

A linear conjunctive grammar for the circuit value problem

Alexander Okhotin
okhotin@cs.queensu.ca

Technical report 2002-460
School of Computing, Queen's University,
Kingston, Ontario, Canada K7L 3N6

September 2002

Abstract

In this paper we show that the **P**-complete language of yes-instances of Circuit Value Problem under a certain supplied encoding can be generated by a linear conjunctive grammar.

Contents

1	Introduction	2
2	The problem and its encoding	3
3	Checking the syntax	6
4	Computing the value of a circuit	8
5	Putting things together	13
A	A complete grammar for CVP	19
B	All circuits of ≤ 4 gates evaluating to one	30

1 Introduction

Conjunctive grammars, introduced in [1], are context-free grammars augmented with an explicit intersection operation. Linear conjunctive grammars are a subclass of conjunctive grammars defined by the same kind of restriction imposed on the body of rules as used to obtain linear context-free grammars out of general context-free.

A linear conjunctive grammar is defined as a quadruple $G = (\Sigma, N, P, S)$, where Σ is the alphabet of terminal symbols, N is the set of nonterminal symbols, $S \in N$ is the start symbol, and the rules from P are of the form

$$A \rightarrow u_1 B_1 v_1 \& \dots \& u_n B_n v_n \quad (n \geq 1) \quad (1a)$$

$$A \rightarrow w, \quad (1b)$$

where A, B_i are nonterminal symbols and u_i, v_i, w are terminal strings. The conjunction has semantics of set-theoretic intersection of the languages generated by the individual strings. The language generated by a grammar is formally defined by a means of derivation, in which a nonterminal can be rewritten with a body of a rule enclosed in parentheses, and a conjunction of several identical terminal strings enclosed in parentheses can be rewritten with one such string without the parentheses. The language family generated by linear conjunctive grammars includes many nontrivial (and non-context-free) languages, such as $\{a^n b^n c^n \mid n \geq 0\}$, $\{wcw \mid w \in \{a, b\}^*\}$ [1], $\{ba^2 ba^4 b \dots ba^{2n-2} ba^{2n} b \mid n \geq 0\}$ $\{(wc)^{|w|-1} w \mid w \in \{a, b\}^*\}$ or the language of all computations of a given Turing machine [3].

Linear conjunctive grammars are known to be computationally equivalent to triangular trellis automata (also known as real-time one-sided cellular automata) [5], which we will define as a quintuple $M = (\Sigma, Q, I, \delta, F)$, where Σ is the input alphabet, Q is a finite nonempty set of states, $I : \Sigma \rightarrow Q$ is a homomorphism that maps each symbol of the input alphabet to a single state, and the string of states $I(w) \in \Sigma^+$ forms the initial configuration of the automaton on an input string $w \in \Sigma^+$, $\delta : Q \times Q \rightarrow Q$ converts a pair of states into a single state, and is used to transform a configuration

$$(q_1, q_2, q_3, \dots, q_{n-1}, q_n) \quad (2a)$$

to a configuration

$$(\delta(q_1, q_2), \delta(q_2, q_3), \dots, \delta(q_{n-1}, q_n)), \quad (2b)$$

which is being done until the string shrinks to one state, when the membership of this single state in F determines whether the input string is accepted. The result of the computation on a string of states $\alpha \in Q^+$ is denoted as $\Delta(\alpha)$, and consequently the language generated by a triangular trellis automaton $M = (\Sigma, Q, I, \delta, F)$ is

$$L(M) = \{w \mid w \in \Sigma^+, \Delta(I(w)) \in F\} \quad (3)$$

Some theoretical properties of linear conjunctive grammars have been worked out in [3, 5]. The membership problem for linear conjunctive grammars (as well as that for general conjunctive grammars) is known to be **P**-complete [2]. In this paper we obtain a somewhat related and quite surprising result that not only they have a **P**-complete membership problem, but some *P-complete languages* can in fact be *generated* by particular linear conjunctive grammar. This is being done by constructing a linear conjunctive grammar for Circuit Value Problem (CVP), the basic **P**-complete problem.

2 The problem and its encoding

Let $(C_1, \dots, C_n)$ ($n \geq 1$) be a circuit with inputs $x_1, \dots, x_m$ ($m \geq 1$), where each gate C_k is one of the following:

1. An input x_i ($1 \leq i \leq m$);
2. Negation of a preceding gate: $\neg C_i$ ($i < k$)
3. Conjunction of two preceding gates: $C_i \wedge C_j$ ($i < j < k$),
4. Disjunction of two preceding gates: $C_i \vee C_j$ ($i < j < k$).

The gate C_n is called the output of the circuit.

The Circuit Value Problem (CVP) is stated as follows: given a circuit and a Boolean vector of input values $(\sigma_1, \dots, \sigma_m)$, determine, whether the circuit evaluates to true on this vector. The pair *(circuit, vector of input values)* is called an instance of CVP.

Let us develop one particular encoding for instances of CVP as strings over a finite alphabet. We shall use the alphabet

$$\Sigma = \{a, b, c_r, c_\wedge, c_\vee, c_w, c_{w\neg}, c_{w0}, c_{w1}\}, \quad (4)$$

in which:

- the symbol a is used to specify the total number of gates in the circuit in unary notation;
- the symbol b is used to refer to the gates of the circuit by writing their numbers in unary notation;
- c_r is a read command;
- $c_\wedge$ and $c_\vee$ are commands for computing conjunction and disjunction respectively;
- c_w means “write”;
- $c_{w\neg}$ means “write negation”;
- c_{w0} and c_{w1} mean “write zero” and “write one” respectively.

Let $(\{x_1, \dots, x_m\}, \{C_1, \dots, C_n\})$ be an instance of circuit value problem. For each gate C_k , define its literal representation $\omega(C_k)$:

$$\omega(C_k = x_i) = \begin{cases} b^{k-1}c_{w0}, & \text{if } x_i = 0 \\ b^{k-1}c_{w1}, & \text{if } x_i = 1 \end{cases} \quad (5a)$$

$$\omega(C_k = C_i \wedge C_j) = b^{i-1}c_rb^{j-i-1}c_\wedge b^{k-j-1}c_w \quad (i < j < k) \quad (5b)$$

$$\omega(C_k = C_i \vee C_j) = b^{i-1}c_rb^{j-i-1}c_\vee b^{k-j-1}c_w \quad (i < j < k) \quad (5c)$$

$$\omega(C_k = \neg C_i) = b^{i-1}c_rb^{k-i-1}c_{w\neg} \quad (i < k) \quad (5d)$$

It should be mentioned that $|\omega(C_k)| = k$ for every k -th gate of any of the four types (5).

Finally, define the literal representation of the whole instance of the problem as

$$a^n \cdot \omega(C_1) \cdot \dots \cdot \omega(C_n) \in \Sigma^* \quad (6)$$

Example 1. Consider the circuit $\{C_1 = x_1, C_2 = x_2, C_3 = C_1 \wedge C_2, C_4 = \neg C_2, C_5 = C_3 \vee C_4\}$ depending on the variables x_1, x_2 , which is given in Figure 1. It can easily be checked that the circuit implements implication $f(x_1, x_2) = x_2 \rightarrow x_1$.

The literal representation of the gates C_3 , C_4 and C_5 does not depend on the input, and is $\omega(C_3 = C_1 \wedge C_2) = c_rc_\wedge c_w$, $\omega(C_4 = \neg C_2) = bc_rc_{w\neg}$, $\omega(C_5 = C_3 \vee C_4) = bbc_rc_\vee c_w$.

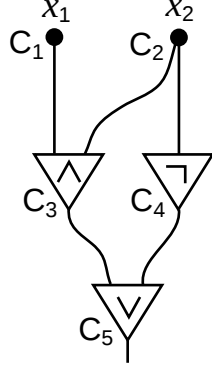


Figure 1: A sample circuit implementing $f(x_1, x_2) = x_2 \rightarrow x_1$.

For given values of x_1 and x_2 , the literal representation of the first two gates is $\omega(C_1 = x_1) = c_{w\sigma_1}$, $\omega(C_2 = x_2) = bc_{w\sigma_2}$, and consequently the whole instance of CVP will be represented as

$$aaaaa c_{w\sigma_1} bc_{w\sigma_2} c_r c_{\wedge} c_w bc_r bc_{w\neg} bbc_r c_{\vee} c_w \quad (7)$$

The encoding of CVP presented in this section is somewhat different from the traditional encoding: first, the numbers of gates are usually given in binary notation; second, the vector of input values is usually given once and input variables can be used more than once. While there is obviously no loss of generality in the assumption that each input value is explicitly used only once, it is necessary to justify that the problem remains **P**-complete under the encoding that uses unary notation to represent the numbers of gates.

It can be easily shown that there is no exponential blowup in the transition to the unary notation, because the numbers notated refer to the numbers of the gates, and thus are bounded by the size of the input; consequently, the descriptions of circuits using binary notation can be converted to unary notation with no more than quadratic blowup, and this conversion can be easily done in logarithmic space, which proves that the language of yes-instances of CVP under the given encoding is a **P**-complete language.

3 Checking the syntax

The first thing to do is to check whether the given string is syntactically correct – i.e., whether it represents an instance of CVP encoded using the method of Section 2. This task is already less trivial than could be expected, because even this language is non-context-free (indeed, if we suppose that it is context-free, then its homomorphic image under $h(\text{every symbol}) = a$, which equals $\{a^{\frac{n(n+3)}{2}} \mid n \geq 1\}$, is also context-free, which is known to be untrue).

We have to check that the string is of the form (6), or, equivalently, of the form

$$a^n u_1 u_2 \dots u_n, \quad (8)$$

where every u_k is a string of exactly k symbols of the form

$$b^* c_{w0}, \quad (9a)$$

$$b^* c_{w1}, \quad (9b)$$

$$b^* c_r b^* c_{w\neg}, \quad (9c)$$

$$b^* c_r b^* c_{\wedge} b^* c_w \quad \text{or} \quad (9d)$$

$$b^* c_r b^* c_{\vee} b^* c_w \quad (9e)$$

The last (and only the last) symbol of each u_k is one of $\{c_w, c_{w\neg}, c_{w0}, c_{w1}\}$ and is called *the write command of the gate*.

The set of valid literal representations of circuits can be defined inductively: a string $w \in \Sigma^*$ is a syntactically correct description of a circuit if and only if

- (Basis) w equals ac_{w0} or ac_{w1} .
- (Induction step) The string w can be factorized as

$$w = a^n uv \quad (n \geq 2; u, v \in (\Sigma \setminus \{a\})^+, |v| = n), \quad (10)$$

where u ends with a write command, v is of the form (9) and $a^{n-1}u \in \Sigma^*$ is a syntactically correct description of a circuit.

It is not hard to construct a linear conjunctive grammar that will generate the set of strings conforming to this definition:

4 Computing the value of a circuit

In this section we solve another problem: given a description of a circuit and assuming that it is syntactically valid, determine whether it evaluates to 1. It turns out that this task can be solved by a triangular trellis automaton; such an automaton will be constructed in this section.

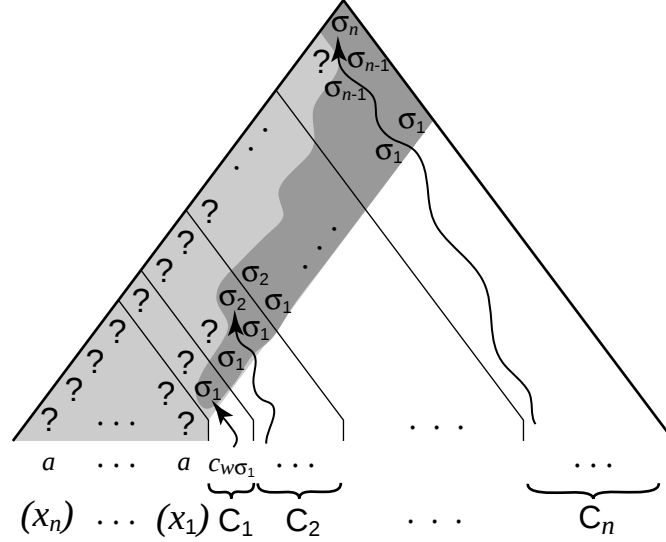


Figure 3: Overview of the computation on a circuit description $a^n \omega(C_1) \dots \omega(C_n)$.

Before presenting a formal construction of the automaton, let us explain the main idea of the construction. The information flow in the automaton's computation is outlined in Figure 3: the gray band formed by n diagonals starting from the prefix a^n of the whole input string of length $O(n^2)$ is used to store the computed values of the gates, and the description of every gate is used as a microprogram implementing a single assignment statement. The value of i -th gate of a circuit $w = a^n \omega(C_1) \dots \omega(C_n)$ is stored in the diagonal corresponding to the $(n-i+1)$ -th symbol a ; initially it is set to "?", and once the value of the gate is computed (exactly at the point $\Delta(I(a^i C_1 \dots C_i))$), the diagonal keeps reproducing this computed value (0 or 1) till the end of the computation; the consequent gates have a read-only access to this value.

Let us consider how the individual gates work, using a conjunction gate $C_k = C_i \wedge C_j$ ($i < j < k$) as an example. Figure 4 shows some middle point

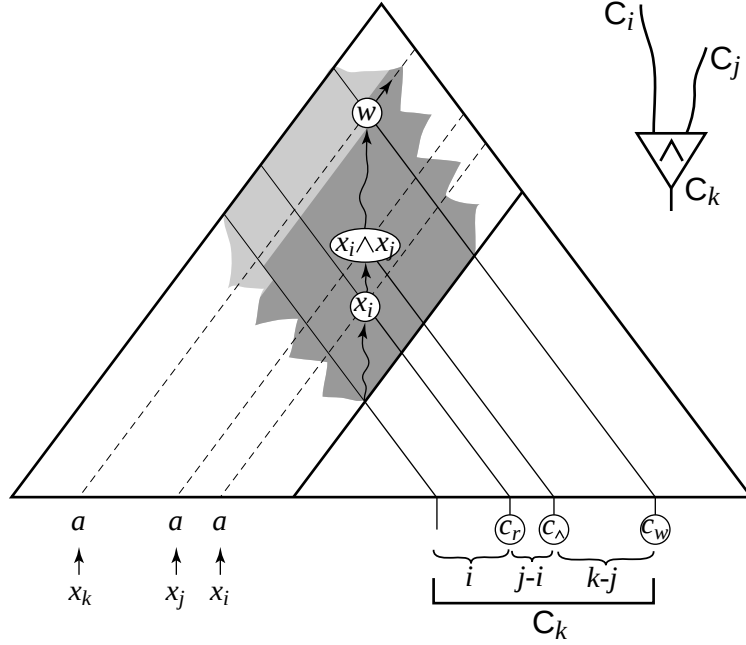


Figure 4: Operation of an individual conjunction gate $C_k = C_i \wedge C_j$.

of the gray band of circuit values, corresponding to the gate C_k . At this point the values of the gates $C_1, \dots, C_{k-1}$ have already been computed, and the area shown in the figure is expected to use these values to compute the value of C_k .

The literal representation of the gate C_k ,

$$b^{i-1}c_r b^{j-i-1}c_\wedge b^{k-j-1}c_w \quad (12)$$

could be, as mentioned above, viewed as a microprogram implementing the assignment statement $C_k = C_i \wedge C_j$. The substring $b^{i-1}c_r$, i symbols long, is used to find the value of the gate C_i in the gray band and to remember its value. The substring $b^{j-i-1}c_\wedge$ seeks the diagonal corresponding to the gate C_j , at the same time carrying the value of the gate C_i ; its last symbol $c_\wedge$, instructs the automaton to read the value of the gate C_j and to compute its conjunction with the remembered value of the gate C_i . This value is carried on by the substring $b^{k-j-1}c_w$, which finds the gate C_k , and replaces the question mark there with the computed value $C_i \wedge C_j$.

Let us now construct a triangular trellis automaton to perform this kind

of computation. Define $M = (\Sigma, Q, I, \delta, F)$, where the set of states Q is

$$\underbrace{\{?, 0, 1\}}_{\text{Digit}} \times (\underbrace{\{-, r, \vee, \wedge, w0, w1, w\neg, w\}}_{\text{Commands}} \cup \underbrace{\{\nwarrow, \nwarrow 0, \nwarrow 1, \nearrow, \nearrow 0, \nearrow 1\}}_{\text{Arrows}} \cup \{-\}) \quad (13)$$

The first component of every pair, *the digit*, can be 0 or 1 only in the diagonals forming the gray band in Figures 3 and 4, once the value of the corresponding gates is computed (shown in dark gray in these figures); all other cells have ? as the first component.

The second component can be *a command*, *an arrow* or *space*. The eight possible *commands* originate from the input symbols $b, c_r, c_\vee, c_\wedge, c_{w0}, c_{w1}, c_{w\neg}$ and c_w respectively; they are propagated in the left direction until executed, where execution results in conversion to an arrow or in alteration of a digit in the diagonal. The *arrows* organize the data flow necessary to execute commands: their intended direction is upward, but since triangular trellis automata do not support direct upward communication, it is being implemented by a pair of a left-up and a right-up arrow. An arrow can optionally carry some value, obtained by an earlier read command, a conjunction command or a disjunction command, and intended to be used by some later conjunction, disjunction, write or write negation command. The states with a *space* as a second component are called the *unmarked states* and do not have any task to perform besides holding the digit and propagating it to the right; these states can only appear in the diagonals corresponding to the values of the gates.

We shall use the following compact notation for the states from Q :

- The states of the form $(?, \text{command})$ will be denoted by the name of the command alone – i.e., $w\neg$ means $(?, w\neg)$.
- The states that have space as the second component will be denoted by the first component alone: $?, 0$ and 1 mean $(?, _)$, $(0, _)$ and $(1, _)$ respectively.
- The states of the form $(0, \text{command})$ and $(1, \text{command})$ will be denoted as the number with the name of the command as a subscript: $0_\wedge$ means $(0, \wedge)$ and 1_{w0} means $(1, w0)$.
- The states that have an arrow as the second component will be denoted with their second component as a superscript: $0^\nearrow$ and $1^{\nwarrow 0}$ mean $(0, \nearrow)$ and $(1, \nwarrow 0)$ respectively.

A state is called *an activator* if one of the following holds: (i) it is an unmarked state, (ii) it is marked with a right arrow, or (iii) it holds the digit 0 or 1, and it is marked with a write command ($w0$, $w1$, w or $w\neg$). The full list of activators is $\{?, 0, 1, 0_{w0}, 1_{w0}, 0_{w1}, 1_{w1}, 0_w, 1_w, 0_{w\neg}, 1_{w\neg}, ?\nearrow, 0\nearrow, 1\nearrow, ?\nearrow^0, 0\nearrow^0, 1\nearrow^0, ?\nearrow^1, 0\nearrow^1, 1\nearrow^1\}$.

The initial function $I : \Sigma \rightarrow Q$ maps a to “?”, b to “–” and every command $c_{..}$ to the state with the same name as the subscript of c (e.g., $I(c_{w0}) = w0$). The values of I are tabulated in Table 1.

input symbol	a	b	c_r	$c_\wedge$	$c_\vee$	c_w	$c_{w\neg}$	c_{w0}	c_{w1}
$I(\text{input symbol})$	?	–	r	$\wedge$	$\vee$	w	$w\neg$	$w0$	$w1$

Table 1: The initial function of the automaton.

Now let us specify the transition function of the automaton. For every $(q', q'') \in Q \times Q$, define $\delta(q', q'')$ as follows:

Reduplication of a digit If q' is either an unmarked state or is marked with a write command, while q'' is either unmarked or marked with a right arrow, then $\delta(q', q'')$ is an unmarked state that inherits the digit from q' .

Reduplication of a command If q' is not an activator and q'' is marked with some command, then $\delta(q', q'')$ is formed by the digit from q' and by the command from q'' .

Turning of an arrow If q'' is marked with a left arrow – i.e., the second component of q'' is one of $\{\nwarrow, \nearrow^0, \nwarrow^1\}$ – then the second component of $\delta(q', q'')$ is a right arrow loaded with the same cargo as the left arrow of q'' ; the digit in $\delta(q', q'')$ is inherited from q' .

Execution of a command If q' is an activator and q'' is marked with a command, then

- If q'' is marked with a write command ($w0$, $w1$, w or $w\neg$), then it can be proved that the digit in q' must be ?. Additionally, it can be proved that if q'' is marked with w or $w\neg$, then q' is marked with a right arrow carrying a digit.

The state $\delta(q', q'')$ is then defined as

1. $(0, \neg)$, if q'' is marked with $w0$;

2. $(1, \sqcup)$, if q'' is marked with $w1$;
3. (the digit on the arrow in $q', \sqcup$), if q'' is marked with w ;
4. (negation of the digit on the arrow in $q', \sqcup$), if q'' is marked with $w\neg$;

Note that this is the only case when the digit in $\delta(q', q'')$ differs from the digit in q' .

- If q'' is marked with a read command r , then $\delta(q', q'')$ retains the digit from q' and is marked with a left arrow carrying the same digit from q' – i.e., $\delta(q', q'')$ can be $0^{\nwarrow 0}$ or $1^{\nwarrow 1}$ depending on q' .
- If q'' is marked with a dash command (meaning “proceed to the next row”), then $\delta(q', q'')$ retains the digit from q' and
 - if q' is marked with a right arrow carrying some digit, then $\delta(q', q'')$ is marked with a left arrow carrying the same digit.
 - if q' is marked with something else, then $\delta(q', q'')$ is marked with a left arrow carrying nothing.
- If q'' is marked with a $\wedge$ or $\vee$ command, then it can be proved that q' is marked with a right arrow carrying a digit. $\delta(q', q'')$ retains the digit from q' and is marked with a left arrow carrying the digit computed by the command:
 - If q'' is marked with $\wedge$ command, then $\delta(q', q'')$ is marked with the conjunction of the first component of q' and the digit carried by the arrow in q' .
 - If q'' is marked with $\vee$ command, then $\delta(q', q'')$ is marked with the disjunction of the first component of q' and the digit carried by the arrow in q' .

The values of the transition function δ constructed in accordance with these rules are given in Tables 2 to 10 on pages 14 to 19 (the whole 45×45 table is too large to fit in a single page, and is therefore splitted into nine tables). The empty cells in these tables are the transitions left undefined; they can be set arbitrarily, because it can be proved that they are not accessible on any syntactically correct input.

The sole accepting state is $(1, \sqcup)$, or 1 in the short notation.

Consider the circuit from Example 1 in Section 2 (see Figure 1), and let $x_1 = 1$ and $x_2 = 0$. Then the corresponding instance of CVP, illustrated in

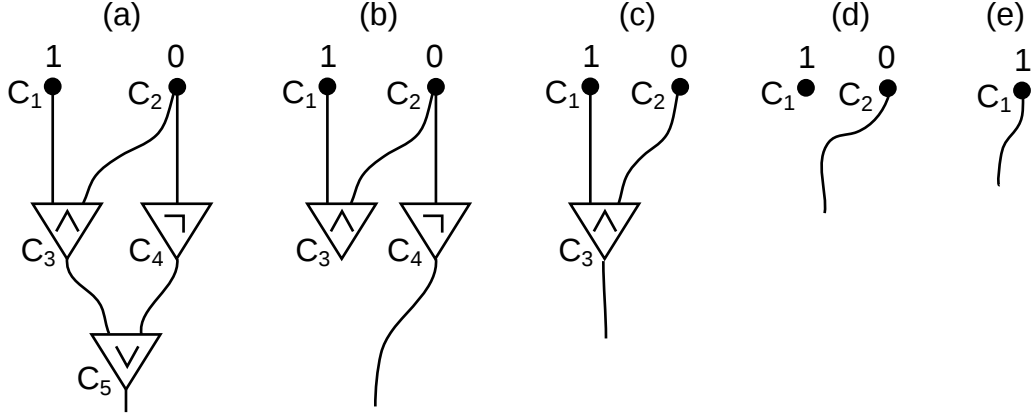


Figure 5: Sample circuits.

Figure 5(a) is encoded as

$$aaaaa\ c_{w1}\ bc_{w0}\ c_r c_{\wedge} c_w\ bc_r bc_{w\neg}\ bbc_r c_{\vee} c_w \quad (14a)$$

and the computation of the automaton on the string (14a) is given in Figure 6 on page 18; this is an accepting computation, because 1 is an accepting state. It is easily seen that the computations of the same automaton on the four substrings of (14a) that also denote circuits,

$$aaaa\ c_{w1}\ bc_{w0}\ c_r c_{\wedge} c_w\ bc_r bc_{w\neg} \quad (14b)$$

$$aaa\ c_{w1}\ bc_{w0}\ c_r c_{\wedge} c_w \quad (14c)$$

$$aa\ c_{w1}\ bc_{w0} \quad (14d)$$

$$a\ c_{w1} \quad (14e)$$

are included in the computation on (14a) (as in Figure 6) as “subcomputations”. It can be observed that the automaton accepts the circuit descriptions (14b) and (14e) (these circuits are illustrated in Figure 5(b,e)) and rejects (14c) and (14d), given in Figure 5(c,d).

5 Putting things together

Having constructed a linear conjunctive grammar for the language of syntactically valid descriptions of circuits (in Section 3) and a triangular trellis

	?	—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \searrow$	$? \searrow^0$	$? \searrow^1$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$
?	?	$? \searrow$	0	1	$? \searrow^1$	$? \searrow^1$	$? \searrow$	$0 \nearrow^0$	0	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
—		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
w0	?	—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
w1	?	—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
$\wedge$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$\vee$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
r		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
w	?	—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
$w\neg$	?	—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
$? \searrow$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \searrow^0$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \searrow^1$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \nearrow$		$? \searrow$	0	1	$? \searrow^1$	$? \searrow^1$	$? \searrow$	?	0	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \nearrow^0$		$? \searrow^0$	0	1	$? \searrow^0$	$? \searrow^1$	$? \searrow$	0	1	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \nearrow^1$		$? \searrow^1$	0	1	$? \searrow^1$	$? \searrow^1$	$? \searrow$	1	0	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			

Table 2: The transition function of the automaton, part 1 of 9 (? by ?).

	0	0 _—	0 _{w0}	0 _{w1}	0 _{$\wedge$}	0 _{$\vee$}	0 _r	0 _w	0 _{$w\neg$}	0 _{$? \searrow$}	0 _{$? \searrow^0$}	0 _{$? \searrow^1$}	0 _{$? \nearrow$}	0 _{$? \nearrow^0$}	0 _{$? \nearrow^1$}
?	?	$? \searrow$	0	1	$? \searrow^1$	$? \searrow^1$	$? \searrow$	$0 \nearrow^0$	0	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
—		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
w0	?	—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
w1	?	—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
$\wedge$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$\vee$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
r		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
w	?	—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
$w\neg$	?	—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$	?	?	?
$? \searrow$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \searrow^0$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \searrow^1$		—	w0	w1	$\wedge$	$\vee$	r	w	$w\neg$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \nearrow$		$? \searrow$	0	1	$? \searrow^1$	$? \searrow^1$	$? \searrow$	?	0	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \nearrow^0$		$? \searrow^0$	0	1	$? \searrow^0$	$? \searrow^1$	$? \searrow$	0	1	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			
$? \nearrow^1$		$? \searrow^1$	0	1	$? \searrow^1$	$? \searrow^1$	$? \searrow$	1	0	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$			

Table 3: The transition function of the automaton, part 2 of 9 (? by 0).

	1	1 _−	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↖	1 _{↖⁰}	1 _{↖¹}	1 _↗	1 _{↗⁰}	1 _{↗¹}
?	?	↖	0	1	↖ ¹	↖ ¹	↖	0↗ ⁰	0	↗	↗ ⁰	↗ ¹	?	?	?
−		−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹			
w0	?	−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹	?	?	?
w1	?	−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹	?	?	?
∧		−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹			
∨		−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹			
r		−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹			
w	?	−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹	?	?	?
w¬	?	−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹	?	?	?
↖		−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹			
↖ ⁰		−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹			
↖ ¹		−	w0	w1	∧	∨	r	w	w¬	↗	↗ ⁰	↗ ¹			
↗		↖	0	1	↖ ¹	↖ ¹	↖	?	0	↗	↗ ⁰	↗ ¹			
↗ ⁰		↖ ⁰	0	1	↖ ⁰	↖ ¹	↖	0	1	↗	↗ ⁰	↗ ¹			
↗ ¹		↖ ¹	0	1	↖ ¹	↖ ¹	↖	1	0	↗	↗ ⁰	↗ ¹			

Table 4: The transition function of the automaton, part 3 of 9 (? by 1).

	?	−	w0	w1	∧	∨	r	w	w¬	↖	↖ ⁰	↖ ¹	↗	↗ ⁰	↗ ¹
0	0	↖	0	1	0↖ ⁰	0↖ ¹	0↖ ⁰	0↗ ⁰	0	0↗	0↗ ⁰	0↗ ¹	0	0	0
0 _−		0 _−	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0↗	0↗ ⁰	0↗ ¹			
0 _{w0}	0	↖	0	1	0↖ ⁰	0↖ ¹	0↖ ⁰	0↗ ⁰	0	0↗	0↗ ⁰	0↗ ¹	0	0	0
0 _{w1}	0	↖	0	1	0↖ ⁰	0↖ ¹	0↖ ⁰	0↗ ⁰	0	0↗	0↗ ⁰	0↗ ¹	0	0	0
0 _∧		0 _−	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0↗	0↗ ⁰	0↗ ¹			
0 _∨		0 _−	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0↗	0↗ ⁰	0↗ ¹			
0 _r		0 _−	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0↗	0↗ ⁰	0↗ ¹			
0 _w	0	↖	0	1	0↖ ⁰	0↖ ¹	0↖ ⁰	0↗ ⁰	0	0↗	0↗ ⁰	0↗ ¹	0	0	0
0 _{w¬}	0	↖	0	1	0↖ ⁰	0↖ ¹	0↖ ⁰	0↗ ⁰	0	0↗	0↗ ⁰	0↗ ¹	0	0	0
0↖		0 _−	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0↗	0↗ ⁰	0↗ ¹			
0↖ ⁰		0 _−	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0↗	0↗ ⁰	0↗ ¹			
0↖ ¹		0 _−	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0↗	0↗ ⁰	0↗ ¹			
0↗		↖	0	1	0↖ ⁰	0↖ ¹	0↖ ⁰	?	0	0↗	0↗ ⁰	0↗ ¹			
0↗ ⁰		0↖ ⁰	0	1	0↖ ⁰	0↖ ⁰	0↖ ⁰	0	1	0↗	0↗ ⁰	0↗ ¹			
0↗ ¹		0↖ ¹	0	1	0↖ ⁰	0↖ ¹	0↖ ⁰	1	0	0↗	0↗ ⁰	0↗ ¹			

Table 5: The transition function of the automaton, part 4 of 9 (0 by ?).

	0	0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↖	0 _{↖⁰}	0 _{↖¹}	0 _↗	0 _{↗⁰}	0 _{↗¹}
0	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 ₋		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{w0}	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 _{w1}	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 _∧		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _∨		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _r		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _w	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 _{w¬}	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 _↖		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{↖⁰}		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{↖¹}		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _↗		0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	?	0	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{↗⁰}		0 _{↖⁰}	0	1	0 _{↖⁰}	0 _{↖⁰}	0 _{↖⁰}	0	1	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{↗¹}		0 _{↖¹}	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	1	0	0 _↗	0 _{↗⁰}	0 _{↗¹}			

Table 6: The transition function of the automaton, part 5 of 9 (0 by 0).

	1	1 ₋	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↖	1 _{↖⁰}	1 _{↖¹}	1 _↗	1 _{↗⁰}	1 _{↗¹}
0	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 ₋		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{w0}	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 _{w1}	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 _∧		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _∨		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _r		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _w	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 _{w¬}	0	0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	0 _{↗⁰}	0	0 _↗	0 _{↗⁰}	0 _{↗¹}	0	0	0
0 _↖		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{↖⁰}		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{↖¹}		0 ₋	0 _{w0}	0 _{w1}	0 _∧	0 _∨	0 _r	0 _w	0 _{w¬}	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _↗		0 _↖	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	?	0	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{↗⁰}		0 _{↖⁰}	0	1	0 _{↖⁰}	0 _{↖⁰}	0 _{↖⁰}	0	1	0 _↗	0 _{↗⁰}	0 _{↗¹}			
0 _{↗¹}		0 _{↖¹}	0	1	0 _{↖⁰}	0 _{↖¹}	0 _{↖⁰}	1	0	0 _↗	0 _{↗⁰}	0 _{↗¹}			

Table 7: The transition function of the automaton, part 6 of 9 (0 by 1).

	?	—	$w0$	$w1$	$\wedge$	$\vee$	r	w	$w\neg$	$? \searrow$	$? \searrow^0$	$? \searrow^1$	$? \nearrow$	$? \nearrow^0$	$? \nearrow^1$
1	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
1_-		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
1_{w0}	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
1_{w1}	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
$1_\wedge$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1_\vee$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
1_r		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
1_w	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
$1_{w\neg}$	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
$1 \searrow$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \searrow^0$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \searrow^1$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \nearrow$		$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	?	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \nearrow^0$		$1 \searrow^0$	0	1	$1 \searrow^0$	$1 \searrow^1$	$1 \searrow^1$	0	1	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \nearrow^1$		$1 \searrow^1$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	1	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			

Table 8: The transition function of the automaton, part 7 of 9 (1 by ?).

	0	0_-	0_{w0}	0_{w1}	$0_\wedge$	$0_\vee$	0_r	0_w	$0_{w\neg}$	$0 \searrow$	$0 \searrow^0$	$0 \searrow^1$	$0 \nearrow$	$0 \nearrow^0$	$0 \nearrow^1$
1	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
1_-		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
1_{w0}	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
1_{w1}	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
$1_\wedge$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1_\vee$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
1_r		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
1_w	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
$1_{w\neg}$	1	$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	$0 \nearrow^0$	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$	1	1	1
$1 \searrow$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \searrow^0$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \searrow^1$		1_-	1_{w0}	1_{w1}	$1_\wedge$	$1_\vee$	1_r	1_w	$1_{w\neg}$	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \nearrow$		$1 \searrow$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	?	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \nearrow^0$		$1 \searrow^0$	0	1	$1 \searrow^0$	$1 \searrow^1$	$1 \searrow^1$	0	1	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			
$1 \nearrow^1$		$1 \searrow^1$	0	1	$1 \searrow^1$	$1 \searrow^1$	$1 \searrow^1$	1	0	$1 \nearrow$	$1 \nearrow^0$	$1 \nearrow^1$			

Table 9: The transition function of the automaton, part 8 of 9 (1 by 0).

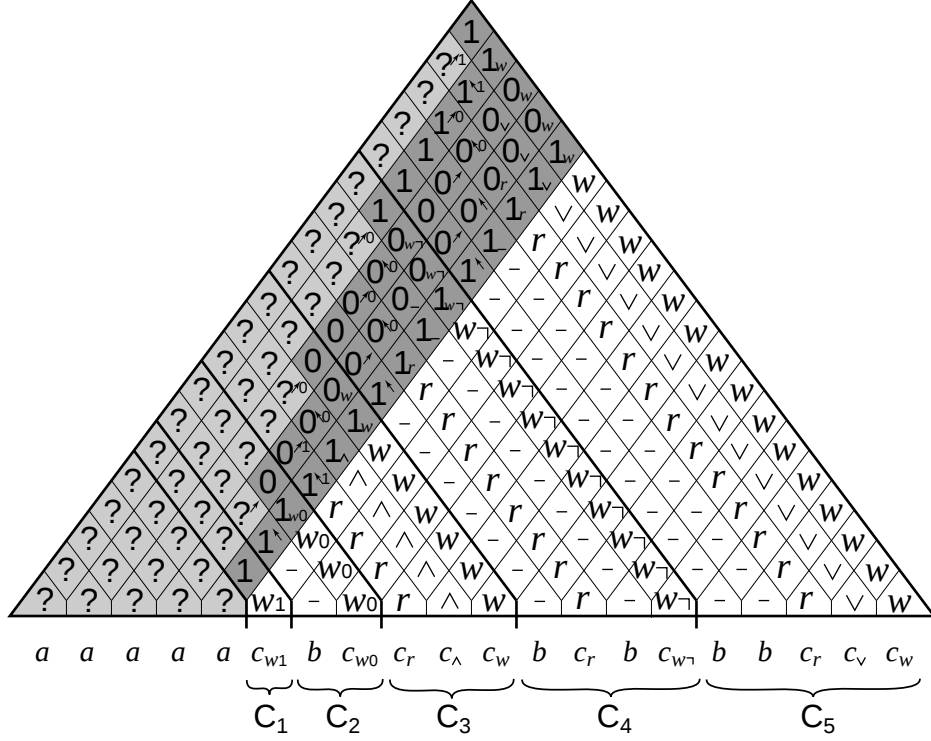


Figure 6: Sample computation of the automaton for CVP.

automaton that determines whether such a syntactically valid description denotes a circuit evaluating to one (in Section 4), we can easily notice that the intersection of these two languages is exactly the language of yes-instance of CVP under the encoding used. Now the results on computational equivalence of linear conjunctive grammars and triangular trellis automata [5], together with the closure of the languages they generate under intersection [3], directly imply that this **P**-complete language is generated by a certain linear conjunctive grammar, or, equivalently, accepted by a certain triangular trellis automaton. A complete linear conjunctive grammar for CVP is provided in Appendix A, and all strings of length less than 20 generated by this grammar are listed in Appendix B.

The result of this purely technical paper, besides partially satisfying our curiosity about the generative capabilities of linear conjunctive grammars, has numerous theoretical implications, which will be discussed in an upcoming article.

	1	1 ₋	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↖	1 _{↖⁰}	1 _{↖¹}	1 _↗	1 _{↗⁰}	1 _{↗¹}
1	1	1 _↖	0	1	1 _{↖¹}	1 _{↖¹}	1 _{↖¹}	0 _{↗⁰}	0	1 _↗	1 _{↗⁰}	1 _{↗¹}	1	1	1
1 ₋		1 ₋	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↗	1 _{↗⁰}	1 _{↗¹}			
1 _{w0}	1	1 _↖	0	1	1 _{↖¹}	1 _{↖¹}	1 _{↖¹}	0 _{↗⁰}	0	1 _↗	1 _{↗⁰}	1 _{↗¹}	1	1	1
1 _{w1}	1	1 _↖	0	1	1 _{↖¹}	1 _{↖¹}	1 _{↖¹}	0 _{↗⁰}	0	1 _↗	1 _{↗⁰}	1 _{↗¹}	1	1	1
1 _∧		1 ₋	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↗	1 _{↗⁰}	1 _{↗¹}			
1 _∨		1 ₋	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↗	1 _{↗⁰}	1 _{↗¹}			
1 _r		1 ₋	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↗	1 _{↗⁰}	1 _{↗¹}			
1 _w	1	1 _↖	0	1	1 _{↖¹}	1 _{↖¹}	1 _{↖¹}	0 _{↗⁰}	0	1 _↗	1 _{↗⁰}	1 _{↗¹}	1	1	1
1 _{w¬}	1	1 _↖	0	1	1 _{↖¹}	1 _{↖¹}	1 _{↖¹}	0 _{↗⁰}	0	1 _↗	1 _{↗⁰}	1 _{↗¹}	1	1	1
1 _↖		1 ₋	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↗	1 _{↗⁰}	1 _{↗¹}			
1 _{↖⁰}		1 ₋	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↗	1 _{↗⁰}	1 _{↗¹}			
1 _{↖¹}		1 ₋	1 _{w0}	1 _{w1}	1 _∧	1 _∨	1 _r	1 _w	1 _{w¬}	1 _↗	1 _{↗⁰}	1 _{↗¹}			
1 _↗		1 _↖	0	1	1 _{↖¹}	1 _{↖¹}	1 _{↖¹}	?	0	1 _↗	1 _{↗⁰}	1 _{↗¹}			
1 _{↗⁰}		1 _{↖⁰}	0	1	1 _{↖⁰}	1 _{↖¹}	1 _{↖¹}	0	1	1 _↗	1 _{↗⁰}	1 _{↗¹}			
1 _{↗¹}		1 _{↖¹}	0	1	1 _{↖¹}	1 _{↖¹}	1 _{↖¹}	1	0	1 _↗	1 _{↗⁰}	1 _{↗¹}			

Table 10: The transition function of the automaton, part 9 of 9 (1 by 1).

A A complete grammar for CVP

The following linear conjunctive grammar is comprised of the grammar defined in Section 3, the grammar produced out of the automaton constructed in Section 4 using the methods of [5] (the only deviation from the methods of [5] is in that some entries of the transition table of the original automaton were left undefined, and no rules corresponding to these entries were created; this helped to make the grammar a somewhat shorter, without altering the language it generates), new start symbol S with a single rule $S \rightarrow S' \& S''$, which effectively forces both the syntactical and semantic conditions to hold at the same time.

The set of terminal symbols is $\Sigma = \{a, b, c_r, c_\wedge, c_\vee, c_w, c_{w\neg}, c_{w0}, c_{w1}\}$. The set of nonterminals is the union of $\{S\}$, $\{S', Count, Skip, Check, Check_1, Check_2, Check_3\}$ and $\{S'', ?, 0, 1, -, 0_-, 1_-, w0, 0_{w0}, 1_{w0}, w1, 0_{w1}, 1_{w1}, \wedge, 0_\wedge, 1_\wedge, \vee, 0_\vee, 1_\vee, r, 0_r, 1_r, w, 0_w, 1_w, w\neg, 0_{w\neg}, 1_{w\neg}, ?\swarrow, 0\swarrow, 1\swarrow, ?\nwarrow^0, 0\nwarrow^0, 1\nwarrow^0, ?\nwarrow^1, 0\nwarrow^1, 1\nwarrow^1, ?\nearrow, 0\nearrow, 1\nearrow, ?\nearrow^0, 0\nearrow^0, 1\nearrow^0, ?\nearrow^1, 0\nearrow^1, 1\nearrow^1\}$; the total number of nonterminals is $1 + 7 + (45 + 1) = 54$.

The set of rules contains the $1 + (6 + 8 + 9 + 11) + (1 + 1665 \cdot 9^2) = 134901$ rules listed below; it should be noted that the overwhelming majority ($1665 \cdot 9^2$) of

[illegible]

[illegible]

[illegible]

23

26

[illegible]

28

$$\begin{array}{lll}
0 \rightarrow s \ 1_{w\neg} \& 1 \nearrow t \quad (\forall s, t \in \Sigma) & 1 \nearrow^0 \rightarrow s \ ?^{\neg^0} \& 1 \nearrow^1 t \quad (\forall s, t \in \Sigma) & 1 \nearrow^1 \rightarrow s \ 0^{\neg^1} \& 1 \nearrow^1 t \quad (\forall s, t \in \Sigma) \\
1 \nearrow \rightarrow s \ ?^{\neg} \& 1 \nearrow^1 t \quad (\forall s, t \in \Sigma) & 1 \nearrow^0 \rightarrow s \ 0^{\neg^0} \& 1 \nearrow^1 t \quad (\forall s, t \in \Sigma) & 1 \nearrow^1 \rightarrow s \ 1^{\neg^1} \& 1 \nearrow^1 t \quad (\forall s, t \in \Sigma) \\
1 \nearrow \rightarrow s \ 0^{\neg} \& 1 \nearrow^1 t \quad (\forall s, t \in \Sigma) & 1 \nearrow^0 \rightarrow s \ 1^{\neg^0} \& 1 \nearrow^1 t \quad (\forall s, t \in \Sigma) & \\
1 \nearrow \rightarrow s \ 1^{\neg} \& 1 \nearrow^1 t \quad (\forall s, t \in \Sigma) & 1 \nearrow^1 \rightarrow s \ ?^{\neg^1} \& 1 \nearrow^1 t \quad (\forall s, t \in \Sigma) &
\end{array}$$

B All circuits of ≤ 4 gates evaluating to one

Following is the list of all strings of length less than 20 that encode circuits evaluating to one, or, equivalently, the list of literal representations of all circuits containing 4 gates or less. These are:

- 1 circuit of 1 gate, represented with a string of length 2,
- 3 circuits of 2 gates, each represented with a string of length 5,
- 18 circuits of 3 gates, each represented with a string of length 9, and
- 198 circuits of 4 gates, each represented with a string of length 14.

$a \ c_{w1}$ (see Figure 5(e) on page 13 and Figure 6 on page 18)

$a \ a \ c_{w0} \ b \ c_{w1}$
 $a \ a \ c_{w0} \ c_r \ c_{w\neg}$
 $a \ a \ c_{w1} \ b \ c_{w1}$
 $a \ a \ a \ c_{w0} \ b \ c_{w0} \ b \ b \ c_{w1}$
 $a \ a \ a \ c_{w0} \ b \ c_{w0} \ b \ c_r \ c_{w\neg}$
 $a \ a \ a \ c_{w0} \ b \ c_{w0} \ c_r \ b \ c_{w\neg}$
 $a \ a \ a \ c_{w0} \ b \ c_{w1} \ b \ b \ c_{w1}$
 $a \ a \ a \ c_{w0} \ b \ c_{w1} \ c_r \ b \ c_{w\neg}$
 $a \ a \ a \ c_{w0} \ b \ c_{w1} \ c_r \ c_{\vee} \ c_w$
 $a \ a \ a \ c_{w0} \ c_r \ c_{w\neg} \ b \ b \ c_{w1}$
 $a \ a \ a \ c_{w0} \ c_r \ c_{w\neg} \ c_r \ b \ c_{w\neg}$
 $a \ a \ a \ c_{w0} \ c_r \ c_{w\neg} \ c_r \ c_{\vee} \ c_w$
 $a \ a \ a \ c_{w1} \ b \ c_{w0} \ b \ b \ c_{w1}$
 $a \ a \ a \ c_{w1} \ b \ c_{w0} \ b \ c_r \ c_{w\neg}$
 $a \ a \ a \ c_{w1} \ b \ c_{w0} \ c_r \ c_{\vee} \ c_w$
 $a \ a \ a \ c_{w1} \ b \ c_{w1} \ b \ b \ c_{w1}$
 $a \ a \ a \ c_{w1} \ b \ c_{w1} \ c_r \ c_{\wedge} \ c_w$
 $a \ a \ a \ c_{w1} \ b \ c_{w1} \ c_r \ c_{\vee} \ c_w$
 $a \ a \ a \ c_{w1} \ c_r \ c_{w\neg} \ b \ b \ c_{w1}$
 $a \ a \ a \ c_{w1} \ c_r \ c_{w\neg} \ b \ c_r \ c_{w\neg}$
 $a \ a \ a \ c_{w1} \ c_r \ c_{w\neg} \ c_r \ c_{\vee} \ c_w$
 $a \ a \ a \ a \ c_{w0} \ b \ c_{w0} \ b \ b \ c_{w0} \ b \ b \ b \ c_{w1}$
 $a \ a \ a \ a \ c_{w0} \ b \ c_{w0} \ b \ b \ c_{w0} \ b \ b \ c_r \ c_{w\neg}$
 $a \ a \ a \ a \ c_{w0} \ b \ c_{w0} \ b \ b \ c_{w0} \ b \ c_r \ b \ c_{w\neg}$

$a a a a c_{w0} b c_{w0} b b c_{w0} c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} b b c_{w1} b b b c_{w1}$
 $a a a a c_{w0} b c_{w0} b b c_{w1} b c_r b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} b b c_{w1} b c_r c_{\vee} c_w$
 $a a a a c_{w0} b c_{w0} b b c_{w1} c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} b b c_{w1} c_r b c_{\vee} c_w$
 $a a a a c_{w0} b c_{w0} b c_r c_{w\neg} b b b c_{w1}$
 $a a a a c_{w0} b c_{w0} b c_r c_{w\neg} b c_r b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} b c_r c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w0} b c_{w0} b c_r c_{w\neg} c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} b c_r c_{w\neg} c_r b c_{\vee} c_w$
 $a a a a c_{w0} b c_{w0} c_r b c_{w\neg} b b b c_{w1}$
 $a a a a c_{w0} b c_{w0} c_r b c_{w\neg} b c_r b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} c_r b c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w0} b c_{w0} c_r b c_{w\neg} c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} c_r b c_{w\neg} c_r b c_{\vee} c_w$
 $a a a a c_{w0} b c_{w0} c_r c_{\wedge} c_w b b b c_{w1}$
 $a a a a c_{w0} b c_{w0} c_r c_{\wedge} c_w b b c_r c_{w\neg}$
 $a a a a c_{w0} b c_{w0} c_r c_{\wedge} c_w b c_r b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} c_r c_{\wedge} c_w c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} c_r c_{\vee} c_w b b b c_{w1}$
 $a a a a c_{w0} b c_{w0} c_r c_{\vee} c_w b b c_r c_{w\neg}$
 $a a a a c_{w0} b c_{w0} c_r c_{\vee} c_w b c_r b c_{w\neg}$
 $a a a a c_{w0} b c_{w0} c_r c_{\vee} c_w c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w1} b b c_{w0} b b b c_{w1}$
 $a a a a c_{w0} b c_{w1} b b c_{w0} b b c_r c_{w\neg}$
 $a a a a c_{w0} b c_{w1} b b c_{w0} b c_r c_{\vee} c_w$
 $a a a a c_{w0} b c_{w1} b b c_{w0} c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w1} b b c_{w0} c_r c_{\vee} b c_w$
 $a a a a c_{w0} b c_{w1} b b c_{w1} b b b c_{w1}$
 $a a a a c_{w0} b c_{w1} b b c_{w1} b c_r c_{\wedge} c_w$
 $a a a a c_{w0} b c_{w1} b b c_{w1} b c_r c_{\vee} c_w$
 $a a a a c_{w0} b c_{w1} b b c_{w1} c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w1} b b c_{w1} c_r b c_{\vee} c_w$
 $a a a a c_{w0} b c_{w1} b b c_{w1} c_r c_{\vee} b c_w$
 $a a a a c_{w0} b c_{w1} b c_r c_{w\neg} b b b c_{w1}$
 $a a a a c_{w0} b c_{w1} b c_r c_{w\neg} b b c_r c_{w\neg}$
 $a a a a c_{w0} b c_{w1} b c_r c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w0} b c_{w1} b c_r c_{w\neg} c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w1} b c_r c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w0} b c_{w1} c_r b c_{w\neg} b b b c_{w1}$
 $a a a a c_{w0} b c_{w1} c_r b c_{w\neg} b c_r c_{\wedge} c_w$
 $a a a a c_{w0} b c_{w1} c_r b c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w0} b c_{w1} c_r b c_{w\neg} c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w1} c_r b c_{w\neg} c_r b c_{\vee} c_w$

$a a a a c_{w0} b c_{w1} c_r b c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w0} b c_{w1} c_r c_{\wedge} c_w b b b c_{w1}$
 $a a a a c_{w0} b c_{w1} c_r c_{\wedge} c_w b b c_r c_{w\neg}$
 $a a a a c_{w0} b c_{w1} c_r c_{\wedge} c_w b c_r c_{\vee} c_w$
 $a a a a c_{w0} b c_{w1} c_r c_{\wedge} c_w c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w1} c_r c_{\wedge} c_w c_r c_{\vee} b c_w$
 $a a a a c_{w0} b c_{w1} c_r c_{\vee} c_w b b b c_{w1}$
 $a a a a c_{w0} b c_{w1} c_r c_{\vee} c_w b c_r c_{\wedge} c_w$
 $a a a a c_{w0} b c_{w1} c_r c_{\vee} c_w b c_r c_{\vee} c_w$
 $a a a a c_{w0} b c_{w1} c_r c_{\vee} c_w c_r b b c_{w\neg}$
 $a a a a c_{w0} b c_{w1} c_r c_{\vee} c_w c_r b c_{\vee} c_w$
 $a a a a c_{w0} b c_{w1} c_r c_{\vee} c_w c_r c_{\vee} b c_w$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w0} b b b c_{w1}$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w0} b b c_r c_{w\neg}$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w0} b c_r c_{\vee} c_w$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w0} c_r b b c_{w\neg}$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w0} c_r c_{\vee} b c_w$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w1} b b b c_{w1}$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w1} b c_r c_{\wedge} c_w$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w1} b c_r c_{\vee} c_w$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w1} c_r b b c_{w\neg}$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w1} c_r b c_{\vee} c_w$
 $a a a a c_{w0} c_r c_{w\neg} b b c_{w1} c_r c_{\vee} b c_w$
 $a a a a c_{w0} c_r c_{w\neg} b c_r c_{w\neg} b b b c_{w1}$
 $a a a a c_{w0} c_r c_{w\neg} b c_r c_{w\neg} b b c_r c_{w\neg}$
 $a a a a c_{w0} c_r c_{w\neg} b c_r c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w0} c_r c_{w\neg} b c_r c_{w\neg} c_r b b c_{w\neg}$
 $a a a a c_{w0} c_r c_{w\neg} b c_r c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w0} c_r c_{w\neg} c_r b c_{w\neg} b b b c_{w1}$
 $a a a a c_{w0} c_r c_{w\neg} c_r b c_{w\neg} b c_r c_{\wedge} c_w$
 $a a a a c_{w0} c_r c_{w\neg} c_r b c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w0} c_r c_{w\neg} c_r b c_{w\neg} c_r b b c_{w\neg}$
 $a a a a c_{w0} c_r c_{w\neg} c_r b c_{w\neg} c_r b c_{\vee} c_w$
 $a a a a c_{w0} c_r c_{w\neg} c_r b c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\wedge} c_w b b b c_{w1}$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\wedge} c_w b b c_r c_{w\neg}$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\wedge} c_w b c_r c_{\vee} c_w$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\wedge} c_w c_r b b c_{w\neg}$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\vee} c_w b b b c_{w1}$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\vee} c_w b c_r c_{\wedge} c_w$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\vee} c_w b c_r c_{\vee} c_w$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\vee} c_w c_r b b c_{w\neg}$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\vee} c_w c_r b c_{\vee} c_w$
 $a a a a c_{w0} c_r c_{w\neg} c_r c_{\vee} c_w c_r c_{\vee} b c_w$

$a a a a c_{w1} b c_{w0} b b c_{w0} b b b c_{w1}$
 $a a a a c_{w1} b c_{w0} b b c_{w0} b b c_r c_{w\neg}$
 $a a a a c_{w1} b c_{w0} b b c_{w0} b c_r b c_{w\neg}$
 $a a a a c_{w1} b c_{w0} b b c_{w0} c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w0} b b c_{w0} c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w0} b b c_{w1} b b b c_{w1}$
 $a a a a c_{w1} b c_{w0} b b c_{w1} b c_r b c_{w\neg}$
 $a a a a c_{w1} b c_{w0} b b c_{w1} b c_r c_{\vee} c_w$
 $a a a a c_{w1} b c_{w0} b b c_{w1} c_r b c_{\wedge} c_w$
 $a a a a c_{w1} b c_{w0} b b c_{w1} c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w0} b b c_{w1} c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w0} b c_r c_{w\neg} b b b c_{w1}$
 $a a a a c_{w1} b c_{w0} b c_r c_{w\neg} b c_r b c_{w\neg}$
 $a a a a c_{w1} b c_{w0} b c_r c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w1} b c_{w0} b c_r c_{w\neg} c_r b c_{\wedge} c_w$
 $a a a a c_{w1} b c_{w0} b c_r c_{w\neg} c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w0} b c_r c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w0} c_r b c_{w\neg} b b b c_{w1}$
 $a a a a c_{w1} b c_{w0} c_r b c_{w\neg} b b c_r c_{w\neg}$
 $a a a a c_{w1} b c_{w0} c_r b c_{w\neg} b c_r b c_{w\neg}$
 $a a a a c_{w1} b c_{w0} c_r b c_{w\neg} c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w0} c_r b c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w0} c_r c_{\wedge} c_w b b b c_{w1}$
 $a a a a c_{w1} b c_{w0} c_r c_{\wedge} c_w b b c_r c_{w\neg}$
 $a a a a c_{w1} b c_{w0} c_r c_{\wedge} c_w b c_r b c_{w\neg}$
 $a a a a c_{w1} b c_{w0} c_r c_{\wedge} c_w c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w0} c_r c_{\wedge} c_w c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w0} c_r c_{\vee} c_w b b b c_{w1}$
 $a a a a c_{w1} b c_{w0} c_r c_{\vee} c_w b c_r b c_{w\neg}$
 $a a a a c_{w1} b c_{w0} c_r c_{\vee} c_w b c_r c_{\vee} c_w$
 $a a a a c_{w1} b c_{w0} c_r c_{\vee} c_w c_r b c_{\wedge} c_w$
 $a a a a c_{w1} b c_{w0} c_r c_{\vee} c_w c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w0} c_r c_{\vee} c_w c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w1} b b c_{w0} b b b c_{w1}$
 $a a a a c_{w1} b c_{w1} b b c_{w0} b b c_r c_{w\neg}$
 $a a a a c_{w1} b c_{w1} b b c_{w0} b c_r c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} b b c_{w0} c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} b b c_{w0} c_r c_{\wedge} b c_w$
 $a a a a c_{w1} b c_{w1} b b c_{w0} c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w1} b b c_{w1} b b b c_{w1}$
 $a a a a c_{w1} b c_{w1} b b c_{w1} b c_r c_{\wedge} c_w$
 $a a a a c_{w1} b c_{w1} b b c_{w1} b c_r c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} b b c_{w1} c_r b c_{\wedge} c_w$
 $a a a a c_{w1} b c_{w1} b b c_{w1} c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} b b c_{w1} c_r c_{\wedge} b c_w$

(see Figures 5(b) and 6)

$a a a a c_{w1} b c_{w1} b b c_{w1} c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w1} b c_r c_{w\neg} b b b c_{w1}$
 $a a a a c_{w1} b c_{w1} b c_r c_{w\neg} b b c_r c_{w\neg}$
 $a a a a c_{w1} b c_{w1} b c_r c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} b c_r c_{w\neg} c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} b c_r c_{w\neg} c_r c_{\wedge} b c_w$
 $a a a a c_{w1} b c_{w1} b c_r c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w1} c_r b c_{w\neg} b b b c_{w1}$
 $a a a a c_{w1} b c_{w1} c_r b c_{w\neg} b b c_r c_{w\neg}$
 $a a a a c_{w1} b c_{w1} c_r b c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} c_r b c_{w\neg} c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} c_r b c_{w\neg} c_r c_{\wedge} b c_w$
 $a a a a c_{w1} b c_{w1} c_r b c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\wedge} c_w b b b c_{w1}$
 $a a a a c_{w1} b c_{w1} c_r c_{\wedge} c_w b c_r c_{\wedge} c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\wedge} c_w b c_r c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\wedge} c_w c_r b c_{\wedge} c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\wedge} c_w c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\wedge} c_w c_r c_{\wedge} b c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\wedge} c_w c_r c_{\vee} b c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\vee} c_w b b b c_{w1}$
 $a a a a c_{w1} b c_{w1} c_r c_{\vee} c_w b c_r c_{\wedge} c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\vee} c_w b c_r c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\vee} c_w c_r b c_{\wedge} c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\vee} c_w c_r b c_{\vee} c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\vee} c_w c_r c_{\wedge} b c_w$
 $a a a a c_{w1} b c_{w1} c_r c_{\vee} c_w c_r c_{\vee} b c_w$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w0} b b b c_{w1}$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w0} b b c_r c_{w\neg}$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w0} b c_r b c_{w\neg}$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w0} c_r b c_{\vee} c_w$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w0} c_r c_{\vee} b c_w$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w1} b b b c_{w1}$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w1} b c_r b c_{w\neg}$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w1} b c_r c_{\vee} c_w$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w1} c_r b c_{\wedge} c_w$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w1} c_r b c_{\vee} c_w$
 $a a a a c_{w1} c_r c_{w\neg} b b c_{w1} c_r c_{\vee} b c_w$
 $a a a a c_{w1} c_r c_{w\neg} b c_r c_{w\neg} b b b c_{w1}$
 $a a a a c_{w1} c_r c_{w\neg} b c_r c_{w\neg} b c_r b c_{w\neg}$
 $a a a a c_{w1} c_r c_{w\neg} b c_r c_{w\neg} b c_r c_{\vee} c_w$
 $a a a a c_{w1} c_r c_{w\neg} b c_r c_{w\neg} c_r b c_{\wedge} c_w$
 $a a a a c_{w1} c_r c_{w\neg} b c_r c_{w\neg} c_r b c_{\vee} c_w$
 $a a a a c_{w1} c_r c_{w\neg} b c_r c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w1} c_r c_{w\neg} c_r b c_{w\neg} b b b c_{w1}$

$a a a a c_{w1} c_r c_{w\neg} c_r b c_{w\neg} b b c_r c_{w\neg}$
 $a a a a c_{w1} c_r c_{w\neg} c_r b c_{w\neg} b c_r b c_{w\neg}$
 $a a a a c_{w1} c_r c_{w\neg} c_r b c_{w\neg} c_r b c_{\vee} c_w$
 $a a a a c_{w1} c_r c_{w\neg} c_r b c_{w\neg} c_r c_{\vee} b c_w$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\wedge} c_w b b b c_{w1}$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\wedge} c_w b b c_r c_{w\neg}$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\wedge} c_w b c_r b c_{w\neg}$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\wedge} c_w c_r b c_{\vee} c_w$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\wedge} c_w c_r c_{\vee} b c_w$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\vee} c_w b b b c_{w1}$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\vee} c_w b c_r b c_{w\neg}$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\vee} c_w b c_r c_{\vee} c_w$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\vee} c_w c_r b c_{\wedge} c_w$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\vee} c_w c_r b c_{\vee} c_w$
 $a a a a c_{w1} c_r c_{w\neg} c_r c_{\vee} c_w c_r c_{\vee} b c_w$

References

- [1] A. Okhotin, “Conjunctive grammars”, *Journal of Automata, Languages and Combinatorics*, 6:4 (2001), 519–535.
- [2] A. Okhotin, “A recognition and parsing algorithm for arbitrary conjunctive grammars”, *Theoretical Computer Science*, to appear.
- [3] A. Okhotin, “On the closure properties of linear conjunctive languages”, *Theoretical Computer Science*, to appear.
- [4] A. Okhotin, “Whale Calf, a parser generator for conjunctive grammars”, Pre-proceedings of *CIAA 2002*, Tours, France, July 3–5, 2002, 211–216. The software is available at <http://www.cs.queensu.ca/home/okhotin/whalecalf/>.
- [5] A. Okhotin, “Automaton representation of linear conjunctive languages”, to be presented at *DLT 2002*, Kyoto, Japan, September 18–21, 2002.