The use of negation gates is essential in our construction.

Let \mathcal{S} be an instance of MOSM, where (m, w) is the designated pair of man and woman. Let M denote the man-optimal stable marriage of \mathcal{S} . We choose j, k such that $\text{Pair}(m_j, w_k)$ holds. Then we recall that $(m, w) \in M$ iff $I_{c(n)}[* \rightarrow 0](m_j^o) = 1$ and $I_{c(n)}[* \rightarrow 0](w_k^o) = 0$. Observe that $I_{c(n)}[* \rightarrow 0](m_j^o) = 1$ and $I_{c(n)}[* \rightarrow 0](w_k^o) = 0$ iff

$$(I_{c(n)}(m_j^o), I_{c(n)}(w_k^o)) = (1, 0) \vee (I_{c(n)}(m_j^o), I_{c(n)}(w_k^o)) = (1, *). \quad (6.6)$$

Let C denote the three-valued circuit computing $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$. Then (6.6) simply asserts that the wire carrying $I_{c(n)}(m_j^o)$ of C must output 1 and the wire carrying $I_{c(n)}(w_k^o)$ of C must output either 0 or *. Let C' be the boolean comparator circuit that simulates the three-valued computation of C using the construction from the proof of Theorem 26, where now we use a pair of wires in C' to simulate each three-valued wire of C . From (6.6) it suffices to check that $I_{c(n)}(m_j^o)$ is coded by $\langle 1, 1 \rangle$ and $I_{c(n)}(w_k^o)$ has first component 0 in its code (the two possibilities are $\langle 0, 0 \rangle$ and $\langle 0, 1 \rangle$). This checking is easily done with comparator gates, together with a negation gate to verify the 0 output. ◀

Corollary 13 and Theorems 26, 25 and 37 give us the following corollary.

► **Corollary 38.** ($VCC^* \vdash$) *The ten problems MOSM, WOSM, SM, CCV, CCV \neg , THREE-VALUED CCV, 3LFMM, LFMM, 3VLFMM and VLFMM are all equivalent under many-one AC^0 -reductions, where the equivalence of SM is with respect to the search problem version of the reduction defined in Definition 2.*

Proof. Corollary 13 and Theorem 26 show that CCV, CCV \neg , THREE-VALUED CCV, 3LFMM, LFMM, 3VLFMM and VLFMM are all equivalent under many-one AC^0 -reductions.

Theorem 37 shows that MOSM and WOSM are AC^0 -many-one-reducible to THREE-VALUED CCV. Theorem 25 also shows that 3LFMM is AC^0 -many-one-reducible to MOSM, WOSM, and SM. Hence, MOSM, WOSM, and SM is equivalent to the above problems under many-one AC^0 -reductions. ◀

7 Conclusion and future work

Our correctness proof of the reduction from SM to CCV is a nice example showing the utility of three-valued logic for reasoning about uncertainty. Since an instance of SM might not have a unique solution, the fact that the fixed point $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$ is three-valued indicates that the construction cannot fully determine how all the men and women can be matched. Thus, different Boolean fixed-point extensions of $I_{c(n)}$ give us different stable marriages.

It is worth noting that Subramanian's method is not the "textbook" method for solving SM. The most well-known is the Gale-Shapley algorithm [4]. In fact, our original motivation was to formalize the correctness of the Gale-Shapley algorithm, but we do not know how to talk about the computation of the Gale-Shapley algorithm in VCC^* due to the fan-out restriction in comparator circuits. Thus, we leave open the question whether VCC^* proves the correctness of the Gale-Shapley algorithm.

We believe that CC deserves more attention, since on the one hand it contains interesting complete problems, but on the other hand we have no real evidence (for example based on relativized inclusions) concerning whether CCV is complete for P, and if not, whether it is comparable to NC. The perfect matching problem (for bipartite graphs or general undirected graphs) shares these same open questions with CCV. However several randomized NC^2 algorithms are known for perfect matching [5, 7], but no randomized NC algorithm is known for any CC-complete problem.

Another open question is whether the three CCV complexity classes mentioned in (1.1) coincide, which is equivalent to asking whether CC (the closure of CCV under many-one AC^0 -reductions) is closed under oracle AC^0 -reductions, or equivalently whether the function class FCC is closed under composition. A possible way to show this would be to show the existence of universal comparator circuits, but we do not know whether such circuits exist.

The analogous question for standard complexity classes such as TC^0 , L , NL , NC , P has an affirmative answer. That is, each class can be defined as the AC^0 many-one closure of a complete problem, and the result turns out to be also closed under AC^0 oracle reducibilities. (A possible exception is the function class $\#L$, whose AC^0 oracle closure is the $\#L$ hierarchy [1]. This contains the integer determinant as a complete problem.)

References

- 1 E. Allender and M. Ogihara. Relationships among PL , $\#L$, and the determinant. *RAIRO, Theoretical Informatics and Applications*, 30(1):1–21, 1996.
- 2 K.E. Batchier. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computer Conference 32*, pages 307–314. ACM, 1968.
- 3 S. Cook and P. Nguyen. *Logical foundations of proof complexity*. Cambridge University Press, 2010.
- 4 D. Gale and L.S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.
- 5 R.M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random NC . *Combinatorica*, 6(1):35–48, 1986.
- 6 E.W. Mayr and A. Subramanian. The complexity of circuit value and network stability. *Journal of Computer and System Sciences*, 44(2):302–323, 1992.
- 7 K. Mulmuley, U.V. Vazirani, and V.V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.
- 8 A. Subramanian. *The computational complexity of the circuit value and network stability problems*. PhD thesis, Dept. of Computer Science, Stanford University, 1990.
- 9 A. Subramanian. A new approach to stable matching problems. *SIAM Journal on Computing*, 23(4):671–700, 1994.