# Conjunctive grammars, cellular automata and logic

Théo Grente, Etienne Grandjean

# Conjunctive Grammars, Cellular Automata and Logic

**Théo Grente** ✉
GREYC, Université de Caen Normandie, France

**Étienne Grandjean** ✉
GREYC, Université de Caen Normandie, France

## Abstract

The expressive power of the class `Conj` of *conjunctive languages*, i.e. languages generated by the *conjunctive grammars* of Okhotin, is largely unknown, while its restriction `LinConj` to *linear conjunctive grammars* equals the class of languages recognized by *real-time one-dimensional one-way* cellular automata. We prove two weakened versions of the open question `Conj` $\subseteq$? `RealTime1CA`, where `RealTime1CA` is the class of languages recognized by *real-time one-dimensional two-way* cellular automata:

1. it is true for *unary* languages;
2. `Conj` $\subseteq$ `RealTime2OCA`, i.e. any conjunctive language is recognized by a *real-time two-dimensional one-way* cellular automaton.

Interestingly, we express the rules of a conjunctive grammar in two *Horn logics*, which *exactly characterize* the complexity classes `RealTime1CA` and `RealTime2OCA`.

**2012 ACM Subject Classification** Theory of computation → Complexity theory and logic; Theory of computation → Formal languages and automata theory

**Keywords and phrases** Computational complexity, Real-time, One-dimensional/two-dimensional cellular automaton, One-way/two-way communication, Grid-circuit, Unary language, Descriptive complexity, Existential second-order logic, Horn formula

**Digital Object Identifier** 10.4230/OASIcs.AUTOMATA.2021.8

**Funding** This work has been partly supported by the PING/ACK project of the French National Agency for Research (ANR-18-CE40-0011).

**Acknowledgements** This paper would not exist without the inspiration of Véronique Terrier. Her in-depth knowledge of cellular automata, their complexity classes and their closure properties in relation to formal language theory, the references and advice she generously gave us, as well as her careful reading, were essential in designing and finalizing the results and the presentation of the paper. E.g., the class diagram of Figure 8 is due to her. We also thank the three reviewers of this paper for their careful reading and their detailed constructive comments and suggestions which helped us improve some parts of this paper.

## 1 Introduction

For decades, logic has maintained close relationships with, on the one hand, computational models [31] and computational complexity [3], in particular through descriptive complexity [7, 16, 21, 11, 14, 2], and on the other hand with formal language theory and grammars [8, 21].

**Conjunctive grammars versus logic.** Okhotin [26] wrote that "context-free grammars may be thought of as a *logic* for inductive description of syntax in which the propositional connectives available... are restricted to *disjunction only*". Thus, twenty years ago, the same author introduced *conjunctive grammars* [22] as an extension of context-free grammars by adding an explicit *conjunction* operation within the grammar rules.


27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021).
Editors: Alonso Castillo-Ramirez, Pierre Guillon, and Kévin Perrot; Article No. 8; pp. 8:1–8:19
OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

As shown by Okhotin [22], conjunctive grammars – and more generally, Boolean grammars [24, 26] – inherit the parsing algorithms of the ordinary context-free grammars, without increasing their computational complexity. However, the expressive power of these grammars is largely unknown. The fact that the class $\texttt{Conj}$ of languages generated by conjunctive grammars has many closure properties – it is trivially closed under reverse, concatenation, Kleene closure, disjunction and conjunction – suggests that this class has equivalent definitions in computational complexity and/or logic.

**Conjunctive grammars versus real-time cellular automata.** Note that the $\texttt{LinConj}$ subclass of languages generated by *linear* conjunctive grammars was found to be equal to the $\texttt{Trellis}$ class of languages recognized by *trellis* automata [25], or equivalently, *one-way real-time* cellular automata. Faced with this result, it is tempting to ask the following question: is the larger class $\texttt{Conj}$ equal to the class $\texttt{RealTime1CA}$ of languages recognized by *two-way* real-time cellular automata? Either answer to this question has strong consequences:

- If $\texttt{Conj} = \texttt{RealTime1CA}$ then each of the two classes will benefit from the closure properties of the other class; in particular, $\texttt{RealTime1CA}$ would be closed under reverse, which was shown by [15] to imply $\texttt{RealTime1CA} = \texttt{LinearTime}_{\texttt{1CA}}$, i.e. real-time is nothing but *linear time* for cellular automata, a surprising positive answer to a longstanding open question [6, 28, 30].
- If $\texttt{Conj} \neq \texttt{RealTime1CA}$ then $\texttt{Conj} \subsetneq \texttt{DSPACE}(n)$ or $\texttt{RealTime1CA} \subsetneq \texttt{DSPACE}(n)$: any of these strict inclusions would be a striking result.

Real-time is the minimal time of cellular automata (CA). Recall that $\texttt{RealTime1CA}$ (resp. $\texttt{Trellis}$) is the class of languages recognized in real-time by *one-dimensional* CA with *two-way* (resp. *one-way*) communication and input word given in parallel. We know the strict inclusion $\texttt{Trellis} \subsetneq \texttt{RealTime1CA}$. The robustness of these classes is attested by their characterization by two sub-logics of $\texttt{ESO}$ – the *existential second-order* logic, which characterizes $\texttt{NP}$ – with *Horn formulas* as their first-order parts[1], and called respectively $\texttt{pred-ESO-HORN}$ and $\texttt{incl-ESO-HORN}$, see [12, 13]. For short, we write $\texttt{RealTime1CA} = \texttt{pred-ESO-HORN}$ and $\texttt{Trellis} = \texttt{incl-ESO-HORN}$.

**Results of this paper.** This paper focuses on the relationships between the class of conjunctive languages and the real-time classes of cellular automata. Although we do not know the answer to the question $\texttt{Conj} =? \texttt{RealTime1CA}$ or even to the question of the inclusion $\texttt{Conj} \subseteq? \texttt{RealTime1CA}$, we prove two weakened versions of this inclusion:

1. $\texttt{Conj}_1 \subseteq \texttt{RealTime1CA}_1$: The inclusion holds when restricted to *unary* languages[2].
2. $\texttt{Conj} \subseteq \texttt{RealTime2OCA}$: The inclusion holds for real-time of *two-dimensional one-way* cellular automata (2-OCA). (We have $\texttt{RealTime1CA} \subseteq \texttt{RealTime2OCA}$.)

To grasp the scope of inclusion (1), it is important to note that unlike the subclass $\texttt{CFL}_1$ of the unary languages of the class of context-free languages, which is reduced to regular languages, $\texttt{CFL}_1 = \texttt{Reg}_1$, the class $\texttt{Conj}_1$ was shown by Jez [17] to be much larger than $\texttt{Reg}_1$. Understanding its precise expressiveness seems as difficult a problem to us as for $\texttt{Conj}$.

Our inclusion (2) improves the inclusion $\texttt{CFL} \subseteq \texttt{RealTime2SOCA}$, where $\texttt{RealTime2SOCA}$ denotes the class of languages recognized by real-time *sequential* two-dimensional one-way cellular automata, proved by Terrier [29], who uses a result by King [18] and improves results by Kosaraju [20] and Chang et al. [4]. Terrier's result derives transitively from (2): $\texttt{CFL} \subseteq \texttt{Conj} \subseteq \texttt{RealTime2OCA} \subseteq \texttt{RealTime2SOCA}$.

---

[1] The class $\texttt{ESO-HORN}$ of languages defined by existential second-order formulas with Horn formulas as their first-order parts is exactly $\texttt{PTIME}$, see [10, 11].

[2] The subclass of the unary languages of a class of languages $\mathcal{C}$ is denoted $\mathcal{C}_1$.

Inclusion (2) seems difficult to improve. Since any problem in `RealTime1CA` is decidable in time $O(n^2)$ (by a RAM algorithm), the hypothetical inclusion $\texttt{Conj} \subseteq \texttt{RealTime1CA}$ implies that any conjunctive language is decidable in time $O(n^2)$: this would be a breakthrough!

**Logic as a bridge from problems and grammars to real-time CAs.** Logic has been the basis of logic programming and database queries for decades, especially Horn logic through the Prolog and Datalog programming languages [1, 19, 11]. Likewise, the above-mentioned logical characterizations of real-time complexity classes of CAs, $\texttt{RealTime1CA} = \texttt{pred-ESO-HORN}$ and $\texttt{Trellis} = \texttt{incl-ESO-HORN}$, have been used to easily show that several problems belong to the `RealTime1CA` or `Trellis` class by inductively expressing/programming the problems in the corresponding Horn logic, see [12, 13].

In this paper, the same logic programming method is adopted. We prove inclusion (1) $\texttt{Conj}_1 \subseteq \texttt{RealTime1CA}_1$ by expressing a unary language generated by a conjunctive grammar in the `pred-ESO-HORN` logic. Inclusion (1) follows, by the equality $\texttt{pred-ESO-HORN} = \texttt{RealTime1CA}$. Similarly, to prove inclusion (2) $\texttt{Conj} \subseteq \texttt{RealTime2OCA}$, we first design a logic denoted `incl-pred-ESO-HORN` so that $\texttt{incl-pred-ESO-HORN} = \texttt{RealTime2OCA}$. Then, we express any conjunctive language in this logic, proving that it belongs to `RealTime2OCA`, as claimed. Thus, the heart of each proof consists in presenting a formula of a certain *Horn logic*, which *inductively* expresses how a word is generated by a conjunctive grammar: the Horn clauses of the formula *naturally* imitate the rules of the grammar.

**Our proof method and the paper structure.** After Section 2 gives some definitions, Sections 3 and 4 present inclusions (1) and (2) and their proofs with a common plan: Subsection 3.1 (resp. 4.1) expresses the inductive generating process of a conjunctive grammar, assumed in binary (Chomsky) normal form in the logic `pred-ESO-HORN` (resp. `incl-pred-ESO-HORN`). Subsection 3.2 (resp. 4.2) shows that any formula of this logic can be normalized into a formula which mimics the computation of a two-dimensional (resp. three-dimensional) *grid-circuit* called `Grid` (resp. `Cube`); Subsection 3.3 (resp. 4.3) translates the grid-circuit into a real-time one-dimensional CA (resp. two-dimensional OCA). Note that we prove the equivalence of our logics with grid-circuits and CA real-time[3]: $\texttt{pred-ESO-HORN} = \texttt{Grid} = \texttt{RealTime1CA}$ and $\texttt{incl-pred-ESO-HORN} = \texttt{Cube} = \texttt{RealTime2OCA}$. Section 5 deals briefly with the meaning of our results and open problems around a diagram of the known relations between the `Conj` class and the CA complexity classes studied here, for the general case and for the unary case.

# 2 Preliminaries

## 2.1 Conjunctive grammars and their binary normal form

Conjunctive grammars extend context-free grammars with a conjunction operation.

▶ **Definition 1** (Conjunctive grammar, conjunctive language). *[22, 23]*

- *A* conjunctive grammar *is a tuple $G = (\Sigma, N, P, S)$ where $\Sigma$ is the finite set of* terminal symbols*, $N$ is the finite set of* nonterminal *symbols, $S \in N$ is the* initial *symbol, and $P$ is the finite set of* rules*, each of the form $A \to \alpha_1 \& ... \& \alpha_m$, for $m \geq 1$ and $\alpha_i \in (\Sigma \cup N)^+$.*

[3] We have chosen to give here a simplified proof of the logical characterization $\texttt{pred-ESO-HORN} = \texttt{Grid} = \texttt{RealTime1CA}$ already proved in [12] so that this paper is self-content, but above all because our proof of the similar result $\texttt{incl-pred-ESO-HORN} = \texttt{Cube} = \texttt{RealTime2OCA}$ is an extension of it.

- *The set of words $L(A) \subseteq \Sigma^+$* generated *by any $A \in N$ is defined by induction: if the rules for $A$ are $A \to \alpha_1^1\&...\&\alpha_{m_1}^1 \mid \cdots \mid \alpha_1^k\&...\&\alpha_{m_k}^k$, then $L(A) \coloneqq \bigcup_{i=1}^{k} \bigcap_{j=1}^{m_i} L(\alpha_j^i)$. (As usual, take the least solution of the language equations defining the sets $L(A)$, for $A \in N$.)*
- *The* language generated *by the grammar $G$ is $L(S)$. It is called a* conjunctive language*.*

Okhotin [26] gave many examples of conjunctive languages which are not context-free. Moreover, Jez [17] proved that there are such languages on unary alphabet, in particular, the set $\{a^{4^k} \mid k \in \mathbb{N}\}$ is a conjunctive language which is not context-free (= not regular).

We will mainly use the binary normal form of conjunctive grammars, which extends the Chomsky normal form of context-free grammars. Each conjunctive grammar can be rewritten in an equivalent binary normal form [22, 26].

▶ **Definition 2** (Binary normal form [22]). *A conjunctive grammar $G = (\Sigma, N, P, S)$ is in* binary normal form *if each rule in $P$ has one of the two following forms:*
- *a long rule: $A \to B_1C_1\&...\&B_mC_m$ ($m \geq 1, B_i, C_j \in N$);*
- *a short rule: $A \to a$ ($a \in \Sigma$).*

## 2.2 Elements of logic

The underlying structure we will adopt to encode an input word $w = w_1 \dots w_n$ over its index interval $[1, n] = \{1, \dots, n\}$ uses the *successor* and *predecessor* functions and the monadic predicates `min` and `max` as its *only* arithmetic functions/predicates:

▶ **Definition 3** (structure encoding a word). *Each nonempty word $w = w_1 \dots w_n \in \Sigma^n$ on a fixed finite alphabet $\Sigma$ is represented by the first-order structure*
$\langle w \rangle \coloneqq ([1, n]; (Q_s)_{s\in\Sigma}, \mathtt{min}, \mathtt{max}, \mathtt{suc}, \mathtt{pred})$
*of domain $[1, n]$, monadic predicates $Q_s$, $s \in \Sigma$, `min` and `max` such that $Q_s(i) \iff w_i = s$, $\mathtt{min}(i) \iff i = 1$, and $\mathtt{max}(i) \iff i = n$, and unary functions `suc` and `pred` such that $\mathtt{suc}(i) = i + 1$ for $i < n$ and $\mathtt{suc}(n) = n$, $\mathtt{pred}(i) = i - 1$ for $i > 1$ and $\mathtt{pred}(1) = 1$.*
*Let $\mathcal{S}_\Sigma$ denote the signature $\{(Q_s)_{s\in\Sigma}, \mathtt{min}, \mathtt{max}, \mathtt{suc}, \mathtt{pred}\}$ of the structure $\langle w \rangle$.*

▶ **Notation 1.** *Let $x + k$ and $x - k$ abbreviate the terms $\mathtt{suc}^k(x)$ and $\mathtt{pred}^k(x)$, for a fixed integer $k \geq 0$. We will also use the intuitive abbreviations $x = 1$, $x = n$ and $x > k$, for a fixed integer $k \geq 1$, in place of the formulas $\mathtt{min}(x)$, $\mathtt{max}(x)$ and $\neg\,\mathtt{min}(x - (k - 1))$, respectively.*

## 2.3 Cellular automata and real-time

▶ **Definition 4** (1-CA and 2-OCA). *A $d$-*dimensional cellular automaton *(CA) is a triple $(\mathtt{S}, \mathcal{N}, \mathtt{f})$ where $\mathtt{S}$ is the finite set of* states*, $\mathcal{N} \subset \mathbb{Z}^d$ is the* neighborhood*, and $\mathtt{f} : \mathtt{S}^{|\mathcal{N}|} \to \mathtt{S}$ is the* transition function*. We are interested in the following two special cases:*
- 1-CA*: It is a* one-dimensional two-way *cellular automaton $(\mathtt{S}, \{-1, 0, 1\}, \mathtt{f})$, for which the state $\langle c, t \rangle$ of any cell $c$ at a time $t > 1$ is updated in this way:*
$\langle c, t \rangle = \mathtt{f}(\langle c - 1, t - 1 \rangle, \langle c, t - 1 \rangle, \langle c + 1, t - 1 \rangle)$.
- 2-OCA*: It is a* two-dimensional one-way *cellular automaton $(\mathtt{S}, \{(0, 0), (-1, 0), (0, -1)\}, \mathtt{f})$ for which the state $\langle c_1, c_2, t \rangle$ of any cell $(c_1, c_2)$ at a time $t > 1$ is updated in this way:*
$\langle c_1, c_2, t \rangle = \mathtt{f}(\langle c_1, c_2, t - 1 \rangle, \langle c_1 - 1, c_2, t - 1 \rangle, \langle c_1, c_2 - 1, t - 1 \rangle)$.

▶ **Definition 5** (permanent and quiescent states). *In a CA, a state $\sharp$ is* permanent *if a cell in state $\sharp$ remains in this state forever. A state $\lambda$ of a CA is* quiescent *if a cell in state $\lambda$ remains in this state as long as the states of its neighborhood cells are quiescent or permanent.*

▶ **Definition 6** (CA as a word acceptor). *A cellular automaton* $(\mathtt{S}, \mathcal{N}, \mathtt{f})$ *with an* input alphabet $\Sigma \subset \mathtt{S}$*, a* permanent state $\sharp$*, a* quiescent state $\lambda$*, and a set of* accepting states $\mathtt{S_{acc}} \subset \mathtt{S}$ *acts as a* word acceptor *if it operates on an input word* $w \in \Sigma^+$ *in respecting the following conditions (see Figure 1).*

***Input.*** *For a 1-CA, the i-th symbol of the input* $w = w_1 \dots w_n$ *is given to the cell* $i$ *at the initial time* 1*:* $\langle i, 1\rangle = w_i$*. All other cells are in the permanent state* $\sharp$*. For a 2-OCA, the i-th symbol of the input is given to the cell* $(i, 1)$ *at time* 1*:* $\langle i, 1, 1\rangle = w_i$*. At time* 1*, the cells* $(c_1, c_2) \in [1, n] \times [2, n]$ *are in the quiescent state* $\lambda$*, all other cells are in the permanent state* $\sharp$*.*

***Output.*** *One specific cell called the* output cell *gives the output, "accept" or "reject", of the computation. For a 1-CA, the output cell is the cell* 1*. For a 2-OCA, the output cell is* $(n, n)$*.*

***Acceptance.*** *An input word is* accepted *by a 1-CA (resp. 2-CA)* at time $t$ *if the output cell enters an accepting state at time* $t$*.*

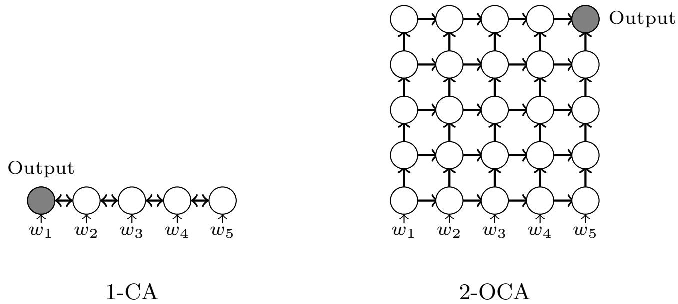


**Figure 1** Input and output of a CA acting as a word acceptor.

▶ **Definition 7** (RealTime1CA, RealTime2OCA). *A word is* accepted in real-time *by a 1-CA (resp. 2-OCA) if the word is accepted in minimal time for the output cell* 1 *(resp.* $(n, n)$*) to receive each of its letters. A language is* recognized in real-time *by a CA if it is the set of words that it accepts in real-time. The class* RealTime1CA *(resp.* RealTime2OCA*) is the class of languages recognized in real-time by a 1-CA (resp. 2-OCA).*

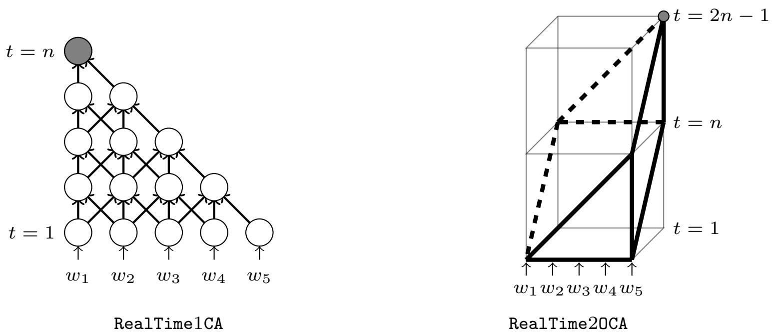


**Figure 2** Space-time diagrams of RealTime1CA and RealTime2OCA.

## 3 Real-time recognition of a unary conjunctive language

In this section, we prove our first main result:

▶ **Theorem 8.** $\mathtt{Conj}_1 \subseteq \mathtt{RealTime1CA}_1$.

## 3.1 Expressing inductively a unary conjunctive language in logic

The generating process of a unary conjunctive language is naturally expressed in the logic `pred-ESO-HORN`, an inductive Horn logic whose only function is the predecessor function.

▶ **Definition 9** (`pred-ESO-HORN`). *A formula of* `pred-ESO-HORN` *is a formula* $\Phi := \exists \mathbf{R} \forall x \forall y \psi(x,y)$ *where* $\mathbf{R}$ *is a finite set of binary predicates and* $\psi$ *is a conjunction of Horn clauses, of signature* $\mathcal{S}_\Sigma \cup \mathbf{R}$*, and of one the three following forms:*

- an input clause: $\mathtt{min}(x) \wedge (\neg)\mathtt{min}(y) \wedge Q_s(y) \to R(x,y)$ *with* $s \in \Sigma$ *and* $R \in \mathbf{R}$*;*
- a computation clause: $\delta_1 \wedge \ldots \wedge \delta_r \to R(x,y)$ *with* $R \in \mathbf{R}$ *and where each hypothesis* $\delta_h$ *is an atom* $S(x,y)$ *or a conjunction* $S(x-i, y-j) \wedge x > i \wedge y > j$*, with* $S \in \mathbf{R}$ *and* $i, j \geq 0$ *two integers such that* $i + j > 0$*;*
- a contradiction clause: $\mathtt{max}(x) \wedge \mathtt{max}(y) \wedge R(x,y) \to \bot$ *with* $R \in \mathbf{R}$.

By abuse of notation, let us also call `pred-ESO-HORN` the class of languages defined by a formula of `pred-ESO-HORN`.

▶ **Notation 2.** *We will freely use equalities (resp. inequalities)* $x = i$ *and* $y = j$ *(resp.* $x > i$*,* $y > j$*), for constants* $i, j$*, in our formulas since they can be easily defined in* `pred-ESO-HORN`*. For example, the binary predicate* $R^{x>2}$ *of intuitive meaning* $R^{x>2}(x,y) \iff x > 2$ *is defined inductively by the following clauses where* $R^{x=a}(x,y)$ *means* $x = a$*:*

- $\mathtt{min}(x) \to R^{x=1}(x,y)$*;* $x > 1 \wedge R^{x=1}(x-1,y) \to R^{x=2}(x,y)$*;*
- $x > 1 \wedge R^{x=2}(x-1,y) \to R^{x>2}(x,y)$*;* $x > 1 \wedge R^{x>2}(x-1,y) \to R^{x>2}(x,y)$*.*

*Also, some other arithmetic predicates easily defined in* `pred-ESO-HORN` *will be used. For example,* $y = 2x$ *can be replaced by the atom* $R^{y=2x}(x,y)$*, where* $R^{y=2x}$ *is defined by the following two clauses using the predicates* $R^{x=1}, R^{y=2}, R^{x>1}$ *and* $R^{y>2}$*:*

- $x = 1 \wedge y = 2 \to R^{y=2x}(x,y)$ *;* $x > 1 \wedge y > 2 \wedge R^{y=2x}(x-1,y-2) \to R^{y=2x}(x,y)$*.*

▶ **Notation 3.** *More generally, let* $R^{\rho(x,y)}$ *denote a binary predicate whose meaning is* $R^{\rho(x,y)}(x,y) \iff \rho(x,y)$*, for a property or a formula* $\rho(x,y)$*. We will also use a set of binary arithmetic predicates denoted by* $\mathbf{R}_{\mathtt{arith}}$*, which consists of* $R^{x=y}, R^{y=2x}$ *and* $R^{\rho(x,y)}$*, for* $\rho(x,y) := x \geq \lceil \frac{y}{2} \rceil$*, and the predicates used to define them in* `pred-ESO-HORN`*.*

Let us prove that for every unary conjunctive language, its complement can be defined in $\mathtt{pred\text{-}ESO\text{-}HORN}_1$.

▶ **Lemma 10.** *For each language* $L \subseteq a^+$*, if* $L \in \mathtt{Conj}_1$ *then* $a^+ \setminus L \in \mathtt{pred\text{-}ESO\text{-}HORN}$*.*

**Proof.** Let $G = (\{a\}, N, P, S)$ be a conjunctive grammar in binary normal form which generates $L$. For each $A \in N$ and each unary word $a^y$, we have, according to the length $y$, the following equivalences which will be the basis of our induction:

- if $y = 1$, then $a^y = a \in L(A) \iff$ the short rule $A \to a$ belongs to $P$;
- if $y > 1$, then $a^y \in L(A) \iff$ there is a long rule $A \to B_1C_1 \& \ldots \& B_mC_m$ in $P$ such that, for each $i \in \{1, \ldots, m\}$, there exists $x \geq \lceil \frac{y}{2} \rceil$ such that either $a^x \in L(B_i)$ and $a^{y-x} \in L(C_i)$, or $a^{y-x} \in L(B_i)$ and $a^x \in L(C_i)$.

We want to construct a first-order formula $\forall x \forall y \psi_G(x,y)$ of signature $\mathcal{S}_\Sigma \cup \mathbf{R}$, for $\Sigma := \{a\}$ and the set of binary predicates $\mathbf{R} := \{\mathtt{Maj}_A, \mathtt{Min}_A \mid A \in N\} \cup \{\mathtt{Sum}_{BC} \mid B, C \in N\} \cup \mathbf{R}_{\mathtt{arith}}$ so that the formula $\Phi_G := \exists \mathbf{R} \forall x \forall y \psi_G$ belongs to `pred-ESO-HORN` and defines the language $a^+ \setminus L$. The intuitive meanings of the predicates $\mathtt{Maj}_A, \mathtt{Min}_A$ and $\mathtt{Sum}_{BC}$ are as follows:

- $\mathtt{Maj}_A(x,y) \iff \lceil \frac{y}{2} \rceil \leq x \leq y$ and $a^x \in L(A)$;
- $\mathtt{Min}_A(x,y) \iff \lceil \frac{y}{2} \rceil \leq x < y$ and $a^{y-x} \in L(A)$ ;

- $\mathtt{Sum}_{\mathtt{BC}}(x,y) \iff$ there is some $x'$ with $\lceil \frac{y}{2} \rceil \leq x' \leq x$ such that either $a^{x'} \in L(B)$ and $a^{y-x'} \in L(C)$, or $a^{y-x'} \in L(B)$ and $a^{x'} \in L(C)$.

Note that for $x = y$, the above equivalence for $\mathtt{Maj}_A$ implies $\mathtt{Maj}_A(x,y) \iff a^y \in L(A)$.

Let us give and justify a list of Horn clauses whose conjunction $\psi'_G$ defines the predicates $\mathtt{Maj}_A$, $\mathtt{Min}_A$ and $\mathtt{Sum}_{BC}$, using the arithmetic predicates of $\mathbf{R}_{\mathtt{arith}}$ (see Notations 2 and 3), namely $R^{x=y}$, $R^{y=2x}$ and $R^{\rho(x,y)}$, for $\rho(x,y) \coloneqq x \geq \lceil \frac{y}{2} \rceil$.

**Short rules.** Each rule $A \to a$ of $P$ is expressed by the input clause:

- $\mathtt{min}(x) \wedge \mathtt{min}(y) \wedge Q_a(y) \to \mathtt{Maj}_A(x,y)$.

**Induction on the length $y$.** If we have for $y > 1$ the inequalities $\lceil \frac{y-1}{2} \rceil \leq x \leq y-1$ and $x \geq \lceil \frac{y}{2} \rceil$ then $\lceil \frac{y}{2} \rceil \leq x \leq y$. This justifies the clause:

- $y > 1 \wedge \mathtt{Maj}_A(x, y-1) \wedge x \geq \lceil \frac{y}{2} \rceil \to \mathtt{Maj}_A(x,y)$ for all $A \in N$.

For $y > 1$ and $y = 2x$, we have $a^x = a^{y-x}$ and $\lceil \frac{y}{2} \rceil \leq x < y$. This justifies the clause:

- $y > 1 \wedge \mathtt{Maj}_A(x, y-1) \wedge y = 2x \to \mathtt{Min}_A(x,y)$ for all $A \in N$.

If for $x, y > 1$ we have the inequalities $\lceil \frac{y-1}{2} \rceil \leq x-1 < y-1$, then $\lceil \frac{y}{2} \rceil \leq x < y$. Moreover, $a^{(y-1)-(x-1)} = a^{y-x}$. This justifies the clause:

- $x > 1 \wedge y > 1 \wedge \mathtt{Min}_A(x-1, y-1) \to \mathtt{Min}_A(x,y)$ for all $A \in N$.

**Concatenation.** For all $B, C \in N$, it is clear that the concatenation predicate $\mathtt{Sum}_{\mathtt{BC}}$ is defined inductively by the following three clauses:

- *initialization:* $\mathtt{Maj}_B(x,y) \wedge \mathtt{Min}_C(x,y) \to \mathtt{Sum}_{\mathtt{BC}}(x,y)$ ; $\mathtt{Min}_B(x,y) \wedge \mathtt{Maj}_C(x,y) \to \mathtt{Sum}_{\mathtt{BC}}(x,y)$;
- *induction:* $\neg\mathtt{min}(x) \wedge \mathtt{Sum}_{\mathtt{BC}}(x-1,y) \to \mathtt{Sum}_{\mathtt{BC}}(x,y)$.

**Long rules.** Each rule $A \to B_1C_1 \& \dots \& B_mC_m$ of $P$ is expressed by the clause:

- $x = y \wedge \mathtt{Sum}_{\mathtt{B_1C_1}}(x,y) \wedge \dots \wedge \mathtt{Sum}_{\mathtt{B_mC_m}}(x,y) \to \mathtt{Maj}_A(x,y)$.

Thus, the formula $\forall x \forall y \psi'_G$ where $\psi'_G$ is the conjunction of the above clauses defines the predicates $\mathtt{Maj}_A, \mathtt{Min}_A$, and $\mathtt{Sum}_{BC}$.

**Definition of $a^+ \setminus L$.** We have the equivalence $\mathtt{Maj}_S(n,n) \iff a^n \in L(S) \iff a^n \in L$. Therefore, the following contradiction clause expresses $a^n \notin L$:

- $\gamma_S \coloneqq \mathtt{max}(x) \wedge \mathtt{max}(y) \wedge \mathtt{Maj}_S(x,y) \to \bot$.

Finally, observe that the formula $\Phi_G \coloneqq \exists \mathbf{R} \forall x \forall y \psi_G$ where $\psi_G$ is $\gamma_{\mathtt{arith}} \wedge \psi'_G \wedge \gamma_S$ and $\gamma_{\mathtt{arith}}$ is the conjunction of clauses that defines the arithmetic predicates of $\mathbf{R}_{\mathtt{arith}}$, belongs to pred-ESO-HORN. Since we have $\langle a^n \rangle \models \Phi_G \iff a^n \notin L$, as justified above, then the language $a^+ \setminus L$ belongs to pred-ESO-HORN, as claimed. ◀

## 3.2 Equivalence of logic with grid-circuits

We introduce the *grid-circuit* as an intermediate object between our logic and the real-time cellular automaton: see Figure 3.

▶ **Definition 11.** *A* grid-circuit *is a tuple* $\mathcal{C} \coloneqq (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathtt{acc}}, \mathtt{g})$ *where*

- $\Sigma$ *is the* input alphabet *and* $(\mathtt{Input}_n)_{n>0}$ *is the family of* input functions $\mathtt{Input}_n$ : $\Sigma^n \times [1,n]^2 \to \Sigma \cup \{\$\}$ *such that, for* $w = w_1 \dots w_n \in \Sigma^n$, $\mathtt{Input}_n(w,x,y) = w_y$ *if* $x = 1$ *and* $\mathtt{Input}_n(w,x,y) = \$$ *otherwise,*
- $\mathbf{Q} \cup \{\sharp\}$ *is the finite set of* states *and* $\mathbf{Q}_{\mathtt{acc}} \subseteq \mathbf{Q}$ *is the subset of* accepting states*,*
- $\mathtt{g} : (\mathbf{Q} \cup \{\sharp\})^2 \times (\Sigma \cup \{\$\}) \to \mathbf{Q}$ *is the* transition function*.*

▶ **Definition 12** (computation of a grid-circuit). *The computation $\mathcal{C}_w$ of a grid-circuit $\mathcal{C} := (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathtt{acc}}, \mathtt{g})$ on a $w = w_1 \dots w_n \in \Sigma^n$ is a regular grid of $(n+1)^2$ sites $(x, y) \in [0, n]^2$, each in a state $\langle x, y\rangle \in \mathbf{Q} \cup \{\sharp\}$ computed inductively:*

- *each site in $\{0\} \times [0, n]$ or $[0, n] \times \{0\}$ is in the particular state $\sharp$;*
- *the state of each site $(x, y) \in [1, n]^2$ is $\langle x, y\rangle = \mathtt{g}(\langle x, y-1\rangle, \langle x-1, y\rangle, \mathtt{Input}_n(w, x, y))$.*

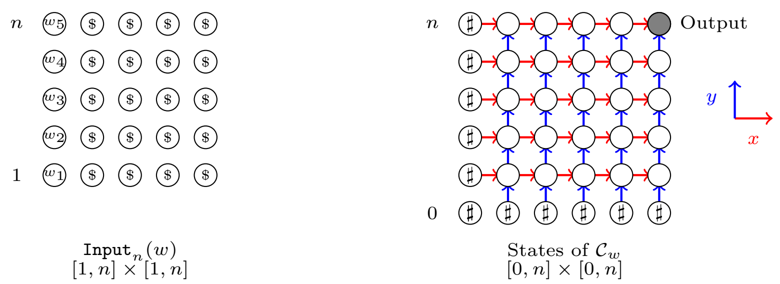


**Figure 3** The grid-circuit.

A word $w = w_1 \dots w_n \in \Sigma^n$ is *accepted* by the grid-circuit $\mathcal{C}$ if the output state $\langle n, n\rangle$ of $\mathcal{C}_w$ belongs to $\mathbf{Q}_{\mathtt{acc}}$. The language *recognized* by $\mathcal{C}$ is the set of words it accepts. We denote by `Grid` the class of languages recognized by a grid-circuit.

Actually, our predecessor Horn logic is equivalent to grid-circuits.

▶ **Lemma 13** ([12]). `pred-ESO-HORN` $=$ `Grid`.

**Proof.** In some sense, a grid-circuit is the "normalized form" of a formula of `pred-ESO-HORN`. So, the inclusion `Grid` $\subseteq$ `pred-ESO-HORN` is proved straightforwardly.

The first step of the proof of the converse inclusion `pred-ESO-HORN` $\subseteq$ `Grid` is to show that every formula $\Phi := \exists \mathbf{R} \forall x \forall y \psi(x, y)$ in `pred-ESO-HORN` is equivalent to a formula $\Phi' \in$ `pred-ESO-HORN` in which the only hypotheses of computation clauses are atoms $S(x, y)$ and conjunctions $S(x-1, y) \wedge x > 1$ and $S(x, y-1) \wedge y > 1$ .

**Elimination of atoms $R(x-i, y-j)$ for $i+j > 1$.** The idea is to introduce new "shift" predicates $R^{x-i', y-j'}$ for fixed integers $i', j' > 0$ with the intuitive meaning:
$R^{x-i', y-j'}(x, y) \iff R(x-i', y-j') \wedge x > i' \wedge y > j'$.
Let us explain the method by an example. Assume we have in $\psi$ the Horn clause
(1) $x > 3 \wedge y > 2 \wedge S(x-3, y-2) \to T(x, y)$. This clause is replaced by the clause
(2) $S^{x-2, y-2}(x-1, y) \wedge x > 1 \to T(x, y)$
for which the predicates $S^{x-1}$, $S^{x-2}$, $S^{x-2, y-1}$ and $S^{x-2, y-2}$ are defined by the respective clauses: $x > 1 \wedge S(x-1, y) \to S^{x-1}(x, y)$, $x > 1 \wedge S^{x-1}(x-1, y) \to S^{x-2}(x, y)$, $y > 1 \wedge S^{x-2}(x, y-1) \to S^{x-2, y-1}(x, y)$, and $y > 1 \wedge S^{x-2, y-1}(x, y-1) \to S^{x-2, y-2}(x, y)$, which imply together the clause $x > 2 \wedge y > 2 \wedge S(x-2, y-2) \to S^{x-2, y-2}(x, y)$ and then also $x > 3 \wedge y > 2 \wedge S(x-3, y-2) \to S^{x-2, y-2}(x-1, y)$.

It is clear that the formula $\Phi := \exists \mathbf{R} \forall x \forall y \psi$ is equivalent to the formula $\Phi' := \exists \mathbf{R}' \forall x \forall y \psi'$ where $\mathbf{R}' := \mathbf{R} \cup \{S^{x-1}, S^{x-2}, S^{x-2, y-1}, S^{x-2, y-2}\}$ and $\psi'$ is the conjunction $\psi_{\mathtt{replace}} \wedge \psi_{\mathtt{def}}$, where $\psi_{\mathtt{replace}}$ is the formula $\psi$ in which clause (1) is replaced by clause (2), and $\psi_{\mathtt{def}}$ is the conjunction of the above clauses defining the new predicates of $\mathbf{R}'$.

Thus, any formula $\Phi \in$ `pred-ESO-HORN` is equivalent to a formula $\Phi' \in$ `pred-ESO-HORN` whose computation clauses only contain hypotheses of the following three forms:
$R(x-1, y) \wedge x > 1$ ; $R(x, y-1) \wedge y > 1$ ; $R(x, y)$. The next step is to eliminate these $R(x, y)$.

**Elimination of hypotheses $R(x,y)$.** (sketch of proof): The first idea is to group together in each computation clause the hypothesis atoms of the form $R(x,y)$ and the conclusion of the clause. As a result, the formula can be rewritten in the form

$$\Phi \coloneqq \exists \mathbf{R} \forall x \forall y \left[ \bigwedge_i C_i(x,y) \wedge \bigwedge_{i \in [1,k]} (\alpha_i(x,y) \rightarrow \theta_i(x,y)) \right]$$

where the $C_i$'s are the input clauses and the contradiction clauses, and each computation clause is written in the form $\alpha_i(x,y) \rightarrow \theta_i(x,y)$, where $\alpha_i(x,y)$ is a conjunction of formulas of the only forms $R(x-1,y) \wedge x > 1$, $R(x,y-1) \wedge y > 1$, and $\theta_i(x,y)$ is a Horn clause in which *all* atoms are of the form $R(x,y)$.

The second idea is to "solve" the Horn clauses $\theta_i$ according to the input clauses and *all the possible* conjunctions of hypotheses $\alpha_i$ that may be true. Notice the two following facts: the hypotheses of the input clauses are input literals and the conjuncts of the $\alpha_i$'s are of the only forms $R(x-1,y) \wedge x > 1$, $R(x,y-1) \wedge y > 1$. So, we can prove by induction on the sum $x+y$ that the obtained formula $\Phi'$ in which no atom $R(x,y)$ appears as a clause hypothesis, is equivalent to the above formula $\Phi$. The complete proof is given in Appendix A.

**Transformation of the formula into a grid-circuit.** Let $\mathbf{R} = \{R_1, \dots, R_m\}$ denote the set of binary predicates of the formula. By a case separation of the clauses, it is easy to transform the formula into an equivalent formula $\Phi \coloneqq \exists \mathbf{R} \forall x \forall y \psi$ where $\psi$ is a conjunction of clauses of the following forms (a-e), in which $s \in \Sigma$, $j \in [1,m]$, and $A, B$ are (possibly empty) subsets of $[1,m]$:

**(a)** $x = 1 \wedge y = 1 \wedge Q_s(y) \rightarrow R_j(x,y)$;
**(b)** $x = 1 \wedge y > 1 \wedge Q_s(y) \wedge \bigwedge_{i \in A} R_i(x, y-1) \rightarrow R_j(x,y)$;
**(c)** $x > 1 \wedge y = 1 \wedge \bigwedge_{i \in A} R_i(x-1, y) \rightarrow R_j(x,y)$;
**(d)** $x > 1 \wedge y > 1 \wedge \bigwedge_{i \in A} R_i(x-1, y) \wedge \bigwedge_{i \in B} R_i(x, y-1) \rightarrow R_j(x,y)$;
**(e)** $x = n \wedge y = n \wedge R_j(x,y) \rightarrow \bot$.

Now, transform this formula into a grid-circuit $\mathcal{C} \coloneqq (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathtt{acc}}, \mathtt{g})$. The idea is that the state of a site $(x,y) \in [1,n]^2$ is the set of predicates $R_i$ such that $R_i(x,y)$ is true. Let $\mathbf{Q}$ be the power set of the set of $\mathbf{R}$ indices: $\mathbf{Q} \coloneqq \mathcal{P}([1,m])$. There are four types of transition (a-d) which mimic the clauses (a-d) above. These are, for $s \in \Sigma$ and $q, q' \in \mathbf{Q}$:

**(a)** $\mathtt{g}(\sharp, \sharp, s) = \{j \in [1,m] \mid \text{there is a clause (a) with } Q_s, \text{ and conclusion } R_j(x,y)\}$;
**(b)** $\mathtt{g}(q, \sharp, s) = \{j \in [1,m] \mid \text{there is a clause (b) with } Q_s, \text{ and } A \subseteq q, \text{ and conclusion } R_j(x,y)\}$;
**(c)** $\mathtt{g}(\sharp, q, \$) = \{j \in [1,m] \mid \text{there is a clause (c) with } A \subseteq q, \text{ and conclusion } R_j(x,y)\}$;
**(d)** $\mathtt{g}(q, q', \$) = \{j \in [1,m] \mid \exists \text{ a clause (d) with } A \subseteq q,\ B \subseteq q', \text{ and conclusion } R_j(x,y)\}$.

Of course, the set of accepting states of $\mathcal{C}$ is determined by the contradiction clauses (e): $\mathbf{Q}_{\mathtt{acc}} \coloneqq \{q \in \mathbf{Q} \mid q \text{ contains no } j \text{ such that } R_j \text{ occurs in a clause (e)}\}$.

We can easily check the equivalence, for each $w \in \Sigma^+$: $\langle w \rangle \models \Phi \iff \mathcal{C}$ accepts $w$. Therefore, the inclusion `pred-ESO-HORN` $\subseteq$ `Grid` is proved. ◀

## 3.3 Grid-circuits are equivalent to real-time 1-CA

▶ **Lemma 14.** *[12]* `Grid` = `RealTime1CA`.

**Proof.** Figure 4 shows how `Grid` is simulated on `RealTime1CA` and Figure 5 shows how `RealTime1CA` is simulated on `Grid`. The proof is detailed in Appendix B. ◀

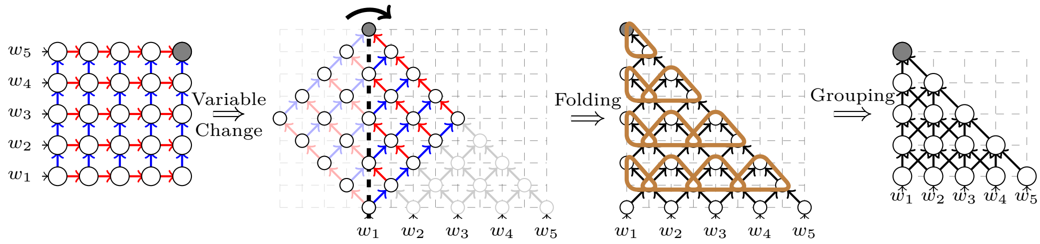


**Figure 4** Simulation of `Grid` on `RealTime1CA`.

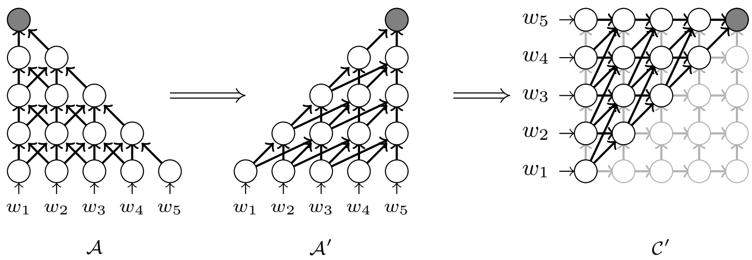


**Figure 5** Simulation of `RealTime1CA` on the grid-circuit.

**Proof of Theorem 8.** Lemmas 13 and 14 give us the following equalities of classes: $\texttt{pred-ESO-HORN} = \texttt{Grid} = \texttt{RealTime1CA}$. These equalities trivially hold when restricted to unary languages: $\texttt{pred-ESO-HORN}_1 = \texttt{Grid}_1 = \texttt{RealTime1CA}_1$.

From the fact that the class $\texttt{RealTime1CA}_1$ is closed under complement and from Lemma 10, we deduce $\texttt{Conj}_1 \subseteq \texttt{pred-ESO-HORN}_1 = \texttt{Grid}_1 = \texttt{RealTime1CA}_1$. ◀

## 4 Real-time recognition of a conjunctive language: the general case

Recall the inclusions[4] $\texttt{RealTime1CA} \subseteq \texttt{RealTime2OCA} \subseteq \texttt{RealTime2SOCA}$.

Our second main result strengthens the inclusion $\texttt{CFL} \subseteq \texttt{RealTime2SOCA}$ of Terrier [29]:

▶ **Theorem 15.** $\texttt{Conj} \subseteq \texttt{RealTime2OCA}$.

### 4.1 Expressing a conjunctive language in logic: the general case

The generating process of a conjunctive language is naturally expressed in the Horn logic `incl-pred-ESO-HORN`. This is a hybrid logic with three first-order variables $x, y, z$, whose name means that it makes inductions on the variable interval $[x, y]$, by *inclusion*, and on the individual variable $z$, by *predecessor*.

▶ **Definition 16** (`incl-pred-ESO-HORN`). *A formula of* `incl-pred-ESO-HORN` *is a formula* $\Phi := \exists \mathbf{R} \forall x \forall y \forall z \psi(x, y, z)$ *where* $\mathbf{R}$ *is a finite set of ternary predicates, and* $\psi$ *is a conjunction of Horn clauses, of signature*[5] $\mathcal{S}_\Sigma \cup \mathbf{R} \cup \{=, \leq\}$*, and of the three following forms:*

- *an* input clause: $x = y \wedge \texttt{min}(z) \wedge Q_s(x) \to R(x, y, z)$ *with* $s \in \Sigma$ *and* $R \in \mathbf{R}$*;*

---

[4] Recall that `RealTime2SOCA` is the class of languages recognized by *sequential* two-dimensional one-way cellular automata in real-time: this is the minimal time, $3n - 1$, for the output cell $(n, n)$ to receive the $n$ letters of the input word, communicated sequentially by the input cell $(1, 1)$.

[5] This definition must consider $=$ and $\leq$ as primitive symbols.

- *a* computation clause: $\delta_1 \wedge \ldots \wedge \delta_r \to R(x,y,z)$ *with* $R \in \mathbf{R}$ *and where each hypothesis* $\delta_h$ *is an atom* $S(x,y,z)$ *or a conjunction* $S(x+i, y-k, z-k) \wedge x+i \leq y-j \wedge z > k$ *with* $S \in \mathbf{R}$ *and* $i,j,k \geq 0$ *three integers such that* $i+j+k>0$*;*
- *a* contradiction clause: $\mathtt{min}(x) \wedge \mathtt{max}(y) \wedge \mathtt{max}(z) \wedge R(x,y,z) \to \perp$ *with* $R \in \mathbf{R}$*.*

Let us also call `incl-pred-ESO-HORN` the class of languages defined by a formula of `incl-pred-ESO-HORN`.

▶ **Lemma 17.** *For each language* $L \subseteq \Sigma^+$*, if* $L \in \mathtt{Conj}$*, then* $\Sigma^+ \setminus L \in \mathtt{incl\text{-}pred\text{-}ESO\text{-}HORN}$*.*

**Proof.** The proof is a variation (an extension) of the proof of the same result, Lemma 10, in the unary case. This is why we insist on the differences. Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar in binary normal form which generates $L$ and let $w$ be a word $w = w_1 \ldots w_n \in \Sigma^+$. For each $A \in N$ and each factor $w_{x,y} := w_x \ldots w_y$, we have, according to the length $y-x+1$ of $w_{x,y}$, the following equivalences which will be the basis of our induction:

- if $x = y$, then $w_{x,y} \in L(A) \iff$ the short rule $A \to w_x$ belongs to $P$;
- if $x < y$, then $w_{x,y} \in L(A) \iff$ there is a long rule $A \to B_1C_1 \& \ldots \& B_mC_m$ in $P$ such that, for each $i \in \{1, \ldots, m\}$, there exists $z \geq \lceil (y-x+1)/2 \rceil$ such that either $w_{x,x+z-1} \in L(B_i)$ and $w_{x+z,y} \in L(C_i)$, or $w_{x,y-z} \in L(B_i)$ and $w_{y-z+1,y} \in L(C_i)$.

Thus, a double induction is performed, on the index interval $[x,y]$ of a factor $w_{x,y}$ and the maximal $z$ among the lengths of the two sub-factors $u, v$ of the $m$ decompositions $w_{x,y} = uv$, $u \in L(B_i)$, $v \in L(C_i)$, for a long rule. This is naturally expressed in the logic `incl-pred-ESO-HORN`.

We want to construct a first-order formula $\forall x \forall y \forall z \psi_G$ of signature $\mathcal{S}_\Sigma \cup \mathbf{R} \cup \{=, \leq\}$, for the set of *ternary* predicates $\mathbf{R} := \{\mathtt{Pref}^{\mathtt{Maj}}_A, \mathtt{Pref}^{\mathtt{Min}}_A, \mathtt{Suff}^{\mathtt{Maj}}_A, \mathtt{Suff}^{\mathtt{Min}}_A \mid A \in N\} \cup \{\mathtt{Concat}_{BC} \mid B, C \in N\} \cup \mathbf{R}_{\mathtt{arith}}$, so that the formula $\Phi_G := \exists \mathbf{R} \forall x \forall y \forall z \psi_G$ belongs to `incl-pred-ESO-HORN` and defines the language $\Sigma^+ \setminus L$. The intuitive meanings of the predicates $\mathtt{Pref}^{\mathtt{Maj}}_A, \mathtt{Pref}^{\mathtt{Min}}_A, \mathtt{Suff}^{\mathtt{Maj}}_A, \mathtt{Suff}^{\mathtt{Min}}_A$ and $\mathtt{Concat}_{BC}$ are as follows:

- $\mathtt{Pref}^{\mathtt{Maj}}_A(x,y,z) \iff \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x+1$ and $w_{x,x+z-1} \in L(A)$;
- $\mathtt{Pref}^{\mathtt{Min}}_A(x,y,z) \iff \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$ and $w_{x,y-z} \in L(A)$;
- $\mathtt{Suff}^{\mathtt{Maj}}_A(x,y,z) \iff \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x+1$ and $w_{y-z+1,y} \in L(A)$;
- $\mathtt{Suff}^{\mathtt{Min}}_A(x,y,z) \iff \left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$ and $w_{x+z,y} \in L(A)$;
- $\mathtt{Concat}_{BC}(x,y,z) \iff$ there is some $z'$ with $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z' \leq z$ such that either $w_{x,x+z'-1} \in L(B)$ and $w_{x+z',y} \in L(C)$, or $w_{x,y-z'} \in L(B)$ and $w_{y-z'+1,y} \in L(C)$.

Note that the above equivalences for $\mathtt{Pref}^{\mathtt{Maj}}_A$ and $\mathtt{Suff}^{\mathtt{Maj}}_A$ imply in the particular case $z = y-x+1$ the equivalences $\mathtt{Pref}^{\mathtt{Maj}}_A(x,y,z) \iff \mathtt{Suff}^{\mathtt{Maj}}_A(x,y,z) \iff w_{x,y} \in L(A)$.

Let us give and justify a list of Horn clauses whose conjunction $\psi'_G$ defines the predicates $\mathtt{Pref}^{\mathtt{Maj}}_A, \mathtt{Pref}^{\mathtt{Min}}_A, \mathtt{Suff}^{\mathtt{Maj}}_A, \mathtt{Suff}^{\mathtt{Min}}_A$ and $\mathtt{Concat}_{BC}$, using the arithmetic predicates $z = y-x+1$, $y-x+1 = 2z$, and $z \geq \left\lceil \frac{y-x+1}{2} \right\rceil$ easily defined in `incl-pred-ESO-HORN`.

**Short rules.** Each rule $A \to s$ of $P$ is expressed by the two clauses:

- $x = y \wedge z = 1 \wedge Q_s(x) \to \mathtt{Pref}^{\mathtt{Maj}}_A(x,y,z)$ ; $x = y \wedge z = 1 \wedge Q_s(x) \to \mathtt{Suff}^{\mathtt{Maj}}_A(x,y,z)$.

**Induction for prefixes.** If we have for $x < y$ the inequalities $\left\lceil \frac{(y-1)-x+1}{2} \right\rceil \leq z \leq (y-1)-x+1$ and $z \geq \left\lceil \frac{y-x+1}{2} \right\rceil$ then $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x+1$. This justifies the clause:

- $x \leq y-1 \wedge \mathtt{Pref}^{\mathtt{Maj}}_A(x,y-1,z) \wedge z \geq \left\lceil \frac{y-x+1}{2} \right\rceil \to \mathtt{Pref}^{\mathtt{Maj}}_A(x,y,z)$, for all $A \in N$.

For $x < y$ and $y-x+1 = 2z$, we have $w_{x,x+z-1} = w_{x,y-z}$ and $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$. This justifies the clause:

- $x \leq y-1 \wedge \mathtt{Pref}^{\mathtt{Maj}}_A(x,y-1,z) \wedge y-x+1 = 2z \to \mathtt{Pref}^{\mathtt{Min}}_A(x,y,z)$, for all $A \in N$.

For $x < y$ and $z > 1$ and $\left\lceil \frac{(y-1)-x+1}{2} \right\rceil \leq z-1 \leq (y-1)-x$, we have $\left\lceil \frac{y-x+1}{2} \right\rceil \leq z \leq y-x$. This justifies the clause:

- $x \leq y-1 \wedge z > 1 \wedge \mathtt{Pref}_A^{\mathtt{Min}}(x, y-1, z-1) \rightarrow \mathtt{Pref}_A^{\mathtt{Min}}(x, y, z)$, for all $A \in N$.

**Induction for suffixes.** As this induction is symmetric to the one for prefixes, we do not justify the following list of induction clauses for the predicates $\mathtt{Suff}_A^{\mathtt{Maj}}$ and $\mathtt{Suff}_A^{\mathtt{Min}}$, $A \in N$:

- $x+1 \leq y \wedge \mathtt{Suff}_A^{\mathtt{Maj}}(x+1, y, z) \wedge z \geq \left\lceil \frac{y-x+1}{2} \right\rceil \rightarrow \mathtt{Suff}_A^{\mathtt{Maj}}(x, y, z)$;
- $x+1 \leq y \wedge \mathtt{Suff}_A^{\mathtt{Maj}}(x+1, y, z) \wedge y-x+1 = 2z \rightarrow \mathtt{Suff}_A^{\mathtt{Min}}(x, y, z)$;
- $x+1 \leq y \wedge z > 1 \wedge \mathtt{Suff}_A^{\mathtt{Min}}(x+1, y, z-1) \rightarrow \mathtt{Suff}_A^{\mathtt{Min}}(x, y, z)$.

**Concatenation.** For all $B, C \in N$, it is clear that the concatenation predicate $\mathtt{Concat}_{BC}$ is defined inductively by the following three clauses:

- *initialization:* $\mathtt{Pref}_B^{\mathtt{Maj}}(x, y, z) \wedge \mathtt{Suff}_C^{\mathtt{Min}}(x, y, z) \rightarrow \mathtt{Concat}_{BC}(x, y, z)$;
  $\mathtt{Pref}_B^{\mathtt{Min}}(x, y, z) \wedge \mathtt{Suff}_C^{\mathtt{Maj}}(x, y, z) \rightarrow \mathtt{Concat}_{BC}(x, y, z)$;
- *induction:* $z > 1 \wedge \mathtt{Concat}_{BC}(x, y, z-1) \rightarrow \mathtt{Concat}_{BC}(x, y, z)$.

**Long rules.** Each rule $A \rightarrow B_1C_1\& \ldots \&B_mC_m$ of $P$ is expressed by the two clauses:

- $z = y-x+1 \wedge \mathtt{Concat}_{B_1C_1}(x, y, z) \wedge \cdots \wedge \mathtt{Concat}_{B_mC_m}(x, y, z) \rightarrow \mathtt{Pref}_A^{\mathtt{Maj}}(x, y, z)$;
- $z = y-x+1 \wedge \mathtt{Concat}_{B_1C_1}(x, y, z) \wedge \cdots \wedge \mathtt{Concat}_{B_mC_m}(x, y, z) \rightarrow \mathtt{Suff}_A^{\mathtt{Maj}}(x, y, z)$.

Thus, the formula $\forall x \forall y \forall z \psi'_G$ where $\psi'_G$ is the conjunction of the above clauses defines the predicates $\mathtt{Pref}_A^{\mathtt{Maj}}, \mathtt{Pref}_A^{\mathtt{Min}}, \mathtt{Suff}_A^{\mathtt{Maj}}, \mathtt{Suff}_A^{\mathtt{Min}}$, and $\mathtt{Concat}_{BC}$.

**Definition of $\Sigma^+ \setminus L$.** We have the equivalence $\mathtt{Pref}_S^{\mathtt{Maj}}(1, n, n) \iff w \in L(S) \iff w \in L$. Therefore, the following contradiction clause expresses $w \notin L$:

- $\gamma_S := \mathtt{min}(x) \wedge \mathtt{max}(y) \wedge \mathtt{max}(z) \wedge \mathtt{Pref}_S^{\mathtt{Maj}}(x, y, z) \rightarrow \bot$.

Finally, observe that the formula $\Phi_G := \exists \mathbf{R} \forall x \forall y \forall z \psi_G$ where $\psi_G$ is $\gamma_{\mathtt{arith}} \wedge \psi'_G \wedge \gamma_S$ and $\gamma_{\mathtt{arith}}$ is the conjunction of clauses that define the arithmetic predicates, belongs to incl-pred-ESO-HORN. Since we have $\langle w \rangle \models \Phi_G \iff w \notin L$, as justified above, then the language $\Sigma^+ \setminus L$ belongs to incl-pred-ESO-HORN, as claimed. ◀

## 4.2 Equivalence of logic with cube-circuits

We now introduce the *cube-circuit*, an extension of the grid-circuit to three dimensions. It will make the link between our logic incl-pred-ESO-HORN and the class RealTime2OCA.

▶ **Definition 18.** *A* cube-circuit *is a tuple* $\mathcal{C} := (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathtt{acc}}, \mathtt{g})$ *where*

- $\Sigma$ *is the* input alphabet *and* $(\mathtt{Input}_n)_{n>0}$ *is the family of* input functions $\mathtt{Input}_n : \Sigma^n \times [1, n]^3 \rightarrow \Sigma \cup \{\$\}$ *such that, for* $w = w_1 \ldots w_n \in \Sigma^n$, $\mathtt{Input}_n(w, x, y, z) = w_x$ *if* $x = y$ *and* $z = 1$*, and* $\mathtt{Input}_n(w, x, y, z) = \$$ *otherwise,*
- $\mathbf{Q} \cup \{\sharp\}$ *is the finite set of* states *and* $\mathbf{Q}_{\mathtt{acc}} \subseteq \mathbf{Q}$ *is the subset of* accepting states*,*
- $\mathtt{g} : (\mathbf{Q} \cup \{\sharp\})^3 \times (\Sigma \cup \{\$\}) \rightarrow \mathbf{Q}$ *is the* transition function.

▶ **Definition 19** (computation of a cube-circuit)**.** *The* computation $\mathcal{C}_w$ *of a cube-circuit* $\mathcal{C} := (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathtt{acc}}, \mathtt{g})$ *on a word* $w = w_1 \ldots w_n \in \Sigma^n$ *is a grid of* $(n+1)^3$ *sites* $(x, y, z) \in [1, n+1] \times [0, n]^2$*, each in a state* $\langle x, y, z \rangle \in \mathbf{Q} \cup \{\sharp\}$ *computed inductively:*

- *each site* $(x, y, z)$ *such that* $x > y$ *or* $z = 0$ *is in the state* $\sharp$*;*
- *the state of each site* $(x, y, z) \in [1, n]^3$ *such that* $x \leq y$ *and* $z > 0$ *is* $\langle x, y \rangle = \mathtt{g}(\langle x+1, y, z \rangle, \langle x, y-1, z \rangle, \langle x, y, z-1 \rangle, \mathtt{Input}_n(w, x, y, z))$.

A word $w = w_1 \dots w_n \in \Sigma^n$ is *accepted* by the cube-circuit $\mathcal{C}$ if the output state $\langle 1, n, n\rangle$ of $\mathcal{C}_w$ belongs to $\mathbf{Q}_{\mathtt{acc}}$. The language *recognized* by $\mathcal{C}$ is the set of words it accepts. We denote by `Cube` the class of languages recognized by a cube-circuit.

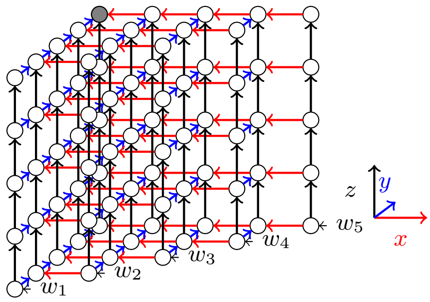


**Figure 6** The cube-circuit.

Actually, the logic `incl-pred-ESO-HORN` is equivalent to cube-circuits.

▶ **Lemma 20.** `incl-pred-ESO-HORN` = `Cube`.

**Proof.** The proof is similar to that of `pred-ESO-HORN` = `Grid` (Lemma 13). The cube-circuit can be seen as the "normalized form" of a formula of `incl-pred-ESO-HORN`, proving the inclusion `Cube` ⊆ `incl-pred-ESO-HORN`. The proof of the inverse inclusion is divided into the same three steps as for Lemma 13, which must be adapted to three variables: 1) elimination of atoms $R(x+i, y-j, z-k)$ for $i+j+k > 1$ (instead of elimination of atoms $R(x-i, y-j)$ for $i+j > 1$); 2) elimination of hypotheses $R(x, y, z)$ (instead of elimination of hypotheses $R(x, y)$); 3) transformation of the resulting formula into a cube-circuit.

Steps 1 and 2 are adapted straightforwardly. Let us describe in detail step 3. Let $\mathbf{R} = \{R_1, \dots, R_m\}$ denote the set of ternary predicates of the formula resulting from step 2. By a case separation of the clauses, it is easy to transform this formula into an equivalent formula $\Phi := \exists \mathbf{R} \forall x \forall y \forall z \psi$ where $\psi$ is a conjunction of clauses of the following forms (a-e), in which $s \in \Sigma$, $j \in [1, m]$, and $A, B, C$ are (possibly empty) subsets of $[1, m]$:

**(a)** $x = y \land z = 1 \land Q_s(x) \to R_j(x, y, z)$;
**(b)** $x < y \land z = 1 \land \bigwedge_{i \in A} R_i(x+1, y, z) \land \bigwedge_{i \in B} R_i(x, y-1, z) \to R_j(x, y, z)$;
**(c)** $x = y \land z > 1 \land \bigwedge_{i \in A} R_i(x, y, z-1) \to R_j(x, y, z)$;
**(d)** $x < y \land z > 1 \land \bigwedge_{i \in A} R_i(x+1, y, z) \land \bigwedge_{i \in B} R_i(x, y-1, z) \land \bigwedge_{i \in C} R_i(x, y, z-1) \to R_j(x, y, z)$;
**(e)** $x = 1 \land y = n \land z = n \land R_j(x, y, z) \to \bot$.

Now, transform this formula into a cube-circuit $\mathcal{C} := (\Sigma, (\mathtt{Input}_n)_{n>0}, \mathbf{Q}, \mathbf{Q}_{\mathtt{acc}}, \mathtt{g})$. The idea is still that the state of a site $(x, y, z) \in [1, n]^3$ is the set of predicates $R_i$ such that $R_i(x, y, z)$ is true, and $\mathbf{Q}$ is again the power set of the set of $\mathbf{R}$ indices: $\mathbf{Q} := \mathcal{P}([1, m])$. There are four types of transition (a-d), which mimic the clauses (a-d) above. These are, for $s \in \Sigma$ and $q, q', q'' \in \mathbf{Q}$:

**(a)** $\mathtt{g}(\sharp, \sharp, \sharp, s) = \{j \in [1, m] \mid \exists$ a clause (a) with $Q_s$, and conclusion $R_j(x, y, z)\}$;
**(b)** $\mathtt{g}(q, q', \sharp, \$) = \{j \in [1, m] \mid \exists$ a clause (b) with $A \subseteq q$, $B \subseteq q'$, and conclusion $R_j(x, y, z)\}$;
**(c)** $\mathtt{g}(\sharp, \sharp, q, \$) = \{j \in [1, m] \mid \exists$ a clause (c) with $A \subseteq q$, and conclusion $R_j(x, y, z)\}$;
**(d)** $\mathtt{g}(q, q', q'', \$) = \{j \in [1, m] \mid \exists$ a clause (d) with $A \subseteq q$, $B \subseteq q'$, $C \subseteq q''$, and conclusion $R_j(x, y, z)\}$.

Here again, the set of accepting states of $\mathcal{C}$ is determined by the contradiction clauses (e): $\mathbf{Q}_{\mathtt{acc}} := \{q \in \mathbf{Q} \mid q$ contains no $j$ such that $R_j$ occurs in a clause (e)$\}$.


$(x, y, z) \quad \mapsto \quad (c_1 = y, c_2 = z, t = z + y - x)$

$(x = c_1 + c_2 - t, y = c_1, z = c_2) \quad \leftarrow\!\shortmid \quad (c_1, c_2, t)$

`Cube` `RealTime2OCA`

**Figure 7** Bijection between the sites of $\mathcal{C}_w$ and the space-time sites of a 2-OCA on $w$.

We can easily check the equivalence, for each $w \in \Sigma^+$: $\langle w \rangle \models \Phi \iff \mathcal{C}$ accepts $w$. Therefore, the inclusion `incl-pred-ESO-HORN` $\subseteq$ `Cube` is proved. ◀

## 4.3 Cube-circuits are equivalent to real-time 2-OCA

One observes that by a one-to-one transformation, the computation $\mathcal{C}_w$ of a cube-circuit $\mathcal{C}$ on a word $w$ is nothing else than the space-time diagram of a real-time 2-OCA on the input $w$. This yields:

▶ **Lemma 21.** `Cube` = `RealTime2OCA`.

**Proof.** The bijection between the sites $(x, y, z)$ of the computation $\mathcal{C}_w$ of a cube-circuit $\mathcal{C}$ on a word $w$ and the sites $(c_1, c_2, t)$ of the space-time diagram of a real-time 2-OCA on the input $w$ is depicted in Figure 7. We check that this bijection respects the communication scheme and the input/output sites of both computation models as shown in Figure 7. By this transformation, the transition function g of the cube-circuit, which is $\langle x, y, z \rangle = \mathtt{g}(\langle x+1, y, z\rangle, \langle x, y-1, z\rangle, \langle x, y, z-1\rangle, \mathtt{Input}_n(w, x, y, z))$ becomes the transition function f of the 2-OCA: $\langle c_1, c_2, t\rangle = \mathtt{f}(\langle c_1, c_2, t-1\rangle, \langle c_1 - 1, c_2, t-1\rangle, \langle c_1, c_2 - 1, t-1\rangle)$, and vice versa. ◀

**Proof of Theorem 15.** Lemmas 20 and 21 give us the following equalities of classes: `incl-pred-ESO-HORN` = `Cube` = `RealTime2OCA`.

From the fact that the class `RealTime2OCA` is closed under complement and from Lemma 17, we deduce `Conj` $\subseteq$ `incl-pred-ESO-HORN` = `Cube` = `RealTime2OCA`. ◀

# 5 Conclusion

We have proved the inclusions $\mathtt{Conj}_1 \subseteq$ `RealTime1CA` and `Conj` $\subseteq$ `RealTime2OCA` by expressing in two logics (proved equivalent to `RealTime1CA` and `RealTime2OCA`, respectively) the inductive process of a conjunctive grammar. These results contribute to a better knowledge of relationships between automata, grammars and logic. We think that they bring us closer to prove or disprove that `Conj` is a subclass of `RealTime1CA`.

Figure 8 recapitulates the known inclusions between the language classes that we have considered here. For each of the $\subseteq$ inclusions of this figure, whether it is strict or not is an open question. Note that it was necessary to add an extra dimension to the space-time diagram to recognize any conjunctive language with a cellular automaton. Otherwise, any context-free or conjunctive language would always be decided by a RAM in time $O(n^2)$, which seems unlikely!

Besides, to grasp the expressive power, largely unknown, of the Conj (resp. $\texttt{Conj}_1$) class, it would be important to obtain exact characterizations of this class in logic and/or computational complexity. This is a fascinating question for future research!

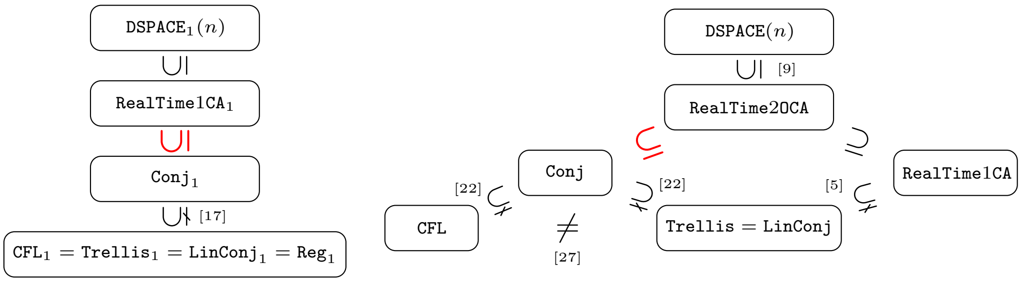


**Figure 8** Relations between language classes over a unary or general alphabet.

## References


1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

2 Nicolas Bacquey, Etienne Grandjean, and Frédéric Olive. Definability by Horn Formulas and Linear Time on Cellular Automata. In *ICALP 2017*, volume 80, pages 99:1–99:14, 2017.

3 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

4 Jik H. Chang, Oscar H. Ibarra, and Michael A. Palis. Efficient simulations of simple models of parallel computation by time-bounded atms and space-bounded tms. *Theor. Comput. Sci.*, 68(1):19–36, 1989. doi:10.1016/0304-3975(89)90116-3.

5 Christian Choffrut and Karel Culik II. On real-time cellular automata and trellis automata. *Acta Informatica*, 21(4):393–407, 1984.

6 Marianne Delorme and Jacques Mazoyer. *Cellular Automata as Language Recognizers in Cellular Automata: a Parallel Model*. Kluwer, 1999.

7 Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation, SIAM-AMS Proceedings*, pages 43–73, 1974.

8 Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Inf. Comput.*, 125(1):32–45, 1996. doi:10.1006/inco.1996.0018.

9 Leslie M. Goldschlager. A universal interconnection pattern for parallel computers. *J. ACM*, 29(4):1073–1086, 1982. doi:10.1145/322344.322353.

10 Erich Grädel. Capturing complexity classes by fragments of second-order logic. *Theoretical Computer Science*, 101(1):35–57, 1992.

11 Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Springer, 2007.

12 Etienne Grandjean and Théo Grente. Descriptive complexity for minimal time of cellular automata. In *LICS, 2019*, pages 1–13, 2019.

13 Etienne Grandjean, Théo Grente, and Véronique Terrier. Inductive definitions in logic versus programs of real-time cellular automata. *hal.archives-ouvertes.fr/hal-02474520/ submitted to Theoretical Computer Science*, 62 pages, February 2020.

14 Etienne Grandjean and Frédéric Olive. A logical approach to locality in pictures languages. *Journal of Computer and System Science*, 82(6):959–1006, 2016.

15 Oscar H. Ibarra and Tao Jiang. Relating the power of cellular arrays to their closure properties. *Theor. Comput. Sci.*, 57:225–238, 1988. doi:10.1016/0304-3975(88)90040-0.

16 Neil Immerman. *Descriptive complexity.* Springer, 1999.

17 Artur Jez. Conjunctive grammars generate non-regular unary languages. *Int. J. Found. Comput. Sci.*, 19(3):597–615, 2008. doi:10.1142/S012905410800584X.

18 K. N. King. Alternating multihead finite automata. *Theor. Comput. Sci.*, 61:149–174, 1988. doi:10.1016/0304-3975(88)90122-3.

19 Hans Kleine Büning and Theodor Lettmann. *Propositional logic - deduction and algorithms*, volume 48 of *Cambridge tracts in theoretical computer science.* Cambridge University Press, 1999.

20 S. Rao Kosaraju. Speed of recognition of context-free languages by array automata. *SIAM J. Comput.*, 4(3):331–340, 1975. doi:10.1137/0204028.

21 Leonid Libkin. *Elements of Finite Model Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

22 Alexander Okhotin. Conjunctive grammars. *J. Autom. Lang. Comb.*, 6(4):519–535, 2001. doi:10.25596/jalc-2001-519.

23 Alexander Okhotin. Conjunctive grammars and systems of language equations. *Program. Comput. Softw.*, 28(5):243–249, 2002. doi:10.1023/A:1020213411126.

24 Alexander Okhotin. Boolean grammars. *Information and Computation*, 194(1):19–48, 2004.

25 Alexander Okhotin. On the equivalence of linear conjunctive grammars and trellis automata. *Theoretical Informatics and Applications*, 38(1):69–88, 2004.

26 Alexander Okhotin. Conjunctive and boolean grammars: The true general case of the context-free grammars. *Computer Science Review*, 9:27–59, 2013.

27 Véronique Terrier. Language not recognizable in real time by one-way cellular automata. *Theoretical Computer Science*, 156(1–2):281–287, March 1996.

28 Véronique Terrier. Closure properties of cellular automata. *Theor. Comput. Sci.*, 352(1-3):97–107, 2006. doi:10.1016/j.tcs.2005.10.039.

29 Véronique Terrier. Low complexity classes of multidimensional cellular automata. *Theor. Comput. Sci.*, 369(1-3):142–156, 2006.

30 Véronique Terrier. Language recognition by cellular automata. In *Handbook of Natural Computing*, pages 123–158. Springer, 2012.

31 Hao Wang. Dominoes and the aea case of the decision problem. In *Proceedings on the Symposium on the Mathematical Theory of Automata, April 1962*, pages 23–55, 1963.


# A Complement of proof for Lemma 13

**Elimination of hypotheses $R(x, y)$.** The first idea is to group together in each computation clause the hypothesis atoms of the form $R(x, y)$ and the conclusion of the clause. Accordingly, the formula obtained $\Phi$ can be rewritten in the form

$$\Phi := \exists \mathbf{R} \forall x \forall y \left[ \bigwedge_i C_i(x,y) \wedge \bigwedge_{i \in [1,k]} (\alpha_i(x,y) \to \theta_i(x,y)) \right]$$

where the $C_i$'s are the input clauses and the contradiction clause and each computation clause is written in the form $\alpha_i(x, y) \to \theta_i(x, y)$ where $\alpha_i(x, y)$ is a conjunction of formulas of the only forms $R(x-1, y) \wedge \neg\mathtt{min}(x)$, $R(x, y-1) \wedge \neg\mathtt{min}(y)$ (*but not* $R(x, y)$), and $\theta_i(x, y)$ is a Horn clause whose *all* atoms are of the form $R(x, y)$.

We number $R_1, \ldots, R_m$ the computation predicates of $\mathbf{R}$. To each subset $J \subseteq [1,k]$ of the family of implications $(\alpha_i(x,y) \to \theta_i(x,y))_{i\in[1,k]}$ let us associate the set

$$K_J := \{h \in [1,m] \mid \bigwedge_{i\in J} \theta_i(x,y) \to R_h(x,y) \text{ is a tautology}\}.$$

Note that the notion of *tautology* used in the definition of $K_J$ is "propositional" because all the atoms involved are of the form $R_i(x,y)$, i.e., refer to the same pair of variables $(x,y)$. Also, note that the function $J \mapsto K_J$ is *monotonic*: for $J' \subseteq J$, we have $K_{J'} \subseteq K_J$ because $\bigwedge_{i\in J'} \theta_i(x,y) \to R_h(x,y)$ implies $\bigwedge_{i\in J} \theta_i(x,y) \to R_h(x,y)$.

Clearly, it is enough to prove the following claim:

▷ Claim 22. The formula $\Phi$ is equivalent to the following formula $\Phi'$, whose clauses have *no hypothesis* $R(x,y)$.

$$\Phi' := \exists \mathbf{R} \forall x \forall y \left[ \bigwedge_i C_i(x,y) \wedge \bigwedge_{J\subseteq[1,k]} \bigwedge_{h\in K_J} \left( \bigwedge_{i\in J} \alpha_i(x,y) \to R_h(x,y) \right) \right]$$

*Proof of the implication* $\Phi \Rightarrow \Phi'$: It is enough to prove the implication

$$\left[ \bigwedge_{i\in[1,k]} (\alpha_i(x,y) \to \theta_i(x,y)) \right] \to \left[ \bigwedge_{i\in J} \alpha_i(x,y) \to \bigwedge_{h\in K_J} R_h(x,y) \right]$$

for all set $J \subseteq [1,k]$. The implication to be proved can be equivalently written:

$$\left[ \bigwedge_{i\in J} \alpha_i(x,y) \wedge \bigwedge_{i\in[1,k]} (\alpha_i(x,y) \to \theta_i(x,y)) \right] \to \bigwedge_{h\in K_J} R_h(x,y).$$

The sub-formula between brackets above implies the conjunction $\bigwedge_{i\in J} \theta_i(x,y)$. As the implication $\bigwedge_{i\in J} \theta_i(x,y) \to \bigwedge_{h\in K_J} R_h(x,y)$ is a tautology (by definition of $K_J$), the implication to be proved is a tautology too.

The converse implication $\Phi' \Rightarrow \Phi$ is more difficult to prove. It uses a folklore property of propositional Horn formulas easy to be proved:

▶ **Lemma 23** (Horn property: folklore). *Let $F$ be a* strict Horn formula *of propositional calculus, that is a conjunction of clauses of the form $p_1 \wedge \ldots \wedge p_k \to p_0$ where $k \geq 0$ and the $p_i$'s are propositional variables. Let $F'$ be the conjunction of propositional variables $q$ such that the implication $F \to q$ is a tautology. $F$ has the same minimal model* [6] *as $F'$.*

*Proof of the implication* $\Phi' \Rightarrow \Phi$: Let $\langle w \rangle$ be a model of $\Phi'$ and let $(\langle w \rangle, \mathbf{R})$ be the minimal model of the Horn formula

$$\varphi' := \forall x \forall y \left[ \bigwedge_i C_i(x,y) \wedge \bigwedge_{J\subseteq[1,k]} \bigwedge_{h\in K_J} \left( \bigwedge_{i\in J} \alpha_i(x,y) \to R_h(x,y) \right) \right].$$

[6] For example, for $F := p_1 \wedge p_3 \wedge (p_1 \wedge p_3 \to p_5) \wedge (p_1 \wedge p_2 \to p_4)$, we have $F' := p_1 \wedge p_3 \wedge p_5$, which has the same minimal model $I$ as $F$; this model is given by $I(p_1) = I(p_3) = I(p_5) = 1$ and $I(p_2) = I(p_4) = 0$.

It is enough to show that $(\langle w\rangle, \mathbf{R})$ also satisfies the formula

$$\varphi := \forall x \forall y \left[ \bigwedge_i C_i(x,y) \wedge \bigwedge_{i\in[1,k]} (\alpha_i(x,y) \to \theta_i(x,y)) \right].$$

As each $\alpha_i$ is a conjunction of formulas of the form $R(x-1,y) \wedge \neg\mathtt{min}(x)$, or $R(x,y-1) \wedge \neg\mathtt{min}(y)$, we make an induction on the domain $\{(a,b) \in [1,n]^2 \mid a+b \le t\}$, for $t \in [1,2n]$. More precisely, we are going to prove, by recurrence on the integer $t \in [1,2n]$, that the minimal model $(\langle w\rangle, \mathbf{R})$ of $\varphi'$ satisfies the "relativized" formula $\varphi_t$ of the formula $\varphi$ defined by

$$\varphi_t := \forall x \forall y \left[ x+y \le t \to \left[ \bigwedge_i C_i(x,y) \wedge \bigwedge_{i\in[1,k]} (\alpha_i(x,y) \to \theta_i(x,y)) \right] \right]$$

As the hypothesis $x+y \le 2n$ holds for all $x,y$ in the domain $[1,n]$, $\varphi_{2n}$ is equivalent to $\varphi$ on the structure $(\langle w\rangle, \mathbf{R})$.

*Basis case:* For $t=1$ the set $\{(a,b) \in [1,n]^2 \mid a+b \le t\}$ is empty so that the "relativized" formula $\varphi_1$ is trivially true in the minimal model $(\langle w\rangle, \mathbf{R})$ of $\varphi'$.

*Recurrence step:* Suppose $(\langle w\rangle, \mathbf{R}) \models \varphi_{t-1}$, for an integer $t \in [2,2n]$. It is enough to show that, for each couple $(a,b) \in [1,n]^2$ such that $a+b=t$, we have $(\langle w\rangle, \mathbf{R}) \models \bigwedge_{i\in[1,k]}(\alpha_i(a,b) \to \theta_i(a,b))$. Let $J_{a,b}$ be the set of indices $i \in [1,k]$ such that the couple $(a,b)$ satisfies $\alpha_i$:

$$J_{a,b} := \{i \in [1,k] \mid (\langle w\rangle, \mathbf{R}) \models \alpha_i(a,b)\}.$$

Recall that each $\alpha_i(a,b)$ is a (possibly empty) conjunction of atoms $R(a',b')$ with $(a',b') = (a-1,b)$ or $(a',b') = (a,b-1)$, therefore such that $a'+b' = t-1$. Let $J \subseteq [1,k]$ be any set. Let us examine the two possible cases:

1) $J \subseteq J_{a,b}$: then the conjunction $\bigwedge_{i\in J} \alpha_i(a,b)$ holds in $(\langle w\rangle, \mathbf{R})$; hence, in $(\langle w\rangle, \mathbf{R})$, the conjunction $\bigwedge_{h\in K_J}(\bigwedge_{i\in J} \alpha_i(a,b) \to R_h(a,b))$ is equivalent to $\bigwedge_{h\in K_J} R_h(a,b)$;

2) $J \setminus J_{a,b} \neq \emptyset$: then the conjunction $\bigwedge_{i\in J} \alpha_i(a,b)$ is false in $(\langle w\rangle, \mathbf{R})$; hence, the conjunction $\bigwedge_{h\in K_J}(\bigwedge_{i\in J} \alpha_i(a,b) \to R_h(a,b))$ holds in $(\langle w\rangle, \mathbf{R})$.

From (1) and (2), we deduce that in $(\langle w\rangle, \mathbf{R})$ the conjunction $\bigwedge_{J\subseteq[1,k]} \bigwedge_{h\in K_J}(\bigwedge_{i\in J} \alpha_i(a,b) \to R_h(a,b))$ is equivalent to the conjunction $\bigwedge_{J\subseteq J_{a,b}} \bigwedge_{h\in K_J} R_h(a,b)$, which can be simplified as $\bigwedge_{h\in K_{J_{a,b}}} R_h(a,b)$ because $J \subseteq J_{a,b}$ implies $K_J \subseteq K_{J_{a,b}}$. Consequently, for all $h \in [1,m]$, the minimal model $(\langle w\rangle, \mathbf{R})$ of the Horn formula $\varphi'$ satisfies the atom $R_h(a,b)$ *iff* $h$ belongs to $K_{J_{a,b}}$. By definition,

$$K_{J_{a,b}} := \{h \in [1,m] \mid \bigwedge_{i\in J_{a,b}} \theta_i(x,y) \to R_h(x,y) \text{ is a tautology}\}$$

or, equivalently,

$$K_{J_{a,b}} := \{h \in [1,m] \mid \bigwedge_{i\in J_{a,b}} \theta_i(a,b) \to R_h(a,b) \text{ is a tautology}\}.$$

As a consequence of Lemma 23, the two conjunctions

$$\bigwedge_{i\in J_{a,b}} \theta_i(a,b) \text{ and } \bigwedge_{h\in K_{J_{a,b}}} R_h(a,b)$$

have the same minimal model, which is also the restriction of the minimal model $(\langle w \rangle, \mathbf{R})$ of $\varphi'$ to the set of atoms $R_h(a,b)$, for $h \in [1,m]$. Therefore, if $i \in J_{a,b}$, then $(\langle w \rangle, \mathbf{R}) \models \theta_i(a,b)$. If $i \in [1,k] \setminus J_{a,b}$, then we have $(\langle w \rangle, \mathbf{R}) \models \neg\alpha_i(a,b)$, by definition of $J_{a,b}$. Therefore, for all $i \in [1,k]$, we get $(\langle w \rangle, \mathbf{R}) \models \neg\alpha_i(a,b) \vee \theta_i(a,b)$. In other words, for all $(a,b)$ such that $a+b=t$, we have : $(\langle w \rangle, \mathbf{R}) \models \bigwedge_{i\in[1,k]}(\alpha_i(a,b) \to \theta_i(a,b))$ and then $(\langle w \rangle, \mathbf{R}) \models \varphi_t$.

This concludes the inductive proof that $(\langle w \rangle, \mathbf{R}) \models \varphi_t$, for all $t \in [1,2n]$, and then $\langle w \rangle \models \Phi$. This proves the converse implication $\Phi' \Rightarrow \Phi$. Claim 22 is demonstrated. ☐

## B Complement of proof for Lemma 14

**`Grid` $\subseteq$ `RealTime1CA`.** To prove this inclusion, we show how to simulate the computation of the grid-circuit on a real-time CA. The simulation is made by a geometric transformation that embeds the grid-circuit in the space-time diagram of a real-time CA. This transformation is divided into three steps:

1. a variable change: we apply to each site $(x,y) \in [1,n]^2$ of the grid-circuit the variable change $(x,y) \mapsto (c' = y - x + 1, t' = x + y - 1)$;
2. a folding: we fold the resulting diagram along the axis $c' = 1$: each site $(c',t')$ with $c' < 1$ is send to its symmetric counterpart $(-c'+1, t')$;
3. a grouping: each site $(c,t) = (\lceil \frac{c'}{2} \rceil, \lceil \frac{t'}{2} \rceil)$ of the new diagram records the set of sites $\{(c'-1,t'-1),(c',t'),(c'+1,t'-1)\}$ with $c'$ and $t'$ odd and greater than 1.

The resulting diagram is the expected space-time diagram of a real-time CA, proving the inclusion.

**`RealTime1CA` $\subseteq$ `Grid`.** To simulate a real-time CA $\mathcal{A} = (\mathtt{S}, \mathtt{S}_{accept}, \{-1,0,1\}, \mathtt{f})$ on the grid, we first turn $\mathcal{A}$ into an equivalent CA $\mathcal{A}' = (\mathtt{S}, \mathtt{S}_{accept}, \{-2,-1,0\}, \mathtt{f})$. This transformation can be seen as the variable change $(c,t) \mapsto (c+t-1, t)$. The diagram of $\mathcal{A}'$ is then embedded on the grid-circuit $\mathcal{C}'$ by applying to its sites $(c',t')$ the variable change $(c',t') \mapsto (t',c')$. The local and uniform communication of the embedded diagram can easily be carried out by the grid-circuit communication scheme.