A Logical Characterization of Small 2NFAs

Christos A. Kapoutsis and Lamana Mulaffer [⋆]

Carnegie Mellon University in Qatar

Abstract. Let $\mathsf{2N}$ be the class of families of problems solvable by families of *two-way nondeterministic finite automata* of polynomial size. We characterize $\mathsf{2N}$ in terms of families of formulas of *transitive-closure logic*. These formulas apply the transitive-closure operator on a quantifier-free disjunctive normal form of *first-order logic with successor and constants*, where (i) apart from two special variables, all others are equated to constants in every clause, and (ii) no clause simultaneously relates these two special variables and refers to fixed input cells. We prove that automata with polynomially many states are as powerful as formulas with polynomially many clauses and polynomially large constants. This can be seen as a refinement of Immerman's theorem that nondeterministic logarithmic space matches positive transitive-closure logic ($\mathsf{NL} = \mathsf{FO}+\mathsf{pos\,TC}$).

1 Introduction

A formal machine M and a logical formula φ are equivalent if they determine the same language: a string w is accepted by M iff it satisfies φ. Such comparisons between machines and formulas are the topic of Descriptive Complexity Theory [4]. Its inaugural result was Fagin's Theorem, which says that polynomial-time nondeterministic Turing machines (NTMs) are equivalent to formulas of *existential second-order logic* ($\mathsf{NP} = \exists\mathsf{SO}$) [2]. An analogous result for space complexity is Immerman's theorem that logarithmic-space NTMs are equivalent to formulas of *positive transitive-closure logic* ($\mathsf{NL} = \mathsf{FO}+\mathsf{pos\,TC}$) [3]. Today we know many such 'logical characterizations' of various computational complexity classes [4].

When it comes to finite automata (on finite strings), an old result of this kind is Büchi's Theorem, that *one-way nondeterministic finite automata* (1NFAs) are equivalent to formulas of *monadic second-order logic with successor* (MSO[S]) [1] —and thus so are, too, all automata recognizing the regular languages, including the deterministic and/or two-way variants (1DFAs, 2DFAs, 2NFAs). But this is a 'computability result', in the sense that the equivalence involves no restriction on the automata resources —as opposed to Fagin's and Immerman's 'complexity results', where the NTMs are restricted to use only polynomial time or logarithmic space, respectively. What if we focus on automata where the main resource, the number of states, is restricted to be polynomial (in a given parameter)?

We first asked this in [6], in the context of building a size-complexity theory of two-way finite automata, or 'Minicomplexity Theory' [5]. Specifically, we asked:

[⋆] Supported by the CMUQ Student-Initiated Undergraduate Research Program 2013.

What is an analog of Fagin's Theorem when we replace NTMs and time with 2NFAs and size? Unfortunately, however, we failed to answer in full generality. Instead, we proved such analogs only for the one-way, rotating, and sweeping restrictions of 2NFAs (where the input head can, respectively, only move forward; or only move forward and jump to the start; or turn only on the end-markers).

The present paper contains the full answer to that question of [6]. In what can be seen as a refinement of Immerman's theorem from above, we prove that polynomial-size 2NFAs are equivalent to a certain class of formulas of $\mathsf{FO}+\mathsf{pos}\,\mathsf{TC}$. Specifically, we focus on formulas consisting of a single, positive application of the transitive-closure operator on a quantifier-free disjunctive normal form of first-order logic with successor and constants, where (i) each of the conjunctive clauses equates every variable, except for two special ones, to some constant, and (ii) none of these clauses can *both* relate the two special variables *and* refer to a fixed input cell. We call such formulas *weak one-dimensional graph-accessibility disjunctive-normal-forms* (weak GA/DNF$_1$s) and prove that they are equivalent to polynomial-size 2NFAs, if their clauses are only polynomially many and their constants are only polynomially large. We thus complete our first step, started in [6], into what one could call 'Descriptive Minicomplexity Theory'.

2 Preparation

Let $\mathbb{Z}$ be all integers and $\mathbb{Z}^{\pm} := \mathbb{Z}-\{0\}$. If $n \geq 0$, then we let $[n] := \{0,\dots,n-1\}$, $\mathbb{Z}^{+}_{n} := \{1,\dots,n\}$, and $\mathbb{Z}^{-}_{n} := \{-n,\dots,-1\}$. If $w \in \Sigma^*$ is a finite string over some alphabet Σ, then $|w|$ and w_x are its length and x-th symbol (if $1 \leq x \leq |w|$).

2.1 Finite Automata

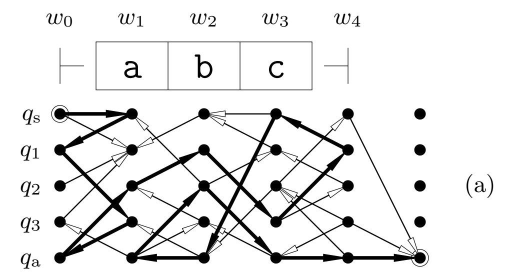
A *two-way nondeterministic finite automaton* is a tuple $N = (S, \Sigma, \delta, q_{\mathrm{s}}, q_{\mathrm{a}})$ of a set of *states* S, an *alphabet* Σ, a *start state* $q_{\mathrm{s}} \in S$, an *accept state* $q_{\mathrm{a}} \in S$, and a set of *transitions* $\delta \subseteq S \times (\Sigma \cup \{\vdash,\dashv\}) \times S \times \{\textsc{l},\textsc{r}\}$, where $\vdash,\dashv \notin \Sigma$ are the two *end-markers* and L,R are the two directions of motion for the input head.

A word $w \in \Sigma^*$ is presented to N between the end-markers, as $\vdash w \dashv$. The computation starts at q_{s} on $\vdash$. At each step, the next state and head motion may be any of those derived from δ and the current state and symbol. End-markers are never violated, except for $\dashv$ if the next state is q_{a}. So, each branch of N's computation can *hang* inside the input; or *loop*; or fall off $\dashv$ into q_{a}, in which case we call it *accepting*. If at least one branch is accepting, we say N *accepts* w.

Let $n = |w|$. A *configuration* of N on w is a pair $(p, x) \in S \times [n{+}3]$; it means N is at state p reading w_x, if $x \leq n{+}1$ (we let $w_0{:=}\vdash$ and $w_{n+1}{:=}\dashv$); or has fallen off $\dashv$ into p, if $x = n{+}2$. The *configuration graph* $G_{N,w}$ of N on w (Fig. 1a) is the directed graph where vertices are configurations of N on w and an edge $(p, x) \to (q, y)$ exists iff N can switch from (p, x) to (q, y) in a single step, i.e., iff

$$y = x{+}1 \quad \& \quad (p, w_x, q, \textsc{r}) \in \delta \qquad \text{or} \qquad y = x{-}1 \quad \& \quad (p, w_x, q, \textsc{l}) \in \delta\,. \tag{1}$$

Clearly, *N accepts w iff $G_{N,w}$ has a path* $(q_{\mathrm{s}}, 0) \rightsquigarrow (q_{\mathrm{a}}, n{+}2)$.



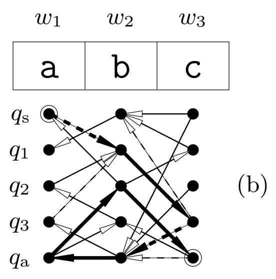


Fig. 1. (a) The configuration graph of a 5-state 2NFA N on a word $w = \texttt{abc}$. Bold arrows show an accepting branch. (b) The *inner* configuration graph of N on w. Dashed arrows are caused by computations on $\vdash w_1$ or $w_3 \dashv$. E.g., $(q_3, 1) \to (q_1, 2)$ is caused by the path $(q_3, 1) \to (q_a, 0) \to (q_2, 1) \to (q_1, 2)$ in (a).

When $n \geq 2$, a denser representation is the *inner configuration graph* $G'_{N,w}$ (Fig. 1b), where now the vertices are only the *inner configurations* $S \times \mathbb{Z}^+_n$ and an edge, or *inner step*, $(p, x) \to (q, y)$ exists iff any of the following holds:

- N can switch from (p, x) to (q, y) in a single step, as in (1);
- $x = 1$, $y = 2$, and the switch can happen by a U-turn computation on $\vdash w_1$;
- $x = n$, $y = n{-}1$, and the switch can happen by a U-turn on $w_n \dashv$.

We will need to say that N *accepts* w *iff* $G'_{N,w}$ *has a path* $(q_s, 1) \rightsquigarrow (q_a, n)$. But, in general, this is false; it becomes true, if N is in the form of Def. 1 (Fact 1ii). Conveniently, with two more states, every 2NFA can be put in this form (Fact 1i).

Definition 1. *A* 2NFA $N = (\,.\,, \Sigma, \delta, q_s, q_a)$ *is in* inner normal form (INF) *if*
 i. δ *contains* $(q_s, \vdash, q_s, \mathrm{R})$*, but no other tuple* $(q_s, \vdash, \,.\,, \,.\,)$*; and*
 ii. δ *contains every* $(q_a, a, q_a, \mathrm{R})$ *for* $a \in \Sigma \cup \{\dashv\}$*, but no other tuple* $(\,.\,, \,.\,, q_a, \mathrm{R})$.

Fact 1.
 i. *Every* s*-state* 2NFA *is equivalent to a* $(s{+}2)$*-state* 2NFA *in* INF.
 ii. *If* N *is in* INF*, then* N *accepts* w *iff* $G'_{N,w}$ *has a path* $(q_s, 1) \rightsquigarrow (q_a, n)$.

2.2 Logical Formulas

In *quantifier-free first-order logic with successor and constants* over alphabet Σ (Q·FO$^{+}_{\Sigma}[\mathsf{S},\mathbb{Z}^{\pm}]$), formulas are built out of first-order *variables* $x_0, x_1, \ldots$, *constants* $\pm 1, \pm 2, \ldots \in \mathbb{Z}^{\pm}$, one *cell* predicate $\alpha(\,.\,)$ for each $\alpha \subseteq \Sigma$, the *equality* predicate $\,.\, = \,.\,$, the *successor* predicate $\mathsf{S}(\,.\,,.\,)$, and the connectives $\neg, \wedge, \vee$. A formula φ is either an *atom*, of the form $\alpha(t)$, $t{=}t'$, or $\mathsf{S}(t, t')$, where each of the *terms* t, t' is either a variable or a constant; or *compound*, of the form $\neg\phi$, $\phi \wedge \psi$, or $\phi \vee \psi$, where ϕ, ψ are simpler formulas. An atom is either *local*, of the form $\alpha(t)$; or *relational*, of the form $t{=}t'$ or $\mathsf{S}(t, t')$. An atom or negation of atom is a *literal*. A conjunction (resp., disjunction) of literals is an $\wedge$-*clause* (resp., an $\vee$-*clause*); a disjunction (resp., conjunction) of $\leq m$ such clauses is an m-clause *disjunctive normal form*, or m-DNF (resp., an m-clause *conjunctive normal form*, or m-CNF). A formula is *non-trivial* if it is not identically true or identically false.

$$\begin{array}{c|c|}
\bot & \mathtt{a} \\ \hline
x_1 & \mathtt{1} \\
x_2 & \mathtt{0}
\end{array}\ \text{(a)}
\qquad
\begin{array}{c|c|c|c|c|c|}
 & 1 & 2 & 3 & 4 & 5 \\
\bot & \mathtt{a} & \mathtt{a} & \mathtt{b} & \mathtt{a} & \mathtt{b} \\ \hline
x_1 & \mathtt{1} & \mathtt{0} & \mathtt{0} & \mathtt{0} & \mathtt{0} \\
x_2 & \mathtt{0} & \mathtt{0} & \mathtt{1} & \mathtt{0} & \mathtt{0}
\end{array}\ \text{(b)}
\qquad
\begin{array}{c|c|c|c|c|c|}
 & 1 & 2 & 3 & 4 & 5 \\
\bot & \mathtt{a} & \mathtt{a} & \mathtt{b} & \mathtt{a} & \mathtt{b} \\ \hline
x_1 & \mathtt{1} & \mathtt{0} & \mathtt{0} & \mathtt{0} & \mathtt{0} \\
x_2 & \mathtt{0} & \mathtt{0} & \mathtt{1} & \mathtt{0} & \mathtt{0} \\
x_5 & \mathtt{0} & \mathtt{1} & \mathtt{0} & \mathtt{0} & \mathtt{0}
\end{array}\ \text{(c)}$$

Fig. 2. (a) A column from $\Sigma|V$, for $\Sigma = \{\mathtt{a},\mathtt{b}\}$, $V = \{x_1, x_2\}$. (b) A well-formed $\hat{w}$ over $\Sigma|V$; here, $\hat{w}(\bot) = \mathtt{aabab}$, $\hat{w}(x_2) = 3$, $\hat{w}(+1) = 1$, $\hat{w}(-1) = 5$. (c) The word $\hat{w}[x_5/2]$.

The *length* $|\varphi|$ of a formula φ is the number of occurences of symbols in it, ignoring punctuation and counting each variable, constant, and cell predicate as a single symbol. More carefully, we define $|\varphi|$ by structural induction on φ:

- for all α, t, t': $|\alpha(t)| = 2$ and $|t = t'| = |\mathsf{S}(t, t')| = 3$;
- for all ϕ, ψ: $\quad |\neg\phi| = 1 + |\phi|$ and $|\phi \wedge \psi| = |\phi \vee \psi| = |\phi| + 1 + |\psi|$.

The *margin* of φ is the maximum absolute value of a constant in it; or 0, if φ has no constants. We write $\varphi(x_2, x_5, \dots)$ to indicate that all variables appearing in φ are among $x_2, x_5, \dots$ (note that all variables are free, as there are no quantifiers).

Semantics. For a set of variables V, let $\Sigma|V$ be the alphabet of all functions $u : \{\bot\} \cup V \to \Sigma \cup \{\mathtt{0},\mathtt{1}\}$ which map $\bot$ into Σ and variables into $\{\mathtt{0},\mathtt{1}\}$ (namely, $u(\bot) \in \Sigma$ and $u(x_i) \in \{\mathtt{0},\mathtt{1}\}$ for all $x_i \in V$). Intuitively, every such u is a column of $1{+}|V|$ cells, labelled by the elements of $\{\bot\} \cup V$ and filled by the respective values of u (Fig. 2a). Likewise, each word $\hat{w} = \hat{w}_1 \cdots \hat{w}_n \in (\Sigma|V)^*$ is a table of n columns and $1{+}|V|$ rows: one row is labelled by $\bot$ and hosts an n-long word over Σ; the rest are labelled by variables and host n-long bitstrings (Fig. 2b).

We say $\hat{w}$ is *well-formed* if $n \geq 2$ and the row of each variable hosts exactly one $\mathtt{1}$ (Fig. 2b). Then, $\hat{w}(\bot)$ is the word $\hat{w}_1(\bot) \cdots \hat{w}_n(\bot) \in \Sigma^*$ hosted in the $\bot$-row; $\hat{w}(x_i)$ is the index x of the one column $\hat{w}_x$ which has a $\mathtt{1}$ in the x_i-row; and, for $c \in \mathbb{Z}^{\pm}$, $\hat{w}(c)$ is the index c of the c-th leftmost column, if $c > 0$, or the index $n{-}|c|{+}1$ of the $|c|$-th rightmost column, if $c < 0$. Moreover, for any $x_i \notin V$ and index $x \in \mathbb{Z}^+_n$, $\hat{w}[x_i/x]$ is the well-formed word over $\Sigma|(V \cup \{x_i\})$ derived from $\hat{w}$ by adding a row labelled x_i with its x-th bit $\mathtt{1}$ and all others $\mathtt{0}$ (Fig. 2c).

Now, given a n-long well-formed $\hat{w}$ over $\Sigma|V$ and a formula φ whose variables are all in V, we say $\hat{w}$ *satisfies* φ and write $\hat{w} \models \varphi$, if what φ 'says' about $\hat{w}(\bot)$ is true when each variable x_i is interpreted as in the x_i-row, namely iff:

$$\begin{array}{llll}
\text{for } \varphi \equiv \ \alpha(t) : & \hat{w}(\bot)_{\hat{w}(t)} \in \alpha & \text{for } \varphi \equiv \ \neg\phi \ : & \hat{w} \not\models \phi \\
\text{for } \varphi \equiv \ t = t' : & \hat{w}(t) = \hat{w}(t') & \text{for } \varphi \equiv \phi \wedge \psi: & \hat{w} \models \phi \text{ and } \hat{w} \models \psi \\
\text{for } \varphi \equiv \mathsf{S}(t, t'): & \hat{w}(t) + 1 = \hat{w}(t') & \text{for } \varphi \equiv \phi \vee \psi: & \hat{w} \models \phi \ \text{ or } \ \hat{w} \models \psi\,.
\end{array}$$

Transitive Closure. Let $\varphi(\overline{x}, \overline{y})$ be a $\text{Q}{\cdot}\text{FO}^+_\Sigma[\mathsf{S},\mathbb{Z}^{\pm}]$ formula over $2k{+}2$ variables $\overline{x} = x_0, \dots, x_k$ and $\overline{y} = y_0, \dots, y_k$. Given an n-long $w \in \Sigma^*$, this defines a binary relation R_φ on $k{+}1$-tuples of indices in $\mathbb{Z}^+_n$. As usual, the *transitive closure* of R_φ is the binary relation R^*_φ which contains a pair $(\overline{u}, \overline{v})$ iff there is a sequence of tuples $\overline{r}_0, \overline{r}_1, \cdots, \overline{r}_\ell$ such that $\overline{u} = \overline{r}_0$; every $(\overline{r}_i, \overline{r}_{i+1})$ is in R_φ; and $\overline{r}_\ell = \overline{v}$.

We augment our logic with the *transitive closure* operator 'TC', which checks if two tuples of indices are in the relation R^*_φ defined by some $\varphi(\overline{x},\overline{y})$: given φ and two tuples of terms $\overline{t},\overline{t}'$, the formula $\mathsf{TC}_\varphi(\overline{t},\overline{t}')$ (or, more legibly, $\mathsf{TC}[\varphi(\overline{x},\overline{y})](\overline{t},\overline{t}')$) has length $1{+}|\varphi|{+}2k{+}2$ and the following semantics, for all well-formed $\hat{w}$:

$$\hat{w} \models \mathsf{TC}_\varphi(\overline{t},\overline{t}') \qquad \text{iff} \qquad \big((\hat{w}(t_0),\dots,\hat{w}(t_k)),\ (\hat{w}(t'_0),\dots,\hat{w}(t'_k)) \big) \in R^*_\varphi\,.$$

Intuitively, let $G_{\varphi,\hat{w}}$ be the directed graph with vertex set $(\mathbb{Z}^+_n)^{k+1}$ and all edges $(\overline{u},\overline{v})$ such that $\hat{w}[\overline{x}/\overline{u},\overline{y}/\overline{v}] \models \varphi(\overline{x},\overline{y})$; then $\hat{w} \models \mathsf{TC}_\varphi(\overline{t},\overline{t}')$ iff $G_{\varphi,\hat{w}}$ has a path $(\hat{w}(t_0),\dots,\hat{w}(t_k)) \rightsquigarrow (\hat{w}(t'_0),\dots,\hat{w}(t'_k))$. We call this new logic Q·FO$^+_\Sigma[\mathsf{S},\mathbb{Z}^\pm]$+TC.

2.3 Finite Automata versus Logical Formulas

A (*promise*) *problem* over alphabet Σ is any pair $\mathfrak{L} = (L,\tilde{L})$ of disjoint subsets of Σ^*. An automaton N *solves* $\mathfrak{L}$ if it accepts all $w \in L$ but no $w \in \tilde{L}$. A formula φ *solves* $\mathfrak{L}$ if it is satisfied by all $w \in L$ but no $w \in \tilde{L}$.

A family of automata $\mathcal{N} = (N_h)_{h\geq 1}$ (resp., of formulas $\mathcal{F} = (\varphi_h)_{h\geq 1}$) *solves* a family of problems $(\mathfrak{L}_h)_{h\geq 1}$ if every N_h (resp., φ_h) solves the respective $\mathfrak{L}_h$. The automata of $\mathcal{N}$ (resp., the formulas of $\mathcal{F}$) are *small* if every N_h has $\leq p(h)$ states (resp., every φ_h has length $\leq p(h)$), for some polynomial p. Therefore, the set

$$\mathsf{2N} := \left\{ (\mathfrak{L}_h)_{h\geq 1} \;\middle|\; \begin{array}{l} \text{there exist 2NFAs } (N_h)_{h\geq 1} \text{ and a polynomial } p \\ \text{such that every } N_h \text{ solves } \mathfrak{L}_h \text{ with } \leq p(h) \text{ states} \end{array} \right\}$$

is the class of problem families which are solvable by families of small 2NFAs.

A formula $\varphi(\overline{x})$ of Q·FO$^+_\Sigma[\mathsf{S},\mathbb{Z}^\pm]$+TC is *equivalent* to a 2NFA N over $\Sigma|\overline{x}$ if for all well-formed $\hat{w} \in (\Sigma|\overline{x})^*$: $\hat{w}$ satisfies φ iff N accepts $\hat{w}$ (note that $|\hat{w}| \geq 2$).

3 Graph-Accessibility Sentences and Our Theorem

A formula of Q·FO$^+_\Sigma[\mathsf{S},\mathbb{Z}^\pm]$ is *local*, if all its atoms are local (i.e., it talks only about the contents of certain cells); *quasi-local*, if every relational atom in it uses at least one constant (i.e., it talks only about certain cells' contents and distance from the end-markers); and *relational*, if all its atoms are relational (i.e., it talks only about the order of certain cells). Orthogonally, the formula is *floating*, if all its terms are variables; *quasi-floating*, if every atom uses at least one variable; and *anchored*, if all its terms are constants. Finally, inside an $\wedge$-clause, a variable x is *anchored* if it appears in at least one literal of the form $x = c$ or $c = x$ (without negation), for some constant c; otherwise, it is *floating*.

Given a Q·FO$^+_\Sigma[\mathsf{S},\mathbb{Z}^\pm]$ formula $\varphi(\overline{x},\overline{y})$ with $\overline{x} = x_0,\dots,x_k$ and $\overline{y} = y_0,\dots,y_k$, a *graph-accessibility sentence* (GAS) with *core* φ and *arity* $k{+}1$ is any formula

$$\mathsf{TC}\big[\,\varphi(\overline{x},\overline{y})\,\big](\overline{s},\overline{t}) \tag{2}$$

where $\overline{s} = s_0,\dots,s_k$ and $\overline{t} = t_0,\dots,t_k$ are constants. If φ is a DNF, namely

$$\varphi(\overline{x},\overline{y}) \equiv \bigvee_{i=1}^{m} \varphi_i(\overline{x},\overline{y})$$

where each φ_i is an $\wedge$-clause and the *degree* m is ≥ 1, then we say (2) is a GA/DNF. If $x_1, \dots, x_k$ and $y_1, \dots, y_k$ are all anchored in every φ_i (so that only x_0, y_0 may be floating), then we say (2) is *one-dimensional* (GA/DNF$_1$). Finally, we say (2) is *weak* if no φ_i contains both anchored local atoms and floating relational ones.

Our theorem states that 2NFAs of polynomial size are as powerful as weak GA/DNF$_1$s of polynomial degree and margin; and that this holds already when the margin is 1 and we also require polynomial length and logarithmic arity.

Theorem 1. *The following are equivalent, for every family of problems $\mathcal{L}$:*
1. *$\mathcal{L}$ has small* 2NFAs.
2. *$\mathcal{L}$ has small weak* GA/DNF$_1$s *of small degree, margin* 1*, and logarithmic arity.*
3. *$\mathcal{L}$ has weak* GA/DNF$_1$s *of small degree and small margin.*

Proof. [(1)⇒(2)] By Lemma 1. [(2)⇒(3)] Trivial. [(3)⇒(1)] By Lemma 5. □

4 From Automata to Formulas

The simpler conversion, from automata to formulas, is treated in the next lemma.

Lemma 1. *Every s-state* 2NFA *is equivalent to a weak* GA/DNF$_1$ *of degree $O(s^2)$, margin* 1*, arity $O(\log s)$ and length $O(s^2 \log s)$.*

Proof. Pick any s-state 2NFA N. We first switch to an equivalent 2NFA $\tilde{N}$ which is in INF (Def. 1) and has $\tilde{s} = 2^r$ states, for some r; easily, $\tilde{s} \leq 2s+2$. Without loss of generality, assume $\tilde{N} = ([\tilde{s}], \Sigma, \tilde{\delta}, 0, \tilde{s}-1)$.

We need a weak GA/DNF$_1$ $\mathsf{TC}[\varphi(\overline{x}, \overline{y})](\overline{s}, \overline{t})$ such that, for all w of length $n \geq 2$:

$$\tilde{N} \text{ accepts } w \quad \iff \quad w \models \mathsf{TC}[\,\varphi(\overline{x}, \overline{y})\,](\overline{s}, \overline{t})\,. \tag{3}$$

By definition, the right-hand side holds iff the graph $G_{\varphi,w}$ induced by φ (cf. p. 5) has a path $(s_0, s_1, \dots, s_k) \rightsquigarrow (t_0, t_1, \dots, t_k)$, where $k+1$ the arity of φ. By Fact 1ii, the left-hand side holds iff the inner configuration graph $G'_{\tilde{N},w}$ induced by $\tilde{N}$ has a path $(0, 1) \rightsquigarrow (\tilde{s}-1, n)$. So, we simply need to pick φ so that $G_{\varphi,w}$ is actually $G'_{\tilde{N},w}$, and then pick $\overline{s}, \overline{t}$ so that they are actually $(0, 1)$ and $(\tilde{s}-1, n)$.

First, we must represent each vertex of $G'_{\tilde{N},w}$, namely each inner configuration $(p, x) \in [\tilde{s}] \times \mathbb{Z}_n^+$, as a vertex of $G_{\varphi,w}$, namely a tuple $\overline{u} = (u_0, u_1, \dots, u_k)$ of indices from $\mathbb{Z}_n^+$. Of course, x can be represented by any component of $\overline{u}$, say u_0. As for p, we represent it in 'binary' using the other components $u_1, \dots, u_k$: we pick $k := r = \lg \tilde{s}$ (to ensure we have enough 'bits') and use indices 1 and n (which are distinct, as $n \geq 2$) as `0` and `1`, respectively. E.g., if $\tilde{s} = 16$ (so, $k = 4$) and $n = 50$, then the configuration $(p, x) = (2, 22)$ maps to $\overline{u} = (22, 1, 1, 50, 1)$. Note that $(0, 1)$ and $(\tilde{s}-1, n)$ map to $(1, 1, \dots, 1)$ and $(n, n, \dots, n)$, i.e., to the interpretations of the tuples of constants $(+1, +1, \dots, +1)$ and $(-1, -1, \dots, -1)$.

Given this representation, we now need a $\varphi(\overline{x}, \overline{y})$ which states that the edge $(\overline{x}, \overline{y})$ exists in $G'_{\tilde{N},w}$, namely that $\tilde{N}$ can switch in a single inner step from the inner configuration $(x_0, x_1, \dots, x_k)$ to the inner configuration $(y_0, y_1, \dots, y_k)$.

As a start, for every state $p \in [\tilde{s}]$ we need a formula $\xi_p(\overline{u})$ which says that the state of the inner configuration $(u_0, u_1, \ldots, u_k)$ is p. E.g., if $n = 50$ and $p = 2$ as above, then $w \models \xi_p(\overline{u})$ should hold iff $\overline{u}$ is of the form $(\,.\,, 1, 1, 50, 1)$, and thus $\xi_p(\overline{u})$ should be $u_1 = +1 \wedge u_2 = +1 \wedge u_3 = -1 \wedge u_4 = +1$. In general, we let

$$\xi_p(\overline{u}) := \bigwedge_{i=1}^{k} (u_i = p_i)\,, \tag{4}$$

where each p_i is either $+1$ or -1 depending on whether the i-th most significant bit in the k-bit binary representation of p is `0` or `1`.

Additionally, for every two states $p, q \in [\tilde{s}]$ and each direction of head motion, we need the set of symbols which allow the corresponding transition:

$$\alpha^{\mathrm{L}}_{p,q} := \{a \in \Sigma \mid (p, a, q, \mathrm{L}) \in \tilde{\delta}\}\,, \qquad \alpha^{\mathrm{R}}_{p,q} := \{a \in \Sigma \mid (p, a, q, \mathrm{R}) \in \tilde{\delta}\}\,. \tag{5}$$

Similarly, for every two states $p, q \in [\tilde{s}]$ and each end-marker, we need the set of symbols which, together with the end-marker, allow the corresponding U-turn:

$$\begin{aligned} \alpha^{\vdash}_{p,q} &:= \{a \in \Sigma \mid \text{computing on } {\vdash}a \text{ from } p \text{ on } a,\ \tilde{N} \text{ can exit right into } q\}\,, \\ \alpha^{\dashv}_{p,q} &:= \{a \in \Sigma \mid \text{computing on } a{\dashv} \text{ from } p \text{ on } a,\ \tilde{N} \text{ can exit left into } q\}\,. \end{aligned} \tag{6}$$

Using the $\wedge$-clauses of (4) and the cell predicates for the sets of (5) and (6), we now build a formula $\varphi(\overline{x}, \overline{y})$ which says that, in a single inner step, $\tilde{N}$ can switch from cell x_0 and 'state' $(x_1, \ldots, x_k)$ to cell y_0 and 'state' $(y_1, \ldots, y_k)$:

$$\begin{aligned} \varphi(\overline{x}, \overline{y}) := \bigvee_{p,q \in [\tilde{s}]} \Big\{ \; & \big[\, \xi_p(\overline{x}) \wedge \xi_q(\overline{y}) \wedge \mathsf{S}(x_0, y_0) \wedge \alpha^{\mathrm{R}}_{p,q}(x_0) \,\big] & (7) \\ \vee\; & \big[\, \xi_p(\overline{x}) \wedge \xi_q(\overline{y}) \wedge \mathsf{S}(y_0, x_0) \wedge \alpha^{\mathrm{L}}_{p,q}(x_0) \,\big] & (8) \\ \vee\; & \big[\, \xi_p(\overline{x}) \wedge \xi_q(\overline{y}) \wedge x_0 = +1 \wedge \mathsf{S}(x_0, y_0) \wedge \alpha^{\vdash}_{p,q}(x_0) \,\big] & (9) \\ \vee\; & \big[\, \xi_p(\overline{x}) \wedge \xi_q(\overline{y}) \wedge x_0 = -1 \wedge \mathsf{S}(y_0, x_0) \wedge \alpha^{\dashv}_{p,q}(x_0) \,\big] \;\Big\} & (10) \end{aligned}$$

Intuitively, φ says that there exist states p, q such that the state of inner configuration $\overline{x}$ is p, the state of inner configuration $\overline{y}$ is q, and: $\overline{y}$ is exactly to the right of $\overline{x}$, and the symbol read in $\overline{x}$ allows a right-moving transition $p \to q$ (line (7)); or $\overline{y}$ is exactly to the left of $\overline{x}$, and the symbol read in $\overline{x}$ allows a left-moving transition $p \to q$ (line (8)); or $\overline{x}, \overline{y}$ are on cells $1, 2$ and the symbol read in $\overline{x}$ together with $\vdash$ allows a left U-turn from p to q (line (9)); or $\overline{y}, \overline{x}$ are on cells $n{-}1, n$ and the symbol read in $\overline{x}$ together with $\dashv$ allows a right U-turn from p to q (line (10)).

Overall, our GAS is that of (3) with φ as in (7)–(10) and $\overline{s} = (+1, \ldots, +1)$ and $\overline{t} = (-1, \ldots, -1)$. As promised, the *margin* is 1 (all constants are ± 1) and the *arity* is $k{+}1 = O(\log s)$. Also, each bracket in (7)–(10) is an $\wedge$-clause of length $O(\log s)$, as the conjunction of two $\wedge$-clauses of length $O(k) = O(\log s)$ and two or three atoms of length $O(1)$; hence, φ is a disjunction of $4\tilde{s}^2 = O(s^2)$ $\wedge$-clauses, of total length $O(s^2 \log s)$; and thus our GAS in (3) is a GA/DNF of *degree* $O(s^2)$ and *length* $O(s^2 \log s)$, too. Finally, each bracket in (7)–(10) anchors each one of $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$ (inside ξ_p and ξ_q) and contains no anchored local atoms, making our GAS in (3) both *one-dimensional* and *weak*, as promised. $\square$

5 From Formulas to Automata

We now show how to convert a weak GA/DNF$_1$ to a 2NFA. Facts 2–5 analyze the structure of the given sentence and its sub-formulas; their proofs are straightforward and mostly syntactic. Lemmas 2–4 build two-way automata which simulate those sub-formulas. The final 2NFA for the given sentence is built in Lemma 5.

Fact 2. *Let $\varphi(\overline{x},\overline{y})=\bigvee_{i=1}^{m}\varphi_i(\overline{x},\overline{y})$ be the core of a* GA/DNF$_1$ *of arity $k+1$. Then every $\wedge$-clause $\varphi_i(\overline{x},\overline{y})$ is equivalent to an $\wedge$-clause of the form*

$$(x_1=c_1)\wedge\cdots\wedge(x_k=c_k)\ \wedge\ (y_1=d_1)\wedge\cdots\wedge(y_k=d_k)\ \ \wedge\ \ \hat{\varphi}(x_0,y_0)\,,$$

for some constants $c_1,\dots,c_k,d_1,\dots,d_k$ and some $\wedge$-clause $\hat{\varphi}(x_0,y_0)$.

Proof. Pick any φ_i. By one-dimensionality, x_1 is anchored in φ_i, so at least one literal is of the form $x_1=c_1$ or $c_1=x_1$, for some constant c_1. Consider the following modifications: (1) if the literal is $c_1=x_1$, change it to $x_1=c_1$; (2) bring the literal upfront; (3) replace any other occurence of x_1 with c_1. Easily, this brings φ_i into the equivalent form $(x_1=c_1)\wedge\vartheta_1(x_0,x_2,x_3,\dots,x_k,\overline{y})$. Similarly, x_2 is also anchored in φ_i, so by repeating modifications (1)–(3) for it, we bring φ_i to the equivalent form $(x_1=c_1)\wedge(x_2=c_2)\wedge\vartheta_2(x_0,x_3,x_4,\dots,x_k,\overline{y})$. Continuing like this for all anchored variables, we eventually get the desired equivalent form $(x_1=c_1)\wedge\cdots\wedge(x_k=c_k)\wedge(y_1=d_1)\wedge\cdots\wedge(y_k=d_k)\wedge\vartheta_{2k}(x_0,y_0)$. $\square$

Fact 3. *Every non-trivial $\wedge$-clause $\varphi(x,y)$ is equivalent to a formula of the form $\phi\wedge\chi(x)\wedge\psi(y)\wedge\omega(x,y)$, where each of ϕ,χ,ψ,ω is an $\wedge$-clause; ϕ is anchored local; χ,ψ are quasi-floating quasi-local; and ω is floating relational.*

Lemma 2. *Suppose φ is an anchored local $\wedge$-clause of margin τ. Then there exists a $O(\tau)$-state* 2DFA *which, whenever run on a string w from the cell of $\vdash$, returns on that same cell and accepts iff $w\models\varphi$.*

Proof. Formula φ is a conjunction of literals of the form $\alpha(c)$ and $\neg\alpha(c)$, where $\alpha\subseteq\Sigma$ and $c\in\mathbb{Z}_\tau^+\cup\mathbb{Z}_\tau^-$. We may assume that every such c appears in exactly one literal of the form $\alpha(c)$: Indeed, if it appears in none, then we add the true literal $\Sigma(c)$; if it appears in exactly one, but of the form $\neg\gamma(c)$, then we replace this with the equivalent $\overline{\gamma}(c)$; if it appears in more than one, then we replace the conjunction $\beta_1(c)\wedge\cdots\wedge\beta_r(c)\wedge\neg\gamma_1(c)\wedge\cdots\wedge\neg\gamma_s(c)$ of these literals with the equivalent single literal $\alpha(c)$ where $\alpha:=\beta_1\cap\cdots\cap\beta_r\cap\overline{\gamma}_1\cap\cdots\cap\overline{\gamma}_s$.

So, φ is essentially a list of 2τ conditions, one for each of the τ leftmost and the τ rightmost cells of w, and $w\models\varphi$ iff all are true. To test this, a 2DFA M starting from $\vdash$ scans the leftmost cells, counting up to τ and confirming all respective conditions; then sweeps to $\dashv$; then scans the rightmost cells backwards, again counting up to τ and confirming all respective conditions; then sweeps to $\vdash$ and accepts —if any condition fails or any cell does not exist (because w is too short), then M rejects. Easily, this can be implemented with $O(\tau)$ states. $\square$

Fact 4. *Every quasi-local formula is equivalent to a formula in which every atom is of the form $\alpha(\,.\,)$ or $x=c$, where $\alpha\subseteq\Sigma$, x is a variable, and c is a constant.*

Lemma 3. *Suppose $\varphi(x)$ is a quasi-floating quasi-local $\wedge$-clause of margin τ. Then there exists a $O(\tau)$-state* 2DFA *which, whenever run on a string w from a cell $1 \leq x^* \leq |w|$, returns on that same cell and accepts iff $w[x/x^*] \models \varphi(x)$.*

Proof. By Fact 4, by the margin τ, and since φ is quasi-floating with x as the only variable, we may assume that every atom is of the form $\alpha(x)$ or $x = c$, where $\alpha \subseteq \Sigma$ and $c \in \mathbb{Z}_\tau^+ \cup \mathbb{Z}_\tau^-$.

So, each literal has the form $\alpha(x)$, $\neg\alpha(x)$, $x = c$, or $\neg(x = c)$, for some α and c. As in the proof of Lemma 2, we may assume the first two forms contribute exactly one literal: the literal $\alpha(x)$, for α the intersection of Σ, of all β from occuring literals $\beta(x)$, and of all $\overline{\gamma}$ from occuring literals $\neg\gamma(x)$. We may also assume that the third form contributes at most one literal for collectively all $c > 0$ and at most one literal for collectively all $c < 0$: if there are two literals $x = c_1$, $x = c_2$ for distinct $c_1, c_2 > 0$, then φ is always false, and thus the 2DFA is just the trivial one which simply halts and rejects —similarly for $c_1, c_2 < 0$.

Overall, without loss of generality, we may assume that $\varphi(x)$ consists of: exactly one $\alpha(x)$ for $\alpha \subseteq \Sigma$; an optional $x = c$ for $c \in \mathbb{Z}_\tau^+$; an optional $x = c$ for $c \in \mathbb{Z}_\tau^-$; and zero or more $\neg(x = c)$ for $c \in \mathbb{Z}_\tau^+ \cup \mathbb{Z}_\tau^-$.

To test $w[x/x^*] \models \varphi$, a 2DFA run on w from cell x^* first verifies $\alpha(x)$ by testing that $w_{x^*} \in \alpha$. It then scans left counting down from τ, until its counter is 0 or it sees $\vdash$ (whichever happens first), and then returns to cell x^*; during this trip, it tests the optional $x = c$ and the zero or more $\neg(x = c)$ for $c > 0$. It then performs a symmetric trip of $\leq \tau$ steps to the right of cell x^* and back, during which it tests the optional $x = c$ and the zero or more $\neg(x = c)$ for $c < 0$. Finally, it accepts if all tests succeeded. Easily, $O(\tau)$ states are enough. ☐

Fact 5. *Every not-identically-false floating relational $\wedge$-clause $\varphi(x, y)$ is equivalent to $\mathsf{S}(x, y)$, $x = y$, $\mathsf{S}(y, x)$, or a conjunction of $\neg\mathsf{S}(x, y)$, $\neg(x = y)$, $\neg\mathsf{S}(y, x)$.*

Lemma 4. *Suppose $\varphi(x, y)$ is an $\wedge$-clause of margin τ which does not contain both anchored local and floating relational atoms. Then there exists a $O(\tau)$-state* 2NFA *which, whenever run on a string w from a cell $1 \leq x^* \leq |w|$, computes so that, for all $1 \leq y^* \leq |w|$:*

$$\begin{array}{r}\textit{a computation path which}\\ \textit{halts \& accepts on cell } y^* \textit{ exists}\end{array} \iff w[x/x^*, y/y^*] \models \varphi(x, y)\,. \qquad (11)$$

Proof. If φ is trivial, then the 2NFA is also trivial. So, assume φ is non-trivial. Let $\phi, \chi(x), \psi(y), \omega(x, y)$ be the $\wedge$-clauses given by Fact 3. Since anchored local and floating relational atoms cannot co-exist, at least one of ϕ and ω is empty.

Case 1. Suppose ω is empty. Then, when run on w from cell x^*, our 2NFA N must create nondeterministic branches which collectively accept on every cell y^* such that $\phi \wedge \chi(x) \wedge \psi(y)$ holds if $x = x^*$ and $y = y^*$. For this, N first checks χ on cell x^*; then resets its head (forgetting x^*) and reads the ends of w to check ϕ; then sweeps w and, on every cell y^*, guesses and verifies that ψ is true on y^*.

Specifically, let Φ, X, Ψ be the $O(\tau)$-state 2DFAs given by Lemma 2 for ϕ and by Lemma 3 for χ and ψ, respectively. Starting on cell x^*, N first simulates X.

This brings it back to x^* having checked χ on x^*. Then N goes to $\vdash$ and starts simulating $\varPhi$. This brings it back to $\vdash$ having checked ϕ. Then N scans w and, on every cell y^*, spawns a new branch which simulates $\varPsi$, eventually returning to y^* having checked ψ on y^*. Finally, N accepts (in that branch) iff all checks succeeded. Easily, N satisfies (11) and has size $O(|\varPhi|+|X|+|\varPsi|) = O(\tau)$.

Case 2. Suppose ϕ is empty. Then φ is equivalent to $\chi(x) \wedge \psi(y) \wedge \omega(x, y)$, where ω is not identically false (since φ is non-trivial), and thus is equivalent to one of $\mathsf{S}(x, y)$, $x = y$, $\mathsf{S}(y, x)$, or to a conjunction of their negations (Fact 5).

2a. If ω is equivalent to $\mathsf{S}(x, y)$: Then the branches of N must collectively accept on every cell y^* such that $\chi(x) \wedge \psi(y) \wedge \mathsf{S}(x, y)$ holds when $x = x^*$ and $y = y^*$. Because of $\mathsf{S}(x, y)$, the only possible y^* of this kind is $x^*{+}1$. So, N should just accept on cell $x^*{+}1$ iff $\chi(x) \wedge \psi(y)$ holds when $x = x^*$ and $y = x^*{+}1$. Hence, N starts on x^* by simulating X. This brings it back to x^* having checked χ on x^*. Then it moves one cell to the right, checks that it is not $\dashv$, and starts simulating $\varPsi$, eventually returning to the cell, having checked ψ on $x^*{+}1$. In the end, N accepts iff all checks succeeded. Note that N is, in fact, deterministic.

2b. If ω is equivalent to $x = y$: Then φ is equivalent to $\chi(x) \wedge \psi(y) \wedge x = y$, so the only possible y^* is x^*. Hence, N works as in Case 2a, but without the one step to the right between the simulations of X and of $\varPsi$.

2c. If ω is equivalent to $\mathsf{S}(y, x)$: Then φ is equivalent to $\chi(x) \wedge \psi(y) \wedge \mathsf{S}(y, x)$, so the only possible y^* is $x^*{-}1$. So, N works as in Case 2a, except that, between the simulations of X and of $\varPsi$, it moves left and checks that it does not read $\vdash$.

2d. If ω is equivalent to a conjunction of $\neg\mathsf{S}(x, y)$, $\neg(x = y)$, $\neg\mathsf{S}(y, x)$: Then ω excludes a certain set of cells $Y_\omega \subseteq \{x^*{-}1, x^*, x^*{+}1\}$ from being accepted. So, N must accept on cell y^* iff $\chi(x) \wedge \psi(y)$ holds for $x = x^*, y = y^*$ and $y^* \notin Y_\omega$. As above, N starts on x^* by simulating X, and returns on it after checking χ on x^*. Then it spawns five branches, one for each of the five cases as to where cell y^* is with respect to cell x^*: before $x^*{-}1$, on $x^*{-}1$, on x^*, on $x^*{+}1$, or after $x^*{+}1$.

- In the first branch: N moves left by two cells, checking that neither is $\vdash$. It then sweeps up to $\vdash$ and, on each cell y^*, spawns a branch which simulates $\varPsi$ and eventually returns on y^* having checked ψ on it.
- In the second branch: If $x^*{-}1 \in Y_\omega$ (i.e., if ω contains $\neg\mathsf{S}(y, x)$), then N just rejects. Otherwise, it moves left once, checks that it is not on $\vdash$, then simulates $\varPsi$. This brings it back to the same cell $x^*{-}1$, having checked ψ on it.
- In the third and fourth branches: N works similarly to the second one. It just rejects, if $x^* \in Y_\omega$ (i.e., if ω contains $\neg(x = y)$) or if $x^*{+}1 \in Y_\omega$ (i.e., if ω contains $\neg\mathsf{S}(x, y)$), respectively. Otherwise, it simulates $\varPsi$ after, respectively, not moving at all or moving once to the right.
- In the last branch: N works symmetrically to the first one. It moves right by two cells checking against $\dashv$, and then simulates $\varPsi$ on each cell before $\dashv$.

In all cases, N accepts in a given branch iff all checks along it have succeeded.

Easily, in all four cases, N satisfies (11) and contains one copy of each of X and $\varPsi$, plus $O(1)$ more states, for a total size of $O(|X|+|Y|) = O(\tau)$. $\square$

Lemma 5. *Every weak* GA/DNF$_1$ *of degree m and margin τ is equivalent to a* 2NFA *with $O(m\tau)$ states.*

Proof. Let $\psi = \mathsf{TC}[\varphi(\overline{x},\overline{y})](\overline{s},\overline{t})$ be as in the statement. Let the arity be $k{+}1$. Then $s_0,\dots,s_k,t_0,\dots,t_k \in \mathbb{Z}_\tau^+ \cup \mathbb{Z}_\tau^-$ and the core φ has the form $\bigvee_{i=1}^m \varphi_i(\overline{x},\overline{y})$, where (Fact 2) each φ_i is equivalent to:

$$(x_1 = c_1^i) \wedge \dots \wedge (x_k = c_k^i) \;\wedge\; (y_1 = d_1^i) \wedge \dots \wedge (y_k = d_k^i) \;\;\wedge\;\; \hat{\varphi}_i(x_0,y_0)\,, \qquad (12)$$

for some constants $c_1^i,\dots,c_k^i,d_1^i,\dots,d_k^i \in \mathbb{Z}_\tau^+ \cup \mathbb{Z}_\tau^-$ and an $\wedge$-clause $\hat{\varphi}_i$ of margin τ where anchored local and floating relational atoms do not co-exist (as ψ is weak).

We build a 2NFA N which accepts an input $w \in \Sigma^*$ of length $n \ge 2$ iff $w \models \psi$, i.e., iff the graph $G_{\varphi,w}$ (see p. 5) has a path from vertex $\overline{s}$ to vertex $\overline{t}$. To check this, N nondeterministically guesses such a path *in stages*, in the standard way: starting each stage, it remembers only the last vertex $\overline{u}$ of the path guessed so far (originally, $\overline{u} := \overline{s}$); then it checks whether $\overline{u} = \overline{t}$ and, if so, accepts; otherwise, it nondeterministically selects a neighbor $\overline{v}$ of $\overline{u}$ and updates its memory to $\overline{u} := \overline{v}$, completing the stage. Below, we describe how N implements this algorithm.

Central in this implementation is how N remembers $\overline{u}$. Clearly, $\overline{u}$ will always be a vertex reachable from $\overline{s}$, so the following fact becomes important:

Claim. If $\overline{u}$ is reachable from $\overline{s}$, then $(u_1,\dots,u_k) = (s_1,\dots,s_k)$ or there is $i = 1,\dots,m$ such that $(u_1,\dots,u_k) = (d_1^i,\dots,d_k^i)$; either way, $u_1,\dots,u_k \in \mathbb{Z}_\tau^+ \cup \mathbb{Z}_\tau^-$.

Proof. If $\overline{u} = \overline{s}$, the claim is trivial. Suppose $\overline{u} \ne \overline{s}$. Then the path $\overline{s} \rightsquigarrow \overline{u}$ has ≥ 1 step. Let $\overline{v} \to \overline{u}$ be the last one. Then $(\overline{v},\overline{u})$ is an edge in $G_{\varphi,w}$, so $w[\overline{x}/\overline{v},\overline{y}/\overline{u}]$ satisfies $\varphi(\overline{x},\overline{y})$; hence, it satisfies some $\varphi_i(\overline{x},\overline{y})$; so, it satisifes the corresponding $(y_1 = d_1^i) \wedge \dots \wedge (y_k = d_k^i)$; which implies that $(u_1,\dots,u_k) = (d_1^i,\dots,d_k^i)$. $\boxdot$

So, N separates $\overline{u}$ into (1) its 'bounded components' $u_1,\dots,u_k \in \mathbb{Z}_\tau^+ \cup \mathbb{Z}_\tau^-$; and (2) its 'unbounded component' $u_0 \in \mathbb{Z}_n^+$. To remember (1), it keeps in its state an index $0 \le i \le m$ such that $(u_1,\dots,u_k) = (d_1^i,\dots,d_k^i)$ —for convenience, let $(d_1^0,\dots,d_k^0) := (s_1,\dots,s_k)$. To remember (2), it places its head on cell u_0 of w.

Overall, each state of N is of the form (i,σ), where i identifies (as described) the list $u_1,\dots,u_k$ and σ shows the status of the current stage. As a special case, $\sigma = \mathtt{B}$ means the stage has just begun. So, if N is in state $(i,\mathtt{B})$ on cell u^*, then it has reached vertex $\overline{u} = (u^*,d_1^i,\dots,d_k^i)$ and is now beginning the next stage.

With this representation, the search for a path $\overline{s} \rightsquigarrow \overline{t}$ takes N through configurations $((i_0,\mathtt{B}),u_0^*),((i_1,\mathtt{B}),u_1^*),\dots,((i_l,\mathtt{B}),u_l^*)$, where $u_0^* = s_0, i_0 = 0$; and the search succeeds iff $u_l^* = t_0$ and $(d_1^{i_l},\dots,d_k^{i_l}) = (t_1,\dots,t_k)$. To complete the description of N, we must explain how N navigates through these configurations.

In a special first stage, N alters its configuration from $(q_{\mathrm{s}},0)$ to $((i_0,\mathtt{B}),u_0^*) = ((0,\mathtt{B}),s_0)$. For this, it moves its head to cell s_0 (by counting s_0 steps from $\vdash$, if $s_0 > 0$; or by moving to $\dashv$ and counting s_0 steps backwards, if $s_0 < 0$) and switches to state $(0,\mathtt{B})$. Easily, this can be done with $O(s_0) = O(\tau)$ states.

From then on, whenever at a configuration $((i,\mathtt{B}),u^*)$, our N works as follows.

First, it checks if $\overline{u} = \overline{t}$, i.e., if (1) $u^* = t_0$ and (2) $(d_1^i,\dots,d_k^i) = (t_1,\dots,t_k)$. Check 2 is hardwired, so it needs no extra states. Check 1 involves a trip to the left (if $t_0 > 0$) or right (if $t_0 < 0$) for t_0 steps or up to the end-marker, and back to cell u^*. There, if both checks succeeded, N accepts; otherwise, it switches to a special state $(i,\mathtt{C})$. Overall, this uses $O(t_0) = O(\tau)$ states of the form $(i,\,.\,)$.

State $(i, \texttt{C})$ means that N is about to choose the next vertex $\overline{v}$ among the out-neighbors of $\overline{u}$, so as to switch to the appropriate next configuration $((\,.\,, \texttt{B}), \,.\,)$. Note that $\overline{v}$ is an out-neighbor of $\overline{u}$ iff $(\overline{u}, \overline{v})$ is an edge of $G_{\varphi,w}$; i.e., iff $w[\overline{x}/\overline{u}, \overline{y}/\overline{v}]$ satisfies some $\wedge$-clause $\varphi_j(\overline{x}, \overline{y})$ as in (12); i.e., iff there exists j such that

- the bounded components $v_1, \dots, v_k$ of $\overline{v}$ are equal to the second tuple of constants $d_1^j, \dots, d_k^j$ in one of the φ_j whose first tuple of constants $c_1^j, \dots, c_k^j$ are the bounded components $d_1^i, \dots, d_k^i$ of $\overline{u}$, namely:

$$(d_1^i = c_1^j) \wedge \cdots \wedge (d_k^i = c_k^j) \;\wedge\; (v_1 = d_1^j) \wedge \cdots \wedge (v_k = d_k^j)\,; \quad \text{and}$$

- the unbounded component v_0 of $\overline{v}$ together with the unbounded component u^* of $\overline{u}$ satisfy the respective $\hat{\varphi}_j$: $w[x_0/u^*, y_0/v_0] \models \hat{\varphi}_j(x_0, y_0)$.

So, to nondeterministically choose such a $\overline{v}$, our N works in two sub-stages:

- First, it chooses $v_1, \dots, v_k$, by simply choosing the index j of some $\wedge$-clause (if any) whose first tuple of constants is exactly $d_1^i, \dots, d_k^i$. This selection is hardwired and takes N to a special state $(j, \texttt{D})$ still on cell u^*.
- Then, it chooses v_0, by simulating the $O(\tau)$-state 2NFA given by Lemma 4 for $\hat{\varphi}_j$, from cell u^* up to every cell v^* such that $w[x_0/u^*, y_0/v^*] \models \hat{\varphi}_j(x_0, y_0)$. This needs $O(\tau)$ states of the form $(j, \,.\,)$ and ends at a state $(j, \texttt{B})$.

Overall, the result is a nondeterministic computation whose accepting branches take N to all configurations $((j, \texttt{B}), v^*)$ such that $\overline{v} = (v^*, d_1^j, \dots, d_k^j)$ is an out-neighbor of $\overline{u} = (u^*, d_1^i, \dots, d_k^i)$. This concludes our description of a full stage.

In total, N uses $O(\tau)$ states for the special first stage and, for each i, another $O(\tau) + O(1) + O(\tau) = O(\tau)$ states for every stage that starts after state $(i, \texttt{B})$. So, the total number of states is $O(\tau) + (1+m) \cdot O(\tau) = O(m\tau)$, as promised. $\square$

6 Conclusion

Completing [6], we descriptively characterized 2N. We can show that Theorem 1 is tight, in that its sentences can solve non-regular problems, if two-dimensional; or problems outside 2N, if the core is in CNF. It would be nice to see descriptive characterizations for other minicomplexity classes, too.

References

1. R. J. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6(1-6):66–92, 1960.
2. R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. M. Karp, editor, *Complexity of Computation*, volume VII of *AMS-SIAM Symposia in Applied Mathematics*, pages 43–73, 1974.
3. N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal of Computing*, 17(5):935–938, 1988.
4. N. Immerman. *Descriptive complexity*. Springer-Verlag, 1998.
5. C. Kapoutsis. Minicomplexity. *Journal of Automata, Languages and Combinatorics*, 17(2–4):205–224, 2012.
6. C. Kapoutsis and N. Lefebvre. Analogs of Fagin's Theorem for small nondeterministic finite automata. In *Proceedings of DLT*, pages 202–213, 2012.