Marking the chops: an unambiguous temporal logic

Kamal Lodaya, Paritosh K. Pandya and Simoni S. Shah

Abstract Interval Temporal Logic [11, 13] is a highly expressive and succinct logic whose satisfiability over finite words is non-elementary in the number of alternations of chop and negation operators. All the sublogics of *ITL* with elementary decidability known to us restrict this alternation depth. In this paper, we define a sublogic of Interval Temporal Logic by replacing chops with marked chops but without any restriction on the alternation depth. We show that the resulting logic admits unique parsing of a word matching a formula, with the consequence that membership is in LogDCFL, and satisfiability is in PSPACE (and NP-complete for a fixed alphabet). As our first result, we give an *effective* model-preserving reduction from *UITL* to the partially ordered two-way deterministic finite automata of Schwentick, Thérien and Vollmer [14]. We show that the size of the resulting automaton is quadratic in the size of the formula. We also have an exponential converse reduction from *po2dfa* to *UITL*. It follows from the work of Schützenberger [12], Thérien and Wilke [19] that this unambiguous *ITL* has same expressive power as the first-order logic with two variables [10].

1 Introduction

Two-variable first-order logic FO^2 was first studied by Mortimer [10]. In recent years, a lot of research has centred around this logic, on words [5], data [1], Mazurkiewicz traces [7], trees [2], etc. In particular for words, the article by Tesson and Thérien [18] reveals the many facets of the class of languages defined by sentences of this logic. The logic was shown to be NEXPTIME-complete and equiv-

Kamal Lodaya
The Institute of Mathematical Sciences, CIT Campus, Chennai *600113*, India

Paritosh K. Pandya and Simoni S. Shah
Tata Institute of Fundamental Research, Colaba, Mumbai *400005*, India
Correspondence e-mail: pandya@tcs.tifr.res.in

alent to a natural fragment of linear temporal logic called *Unary TL* by Etessami, Vardi and Wilke [5] and to partially ordered two-way deterministic finite automata (henceforth *po2dfa*) by Schwentick, Thérien and Vollmer [14]. Thérien and Wilke showed [19] that it corresponds to the variety DA of unambiguous languages studied by Schützenberger [12]. Weis and Immerman [20] and Kufleitner and Weil [8] have recently examined the quantifier alternation hierarchy within $FO^2[<]$.

A proper treatment of syntax, we feel, is lacking. Tesson and Thérien's paper [18] does give a rudimentary syntax in terms of deterministic and co-deterministic products, which we close under boolean operations and call an deterministic or unambiguous subclass of propositional interval temporal logic *ITL*.

ITL is a highly succinct logic for specifying properties of finite words [11, 13]. The unconstrained chop operator (similar to concatenation of star-free expressions) leads to high decision complexity: the satisfiability of *ITL* is non-elementary in the number of alternations of the negation and chop operator [17]. Sublogics of *ITL* with elementary satisfiability have been obtained by constraining this alternation depth in some manner. *UITL* replaces chops with marked chops but without any restriction on their alternation depth with negation. Our first theorem is a consequence of the unique parsability: membership of a word *w* in the language of a formula is in LOGDCFL and nonemptiness is in NP for a fixed alphabet.

Also exploiting this unique parsability, an effective quadratic translation from *UITL* to $FO^2[<]$ has been given by Shah [15]. From the work of Thérien and Wilke [19] it follows that *UITL* is expressively contained in in the unambiguous languages of Schützenberger [12]. That it was an open problem whether this *UITL* syntax matches the expressive power of FO^2 we learnt from [8], which was written concurrently and independently of this paper. We answer the question positively in this paper.

Our second theorem is an $O(n^2)$ translation from a formula of our logic to a *po2dfa* which accepts exactly the models of the formula. A partially ordered 2DFA [14] (also called linear by Löding and Thomas [9]) is a two-way DFA which has the property that once the automaton exits a state, it is never entered again. The translation from formulae to automata illustrates the difficulty of working with weak models such as *po2dfa*. To complete the characterisation of the expressive power, as our third theorem we construct for each *po2dfa* a formula exactly specifying its language. This solves the open problem mentioned above, as does the paper [8] using completely different techniques. The constructed formula is exponential in the size of the automaton.

$FO^2[<]$ and *Unary TL* are at a remove from the very deterministic notion of *po2dfa*. Our logic, which can be thought of as *ITL* but where the chop operator is forced to be deterministic, is much closer to the automata. As a consequence, satisfiability drops from nonelementary for *ITL* to PSPACE for our logic. In earlier work [6], we found that such unambiguity considerably improves the computational performance of a validity checking tool for *ITL*.

The idea of having deterministic temporal logics has been explored before. A "marked" operator in temporal logic *atnext* was studied by Borchert and Tesson [3]. Kufleitner simplified it to deterministic marked next and previous modalities X_a and

Y_a to define a point-based linear temporal logic, and showed that it is expressively complete for $FO^2[<]$ over Mazurkiewicz traces (and hence also over words) [7]. However, a concrete exploitation of this to give explicit and efficient reduction from logic to automata seems new.

The rest of the paper is organized as follows. Section 2 defines the syntax and semantics of *UITL*. Section 3 discusses the partially ordered 2DFA and some expressions we use as a convenient notation for them. Section 4 gives the reduction from formulae of *UITL* to *po2dfa*. Section 5 gives the construction of a formula specifying the language accepted by a *po2dfa*.

Acknowledgements The authors thank Manfred Kufleitner, Pascal Weil and Meena Mahajan for valuable inputs. The first author acknowledges the Indo-French project *Timed-Discoveri* for support. The second and the third authors acknowledge partial support from the Microsoft Research Grant for the project "formal specification and analysis of component-based designs."

2 Unambiguous interval temporal logic: its syntax and semantics

We propose a fragment of *ITL* [11] where the chop operator is replaced by marked chop operators F_a and L_a. Our syntax derives from closing the $*_{n,k}$-expressions of Tesson and Thérien [18] under Boolean operations.

Fix an alphabet Σ. Let $a \in \Sigma, A \subseteq \Sigma$. Let D, D_1, D_2 range over formulas in *UITL*. The abstract syntax of *UITL* is given below.

$$\lceil\lceil A \rceil\rceil \mid \lceil\lceil A \rceil \mid \lceil A \rceil\rceil \mid \lceil A \rceil \mid D_1 \vee D_2 \mid \neg D \mid D_1 F_a D_2 \mid D_1 L_a D_2 \mid \oplus D \mid \ominus D$$

Let w be a nonempty finite word over Σ and let $pos(w) = \{1, \ldots, \#w\}$ be the set of positions. Let $INTV(w) = \{[i,j] \mid i,j \in pos(w), i \leq j\}$ be the set of intervals overs w. The satisfaction of a formula D is defined over intervals of a word model w as follows.

$w,[i,j] \models \lceil\lceil A \rceil\rceil$ iff for all $k : i \leq k \leq j.\ \ w[k] \in A$
$w,[i,j] \models \lceil A \rceil$ iff for all $k : i < k < j.\ \ w[k] \in A$
$w,[i,j] \models \lceil\lceil A \rceil$ iff for all $k : i \leq k < j.\ \ w[k] \in A$
$w,[i,j] \models \lceil A \rceil\rceil$ iff for all $k : i < k \leq j.\ \ w[k] \in A$
$w,[i,j] \models D_1 F_a D_2$ iff for some $k : i \leq k \leq j.\ \ w[k] = a$ and
(for all $m : i \leq m < k.\ w[m] \neq a$) and
$w,[i,k] \models D_1$ and $w,[k,j] \models D_2$
$w,[i,j] \models D_1 L_a D_2$ iff for some $k : i \leq k \leq j.\ \ w[k] = a$ and
(for all $m : k < m \leq j.\ w[m] \neq a$) and
$w,[i,k] \models D_1$ and $w,[k,j] \models D_2$
$w,[i,j] \models \oplus D$ iff $i < j$ and $w,[i+1,j] \models D$
$w,[i,j] \models \ominus D$ iff $i < j$ and $w,[i,j-1] \models D$

As usual, $w \models D$ iff $w, [1, \#w] \models D$ and $L(D) \overset{\text{def}}{=} \{w \mid w \models D\}$ is the language defined by D.

The proposition $\lceil\lceil A \rceil\rceil$ states that letters of all positions in the interval (including the endpoints) are in A. Similarly, $\lceil A \rceil$ says that all the strictly interior positions in an interval have only letters from A; thus it trivially holds for point (i.e. $[i,i]$) and unit (i.e. $[i, i+1]$) intervals. By similar reasoning, $\lceil\ \rceil \overset{\text{def}}{=} \lceil\lceil \emptyset \rceil$ holds only on point intervals, and $\neg\lceil\lceil \emptyset \rceil \wedge \lceil \emptyset \rceil$ only on unit intervals. The semantics of the "first" and "last" marked chops and the "next" and "previous" operators should be clear.

The derived operators $\wedge, \supset, \Leftrightarrow$ have their usual definitions. The constant $\top$ (denoting *true*) can be defined as $\lceil\lceil \Sigma \rceil\rceil$. We take both these to be of constant size, but in general the size of $\lceil\lceil A \rceil\rceil$ is $O(|A|)$. Conversely, $\lceil\lceil A \rceil\rceil \Leftrightarrow \bigwedge_{a \notin A} \neg(\top F_a \top)$. Similar equivalences can be given for $\lceil\lceil A \rceil$, $\lceil A \rceil\rceil$ and $\lceil A \rceil$. Negations can be pushed inwards to the level of literals using $\neg(D_1 F_a D_2) \Leftrightarrow (\lceil\lceil \Sigma \setminus \{a\} \rceil\rceil \vee (\top F_a \neg D_2) \vee (\neg D_1 F_a \top))$ and $\neg(\oplus D) \Leftrightarrow \lceil\ \rceil \vee \oplus(\neg D)$. All these translations are linear in the size of the formula. For later use in Section 5, we also designate as simple formulae those made of the atomic formulae and the marked chop operators F_a and L_a as well as operators $\ominus$ and $\oplus$ with the Boolean operators being disallowed.

Example 1. Consider the formula $D \overset{\text{def}}{=} (\top L_a((\neg(\top F_b \top)) F_d \top))$ over the alphabet $\Sigma = \{a, b, c, d\}$. Intuitively, it states that between the last occurrence of a and subsequent first occurrence of d there is no occurrence of letter b. Thus it specifies the language $\Sigma^* a c^* d \{b, c, d\}^*$. □

Example 2. Formula $\lceil\ \rceil F_a(\oplus(\lceil\ \rceil F_a(\oplus(\lceil\ \rceil F_a \lceil\ \rceil))))$ holds exactly for the word aaa. Note that it is impossible to express this without using $\oplus$ or $\ominus$ operators. □

Following Kufleitner and Weil [8], we define two hierarchies of formulae. $R_1 = L_1$ consists of the formulae made up of the four kinds of atomic formulae and the Boolean operations, marked chops being disallowed. R_{n+1} extends L_n by allowing F_a operators (deterministic products) over formulas of L_n and closing under Boolean operations; symmetrically, L_{n+1} is the Boolean closure of L_a operators (co-deterministic products) over R_n. Thus $UITL = \bigcup_n R_n = \bigcup_n L_n$ is the full deterministic/co-deterministic hierarchy over the piecewise testable languages of Simon (which are characterized by R_1 [16], e.g. see the survey of Diekert, Gastin and Kufleitner [4]). R_1, R_2 and L_2 are known as J, R and L in the literature.

2.1 Unique parsing

It is convenient to represent *UITL* a formula D by its syntax tree, where each interior node n is labelled by an operator and the subtree rooted at n represents a subformula of D, denoted by $\mathit{Subf}(n)$. For the root of the tree, $\mathit{Subf}(\mathit{root}) = D$. For example, a node n with $\mathit{Subf}(n) = D_1 F_a D_2$ has two children, say n_1 and n_2 with $\mathit{Subf}(n_1) = D_1$ and $\mathit{Subf}(n_2) = D_2$. We will say n matches $n_1 F_a n_2$. A leaf node n is labelled by one of $\lceil\lceil A \rceil\rceil, \lceil\lceil A \rceil, \lceil A \rceil\rceil, \lceil A \rceil$.

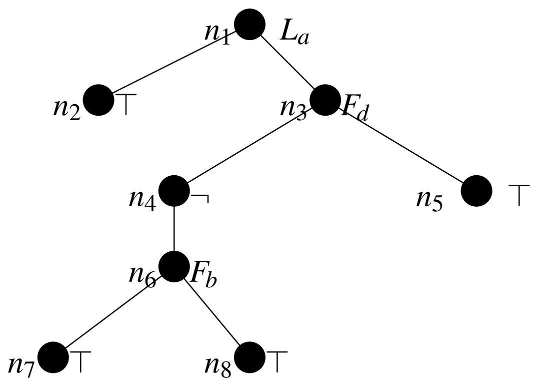

Fig. 1 Syntax Tree of Formula in Example 1

Fix a formula D and let *Nodes* be the set of nodes in its syntax tree with *root* being the root node. Let $MNodes \subset Nodes$ be the subset of nodes whose operator has the form F_a, L_a, $\oplus$, or $\ominus$. For any node n, let $Ancestry(n)$ be the sequence of *Nodes* encountered on the unique path from n to the root node. For technical convenience we will append two fresh nodes $n_\triangleright$ followed by $n_\triangleleft$ to the ancestry. Formally, $Ancestry(root) = n_\triangleright.n_\triangleleft$. Also if n matches $n_1 \ op \ n_2$, then $Ancestry(n_1) = Ancestry(n_2) = n.Ancestry(n)$. We will follow the convention that $n_\triangleright$ is an $L_\triangleright$ operator and $n_\triangleleft$ is an $F_\triangleleft$ operator. Let $\ell Ancestry(n)$ be the subsequence of nodes from n to the root which are labelled with marked chops or $\ominus$ such that n is in their right subtree; $rAncestry(n)$ is similarly defined with left subtrees, marked chops and $\ominus$.

Example 3. Consider the formula D in Example 1. Figure 1 gives the syntax tree of D. At n_4, we have $Subf(n_4) = \neg(\top F_b \top)$. It is easy to see that $Ancestry(n_7) = n_6 n_4 n_3 n_1 n_\triangleright n_\triangleleft$ with $rAncestry(n_7) = n_6 n_3 n_\triangleleft$ and $\ell Ancestry(n_7) = n_1 n_\triangleright$. □

Next we consider the evaluation of D over a word w. For any word w and any subformula of a formula D we can associate a unique interval (or none) where the formula must be evaluated. This interval is fixed by the context in which the subformula occurs and does not depend upon the subformula itself. For example, the subformula $D_1 = \neg(\top F_b \top)$ of D in Example 1 is associated with the interval which begins with the last occurrence of a in w and it ends at the first subsequent occurrence of d. We call this property *unique parsability*. Formally, given word w, we can associate with each $n \in Nodes$ either a unique interval $[i, j]$ where $Subf(n)$ needs to be evaluated, or $\mathbf{u}$ denoting that the subformula of the node need not be evaluated. This is denoted by $Intv_w(n)$. Moreover, for each $n \in MNodes$ (which corresponds to a marked chop operator) we associate a chopping position $cPos_w(n)$.

Definition 1. $Intv_w : Nodes \rightarrow INTV(w) \cup \{\mathbf{u}\}$ and $cPos_w : MNodes \rightarrow pos(w) \cup \{\mathbf{u}\}$ are defined by induction on the depth of the node (from root) as follows.

- $Intv_w(root) = [1, \#w]$.

- If n matches $n_1 \vee n_2$ then $Intv_w(n_1) = Intv_w(n_2) = Intv_w(n)$. Similarly, If n matches $\neg n_1$ then $Intv_w(n_1) = Intv_w(n)$.
- If n matches $n_1 F_a n_2$ or $n_1 L_a n_2$ or $\ominus n_1$ or $\oplus n_1$ and $Intv_w(n) = \mathbf{u}$ then $Intv_w(n_1) = Intv_w(n_2) = \mathbf{u}$ and $cPos_w(n) = \mathbf{u}$.
- If n matches $n_1 F_a n_2$ or $n_1 L_a n_2$, $Intv_w(n) = [i,j]$ and if for all $k : i \le k \le j$ we have $w[k] \ne a$ then $Intv_w(n_1) = Intv_w(n_2) = \mathbf{u}$ and $cPos_w(n) = \mathbf{u}$.
- Let n match $n_1 F_a n_2$ with $Intv_w(n) = [i,j]$. Let $k : i \le k \le j$ be such that $w[k] = a$ and for all $m : i \le m < k$ we have $w[m] \ne a$. Then, $Intv_w(n_1) = [i,k]$ and $Intv_w(n_2) = [k,j]$. Also, $cPos_w(n) = k$.
- n matches $n_1 L_a n_2$ with $Intv_w(n) = [i,j]$. Let Let $k : i \le k \le j$ be such that $w[k] = a$ and for all $m : k < m \le j$ we have $w[m] \ne a$. Then, $Intv_w(n_1) = [i,k]$ and $Intv_w(n_2) = [k,j]$. Also, $cPos_w(n) = k$.
- If n matches $\oplus n_1$ or $\ominus n_1$ and $Intv_w(n) = [i,i]$ then $intv(n_1) = \mathbf{u}$ and $cPos_w(n) = \mathbf{u}$.
- If n matches $\oplus n_1$ and $Intv_w(n) = [i,j]$ with $i < j$ then $intv(n_1) = [i+1, j]$ and $cPos_w(n) = i+1$.
- If n matches $\ominus n_1$ and $Intv_w(n) = [i,j]$ with $i < j$ then $intv(n_1) = [i, j-1]$ and $cPos_w(n) = j-1$.

These definitions are extended to endmarker nodes $n_\triangleright$ and $n_\triangleleft$ as follows.
$Intv_w(n_\triangleright) = Intv_w(n_\triangleleft) = [1, \#w]$. Also, $cPos_w(n_\triangleright) = 1$ and $cPos_w(n_\triangleleft) = \#w$.

Proposition 1. *Let $Ancestry(n) = n_1, n_2, \ldots, n_k$ for a node n. For all i,j such that $1 \le i \le j \le k$, $Intv_w(n_i)$ is $\mathbf{u}$ or included in $Intv_w(n_j)$. Also, if n_i is labelled F_a or L_a and $cPos_w(n_i) \ne \mathbf{u}$ then $w[cPos_w(n_i)] = a$.* ☐

Using the notion of unique interval associated with a node, we can define the truth value of a node n in word w as follows.

Definition 2. Define $Val_w : Nodes \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ as follows. (Observe that $Val_w(root)$ is never $\mathbf{u}$ since $Intv_w(root)$ is always $[1, \#w]$.)

$$\begin{array}{l}
Val_w(n) = \mathbf{u} \text{ iff } Intv_w(n) = \mathbf{u} \\
Val_w(n) = \mathbf{t} \text{ iff } Intv_w(n) = [i,j] \text{ and } w,[i,j] \models Subf(n) \\
Val_w(n) = \mathbf{f} \text{ iff } Intv_w(n) = [i,j] \text{ and } w,[i,j] \not\models Subf(n)
\end{array}$$

Example 4. Consider the formula D with syntax tree as given in Figure 1. Consider the word $w = acdabacbcdbcbd$ with $pos(w) = \{1, \ldots, 14\}$. Then, we have $Intv_w(n_1) = [1,14]$. As n_1 is labelled L_a we have $cPos_w(n_1) = 6$ and $Intv_w(n_2) = [1,6]$ and $Intv_w(n_3) = [6,14]$. Also, n_3 is labelled F_d and we get $cPos_w(n_3) = 10$. This gives us $Intv_w(n_4) = [6,10]$ and $Intv_w(n_5) = [10,14]$. Then, $Intv_w(n_6) = [10,14]$ and as n_6 is labelled F_b we have $cPos_w(n_6) = 6$ and $Intv_w(n_7) = [6,8]$ and $Intv_w(n_8) = [8,10]$. Note that $n_4 = \neg n_6$ and $n_6 = n_7 F_b n_8$ and $n_7 = \top$ and $n_8 = \top$. Hence, $Val_w(n_7) = \mathbf{t}$, $Val_w(n_7) = \mathbf{t}$ giving $Val_w(n_6) = \mathbf{t}$ and $Val_w(n_4) = \mathbf{f}$. Similarly we can compute that $Val_w(n_1) = \mathbf{f}$. ☐

Theorem 1. *Membership of a word w in the language of a formula D is* NC^1*-hard and in the class* LOGDCFL. *Nonemptiness of the language of a formula D is* NP*-hard and in* NP *if the size of the alphabet Σ is fixed.*

Proof. After pushing negations inward and constructing the syntax tree, the Val_w function can be evaluated by a 2DPDA with auxiliary storage $O(\log(|D| + |w|))$ in time $O(poly(|D| + |w|))$. This yields a LOGDCFL procedure. If a formula D is satisfiable, we can translate it to a sentence of FO^2 of quadratic size [15] and use Weis and Immerman's result [20] to show the existence of a model of size $O((|D|^2)^{|\Sigma|})$. For a fixed size alphabet Σ, guessing the model and verifying its truth value is an NP procedure.

For the lower bounds, a Boolean assignment over n variables can be coded as a word of length n over a two-letter alphabet. The truth value of the i'th variable can be accessed using the $\oplus$ modality, which is also used to say that a model is of size n. These formulas are linear in n. Hence, any Boolean formula can be encoded as a *UITL* formula by replacing each variable by its corresponding *UITL* formula. Now we use the standard results for Boolean formulas. □

2.2 Handling context

We can further refine the characterisation of the intervals of a node. The following lemma relates intervals of nodes in an ancestry to their chopping positions in the same ancestry. Figure 2 depicts some of these relationships.

Let w be a word and let $i, j \in pos(w)$ with $i \leq j$. Then, $a \notin w[i:j)$ will abbreviate $\forall k : i \leq k < j.\ a \neq w[k]$. Similarly, we can define $a \notin w(i:j]$.

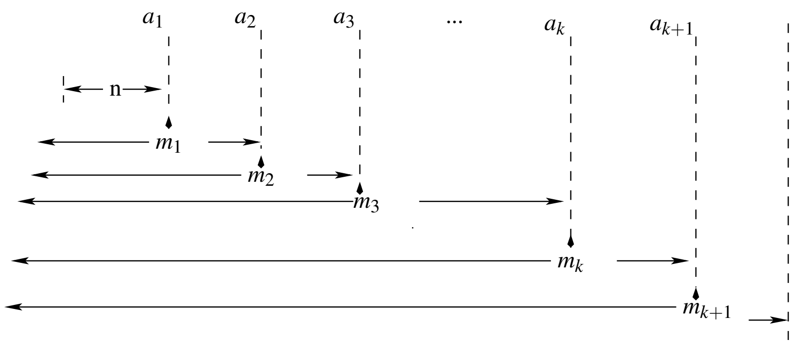

Fig. 2 Right handle and its intervals

Lemma 1. *For a node n, let* $\ell Ancestry(n) = n_1, n_2, \ldots, n_p, n_{p+1}$ *and* $rAncestry(n) = m_1, m_2, \ldots, m_r, m_{r+1}$. *(So* $n_{p+1} = n_{\triangleright}$ *and* $m_{r+1} = n_{\triangleleft}$.*) Then,*

- $Intv_w(n) = [cPos_w(n_1), cPos_w(m_1)]$. *If either of these chopping positions is* ***u*** *then* $Intv_w(n) = \boldsymbol{u}$.

- $cPos_w(n_1) \geq cPos_w(n_2) \geq \ldots \geq cPos_w(n_p) \geq cPos_w(n_{p+1}) = 1$, *and* $cPos_w(m_1) \leq cPos_w(m_2) \leq \ldots \leq cPos_w(m_r) \leq cPos_w(m_{r+1}) = \#w$.
- $Intv_w(n_i) = [cPos_w(n_{i+1}), cPos_w(m_k)]$ *for some* $m_k \in rAncestry(n)$. *Also,* $Intv_w(m_i) = [cPos_w(n_k), cPos_w(m_{i+1})]$ *for some* $m_k \in \ell Ancestry(n)$.
- *If* m_i *is a* L_a *node then* $w[cPos_w(m_i)] = a$ *and* $a \notin w(cPos_w(m_i) : cPos_w(m_{i+1})]$.
- *If* m_i *is a* F_a *node then* $w[cPos_w(m_i)] = a$ *and* $a \notin w[cPos_w(n_1) : cPos_w(m_i))$.
- *If* m_i *is* $\ominus$ *node then* $cPos_w(m_{i+1}) = cPos_w(m_i) + 1$.
- *If* n_i *is* $\oplus$ *node then* $cPos_w(n_{i+1}) = cPos_w(n_i) - 1$.

Proof. By induction on depth of n from the root. □

As opposed to the bottom-up evaluation of truth value, the identification of chopping positions and subintervals is defined top-down. This enables us to find the context necessary for checking whether a position m is within $Intv_w(n)$.

Definition 3. Let $\ell Handle(n)$ be the smallest prefix of $\ell Ancestry(n)$ ending with an L operator. Symmetrically let $rHandle(n)$ smallest prefix of $rAncestry(n)$ ending with an F operator. The sequence of labels of $rHandle(n)$ will have the form $H_1 H_2 \ldots H_k F_{a_{k+1}}$ where H_i is either L_{a_i} or $\ominus$. When clear from context we will often directly refer to such a sequence of labels as $rHandle(n)$. Given $rHandle(n)$, and indices i, j such that $1 \leq i \leq j \leq k+1$ let $rGap(n,i,j)$ be the count of $\ominus$ labels occurring within labels H_i to H_{j-1}. For example, given $rHandle(n) = L_{a_1} \ominus\ominus\ominus F_{a_5}$ we have $rGap(n,1,4) = 2$. Symmetrically, we can define $lGap(n,i,j)$ for $\ell Handle(n)$. □

In our running example, $rHandle(n_7) = F_b$ (the label of n_6) and $\ell Handle(n_7) = L_a$, the label of n_1.

Definition 4. Let $Intv_w(n) = [i,j]$, $rHandle(n) = H_1 H_2 \ldots H_k F_{a_{k+1}}$ as in Definition 3 and let m be a position. Then define $rwithin(n,m)$ as follows, and $lwithin(n,m)$ symmetrically.

$$\begin{aligned} rwithin(n,m) \stackrel{\text{def}}{=} \; & \exists r_1 \leq r_2 \leq \ldots \leq r_k \leq r_{k+1}. \quad (i \leq m \leq r_1) \\ & \text{and } (\forall 1 \leq p \leq k+1.\ (H_p = L_{a_p} \Rightarrow w[r_p] = a_p)) \\ & \text{and } (\forall 1 \leq i \leq j \leq k+1.\ (r_j - r_i \geq rGap(n,i,j)) \\ & \text{and } w[r_{k+1}] = a_{k+1} \notin w[i : r_{k+1}) \end{aligned}$$

Lemma 2 (Context). *Let* $Intv_w(n) = [i,j]$. *Then, for all* $m \in pos(w)$ *we have (a)* $i \leq m \leq j$ *iff* $rwithin(n,m)$, *and (b)* $i \leq m \leq j$ *iff* $lwithin(n,m)$.

Proof. We prove (a). Let $Intv_w(n) = [i,j]$ and $rHandle(n) = n_1 \ldots n_{k+1}$ with labels $H_1 H_2 \ldots H_k F_{a_{k+1}}$. Let $j_p = cPos_w(n_p)$ for $1 \leq p \leq k+1$. As $Intv_w(n) \neq \mathbf{u}$ we also have that $cPos_w(n_p) \neq \mathbf{u}$ as n_p is ancestor of n.

For the forward direction, suppose $i \leq m \leq j$. Take $r_p = j_p$. Then, by Lemma 1, we have $r_1 \leq r_2 \leq \ldots \leq r_{k+1}$ and $w[r_p] = a_p$, for all p with $H_p = L_{a_p}$. Also, by Lemma 1, when n_p has label $\ominus$, then $j_{p+1} = j_p + 1$. Hence, for any $1 \leq p \leq q \leq n$, we have $r_q - r_p \geq rGap(n,p,q)$. Also denote $Intv_w(n_{k+1}) = [b_{k+1}, e_{k+1}]$ then $b_{k+1} \leq i$.

Since n_{k+1} is labelled $F_{a_{k+1}}$, from its semantics we have that $a_{k+1} \notin w[i : j_{k+1})$. Hence the result follows.

Conversely, suppose there exist $r_1 \leq r_2 \leq \ldots \leq r_{k+1}$ such that $i \leq m \leq r_1$ and for all $1 \leq p \leq k+1$ if $H_p = L_{a_p}$ then $w[r_p] = a_p$ and for all $1 \leq p \leq q \leq k+1$ we have $r_q - r_p \geq rGap(n.p,q)$. We have to show that $m \leq j$. Assume to the contrary that $m > j$. By Lemma 1, $j_1 = j$ and $r_1 \geq m$. Hence, $r_1 > j_1$.

Consider any $1 \leq p \leq k$ such that $r_p > j_p$. There are two cases. In case 1, if p is labelled L_{a_p} then by its semantics $a_p \notin w(j_p, j_{p+1}]$. Hence, as $r_p > j_p$ and $w[r_p] = a_p$, it follows that $r_p > j_{p+1}$ which implies that $r_{p+1} > j_{p+1}$. In the second case, if p is labelled with $\ominus$ then $rGap(n,p,p+1) = 1$ and hence $r_{p+1} - r_p \geq 1$. Also, by Lemma 1, we have $j_{p+1} = j_p + 1$. Hence, as $r_p > j_p$, it follows that $r_{p+1} > j_{p+1}$.

We already have that $r_1 > j_1$. Hence by induction and using the previous step we can prove that $r_{k+1} > j_{k+1}$. But by the condition that $w[r_{k+1}] = a_{k+1}$ and $a_{k+1} \notin w[i : r_{k+1})$ we have that $j_{k+1} = r_{k+1}$, which is a contradiction. Hence we conclude that $m \leq j$. □

3 Partially ordered two-way deterministic finite automata

Partially ordered two-way DFA were introduced by Schwentick, Thérien and Vollmer [14] to characterize the unambiguous languages. We present a variant of their definition and propose a set of operators to compose these automata. Let $\Sigma' = \Sigma \cup \{\triangleright, \triangleleft\}$ include two endmarkers. Given $w \in \Sigma^*$, the two way automaton actually scans string $w' = \triangleright w \triangleleft$ with letters $\triangleright$ and $\triangleleft$ at positions 0 and $|w|+1$ respectively.

Definition 5. A *po2dfa* over Σ' is a tuple $M = (Q, \leq, \delta, s, t, r)$ where $(Q, \leq)$ is a poset of states such that r, t are the only minimal elements. s is the initial state, t is the accept state and r is the rejecting state. The set $Q \setminus \{t, r\}$ is partitioned into Q_l and Q_r (the states reached from the left and the right) with $s \in Q_l$. $\delta : ((Q_l \cup Q_r) \times \Sigma) \to Q) \cup ((Q_l \times \{\triangleleft\}) \to Q \setminus Q_r) \cup ((Q_r \times \{\triangleright\}) \to Q \setminus Q_l)$ is a transition function satisfying $\delta(q,a) \leq q$. □

If M is in a state q, reading a symbol a, it enters a state $\delta(q,a)$, and moves its head to the right if $\delta(q,a) \in Q_l$, left if $\delta(q,a) \in Q_r$, and stays in the same position if $\delta(q,a) \in \{t, r\}$. The transition function is designed to ensure that the automaton does not "fall off" either end of the input. A transition with $\delta(q,a) < q$ is said to make progress.

A po2dfa M running over word w is said to be in a configuration (q, p) if it is in a state q and head reading the position p in word. The run of a po2dfa M on an input word w starting with input head position p_0 is a sequence $(q_0, p_0), (q_1, p_1), \ldots (q_f, p_f)$ of configurations such that:

- $q_0 = s$ and $q_f \in \{t, r\}$, for all $i (1 \leq i < l)$, $\delta(q_i, w(p_i)) = q_{i+1}$, and
- $p_{i+1} = p_i + 1$ if $q_{i+1} \in Q_l$ or $p_{i+1} = p_i - 1$ if $q_{i+1} \in Q_r$.

We abbreviate such a run by writing $M(w,p_0) = (q_f, p_f)$. The run is accepting if $q_f = t$; rejecting if $q_f = r$. A pass is a contiguous partial run where the automaton moves in one direction. An n-pass automaton is one which makes at most n passes on any input before accepting or rejecting. The automaton M is said to be start-free if for any w, M accepts w from some position iff M accepts w starting from any position.

3.1 Composition of automata

For the description of *po2dfa* we will use turtle expressions, which are extensions of the turtle programs introduced by Schwentick, Thérien and Vollmer [14]. The syntax follows and we explain the semantics below. Let A, B range over subsets of Σ'.

$$E ::= Acc \mid Rej \mid A \xrightarrow{1} \mid A \xleftarrow{1} \mid A \xrightarrow{B} \mid A \xleftarrow{B} \mid E_1 ? E_2, E_3$$

Automaton Acc accepts immediately without moving the head. Similarly, Rej rejects immediately. $A \xrightarrow{B}$ accepts at the next occurrence of a letter from B to the right, maintaining the constraint that the intervening letters are from $A \setminus B$. If no such occurrence exists the automaton rejects at the right end-marker or if a letter outside A intervenes, the automaton rejects at its position. Automaton $A \xrightarrow{1}$ accepts one position to the right if the current letter is from A, else rejects at the current position. $A \xleftarrow{B}$ and $A \xleftarrow{1}$ are symmetric in the leftward direction. The conditional construct $E_1 ? E_2, E_3$ first executes E_1 on w. On its accepting w at position j it continues with execution of E_2 from j. On E_1 rejecting w at position j it continues with E_3 from position j.

Here are some abbreviations which illustrate the power of the notation: $E_1 ; E_2 = E_1 ? E_2, Rej$, $\quad \neg E_1 = E_1 ? Rej, Acc$. Moreover, if E_2 is start-free then $E_1 \vee E_2 = E_1 ? Acc, E_2$ and $E_1 \wedge E_2 = E_1 ? E2, Rej$. Notice that automata for these expressions are start-free if E_1 is start-free. We will use $A \xrightarrow{a}$ for $A \xrightarrow{\{a\}}$, $\xrightarrow{a}$ for $(\Sigma' \xrightarrow{a})$ and $\xrightarrow{1}$ for $(\Sigma' \xrightarrow{1})$. Similarly define $\xleftarrow{a}$ and $\xleftarrow{1}$. We will use the convention that $\overline{a_1, \ldots, a_k}$ denotes $\Sigma' \setminus \{a_1, \ldots, a_k\}$.

Proposition 2. *Given turtle expression E we can construct a po2dfa accepting the same language with number of states linear in* $|E|$.

We have to resort to Section 2 for the correctness of the next construction.

Definition 6. Consider a node n with $rHandle(n) = H_1 H2 \ldots H_k F_{a_{k+1}}$ as in Definition 3 and let $A \subseteq \Sigma'$. Define the one-pass automata $\mathscr{C}^+(n, A)$ and $\mathscr{C}^+(n, \xrightarrow{1})$ as follows, and symmetrically also $\mathscr{C}^-(n, A)$ and $\mathscr{C}^-(n, \xleftarrow{1})$. Let $perf(H_i)$ be $(\overline{a_{k+1}} \xrightarrow{a_i})$ if $H_i = L_{a_i}$ and $\overline{a_{k+1}} \xrightarrow{1}$ if $H_i = \ominus$.

$$\begin{aligned}\mathscr{C}^+(n,A) &= (\overline{a_{k+1}} \xrightarrow{A}); perf(H_1);\ldots;perf(H_k);(\overline{a_{k+1}} \xrightarrow{a_{k+1}})\\ \mathscr{C}^+(n,\xrightarrow{1}) &= (\overline{a_{k+1}} \xrightarrow{1}); perf(H_1);\ldots;perf(H_k);(\overline{a_{k+1}} \xrightarrow{a_{k+1}})\end{aligned}$$

Since $rHandle(n)$ and $\ell Handle(n)$ are linear in the depth of n it follows that the sizes of the $\mathscr{C}^-(n), \mathscr{C}^+(n)$ automata are also linear in the depth of n.

Lemma 3. *Let $Intv_w(n) = [i,j]$.*

- *Started at position i, $\mathscr{C}^+(n,A)$ accepts iff $\exists k.\ rwithin(n,k)$ and $w[k] \in A$ and $w[i:k) \notin A$.*
- *Started at position i, $\mathscr{C}^+(n,\xrightarrow{1})$ accepts iff $i+1 \leq j$.*

Symmetric properties hold for $\mathscr{C}^-$.

Proof. The context lemma (Lemma 2) proved the required "within" property. That the automata check this "within" is easy to see. □

4 From formulae to automata

Now we are all set to construct a *po2dfa* $\mathscr{M}(D)$ which precisely accepts the word models of a given formula D. Our turtle expressions are a convenient syntax for the two-way movement of *po2dfa*. For example, expression $\xrightarrow{\lhd};\xleftarrow{a};\xrightarrow{d}$ denotes an automaton which first finds the endpoint of the word (by looking for endmarker $\lhd$) it then reverses its direction and searches for the first a in backward direction and then searches in the forward direction for the first subsequent d. Clearly, such an automaton locates the right endpoint of the interval of the subformula $D_1 = \neg(\top F_b \top)$ of D in Example 1. In general, for each subformula D_1 we can construct automata $\mathscr{L}(D_1)$ and $\mathscr{R}(D_1)$ which locate the left and right endpoints of the unique interval associated with D_1. Now it remains to check that the subword of this interval satisfies the subformula D_1. D_1 evaluates to true iff there is no (first) occurrence of letter b within its unique interval. While turtle expressions lack a simple way of checking a property within a specific subinterval,Lemma 3 shows how we can use "handles" to code this checking. Putting all this together, we give a construction of a language equivalent *po2dfa* of size d^2 for a formula of size d.

Definition 7. By induction on depth of a node n, define automata $\mathscr{L}(n)$ and $\mathscr{R}(n)$.

- $\mathscr{L}(root) = \xleftarrow{\rhd};\xrightarrow{1}$ and $\mathscr{R}(root) = \xrightarrow{\lhd};\xleftarrow{1}$.
- Let n match $\neg n_1$. Then $\mathscr{L}(n_1) = \mathscr{L}(n)$ and $\mathscr{R}(n_1) = \mathscr{R}(n)$.
- Let n match $n_1 \vee n_2$. Then $\mathscr{L}(n_1) = \mathscr{L}(n_2) = \mathscr{L}(n)$ and $\mathscr{R}(n_1) = \mathscr{R}(n_2) = \mathscr{R}(n)$.
- Let n match $n_1 F_a n_2$. Then $\mathscr{L}(n_1) = \mathscr{L}(n)$ and $\mathscr{R}(n_2) = \mathscr{R}(n)$.
  Also, $\mathscr{R}(n_1) = \mathscr{L}(n);\xrightarrow{a}$ and $\mathscr{L}(n_2) = \mathscr{R}(n_1)$.
- Let n match $n_1 L_a n_2$. Then $\mathscr{L}(n_1) = \mathscr{L}(n)$ and $\mathscr{R}(n_2) = \mathscr{R}(n)$.
  Also, $\mathscr{R}(n_1) = \mathscr{R}(n);\xleftarrow{a}$ and $\mathscr{L}(n_2) = \mathscr{R}(n_1)$.

- Let n match $\oplus n_1$. Then, $\mathscr{L}(n_1) = \mathscr{L}(n);\xrightarrow{1}$ and $\mathscr{R}(n_1) = \mathscr{R}(n)$.
- Let n match $\ominus n_1$. Then, $\mathscr{L}(n_1) = \mathscr{L}(n)$ and $\mathscr{R}(n_1) = \mathscr{R}(n);\xleftarrow{1}$.

Lemma 4. *As the inductive automaton construction follows the inductive definition of $Intv_w(n)$, it is immediate that for any node n with $Intv_w(n) = [i, j]$ (not $\mathbf{u}$), for any position k in w, $\mathscr{L}(n)(w,k) = (t,i)$ and $\mathscr{R}(n)(w,k) = (t, j)$. Thus, $\mathscr{L}(n)$ and $\mathscr{R}(n)$ are start-free. Note that $\mathscr{L}(n)$ and $\mathscr{R}(n)$ grow linearly with the depth of n.* □

Definition 8. We define $\mathscr{M}(n)$ for each node n by induction on the height of n.

- If n is labelled $\lceil\lceil A\rceil\rceil$ then $\mathscr{M}(n) = \mathscr{L}(n);\mathscr{C}^+(n,\overline{A})?Rej,Acc$. The translations for $\lceil\lceil A\rceil$, $\lceil A\rceil\rceil$ and $\lceil A\rceil$ are similar.
- For a Boolean expression, $\mathscr{M}(n)$ is defined by the corresponding turtle expression. E.g. if n matches $n_1 \vee n_2$ then $\mathscr{M}(n) = \mathscr{M}(n_1) \vee \mathscr{M}(n_2)$.
- Let n match $n_1 F_a n_2$. Let $\mathscr{M}(n) = \mathscr{L}(n);\mathscr{C}^+(n,a);\mathscr{M}(n_1);\mathscr{M}(n_2)$.
- Let n match $n_1 L_a n_2$. Let $\mathscr{M}(n) = \mathscr{R}(n);\mathscr{C}^-(n,a);\mathscr{M}(n_1);\mathscr{M}(n_2)$.
- Let n match $\oplus n_1$. Let $\mathscr{M}(n) = \mathscr{L}(n);\mathscr{C}^+(n,\xrightarrow{1});\mathscr{M}(n_1)$.
- Let n match $\ominus n_1$. Let $\mathscr{M}(n) = \mathscr{R}(n);\mathscr{C}^-(n,\xleftarrow{1});\mathscr{M}(n_1)$.

Example 5. For the formula $D \overset{\text{def}}{=} \top L_a((\neg(\top F_b \top))F_d \top)$ of Example 1 we give the construction of *po2dfa*. The formula is represented as syntax tree in Figure 1.

- The root n_1 matches $n_2 L_a n_3$. Hence, $\mathscr{M}(n_1) = \mathscr{R}(n_1);\mathscr{C}^-(n_1,a);\mathscr{M}(\top);\mathscr{M}(n_3)$. Also $\mathscr{L}(n_1) = \xleftarrow{\triangleright};\xrightarrow{1}$ and $\mathscr{R}(n_1) = \xrightarrow{\triangleleft};\xleftarrow{1}$ giving $\mathscr{C}^-(n_1,a) = \xleftarrow{a}$. Since $n_2 = \top$, we have $\mathscr{M}(n_2) = Acc$.
- n_3 matches $n_4 F_d n_4$ with $n_5 = \top$. Hence $\mathscr{M}(n_3) = \mathscr{L}(n_3);\mathscr{C}^+(n_3,d);\mathscr{M}(n_4);Acc$. Now, $\mathscr{L}(n_3) = \mathscr{R}(n_1);\xleftarrow{a}$ and $\mathscr{R}(n_3) = \mathscr{R}(n_1)$. Also $rHandle(n_3) = F_\triangleleft$. Hence, $\mathscr{C}^+(n_3,F_d) = \xrightarrow{d}$.
- n_4 matches $\neg n_6$. Hence $\mathscr{M}(n_4) = \mathscr{M}(n_6)?Rej,Acc$.
- $Subf(n_6) = \top F_b \top$. Hence, $\mathscr{M}(n_6) = \mathscr{L}(n_6);\mathscr{C}^+(n_6,b);Acc;Acc$. We have $rHandle(n_6) = F_d$. Hence, $\mathscr{C}^+(n_6,F_b) = (\overline{d} \xrightarrow{b});\xrightarrow{d}$ and $\mathscr{L}(n_6) = \mathscr{L}(n_3) = \xrightarrow{\triangleleft};\xleftarrow{1};\xleftarrow{a}$. □

Theorem 2. *Given a formula D, the language $L(D)$ is accepted by the po2dfa automaton $\mathscr{M}(root)$ of Definition 8 where root is the root node of parse tree of D. Moreover, $\mathscr{M}(root)$ has $O(|D|^2)$ states.*

Proof. Construct the syntax tree and let $Intv_w(n) = [i, j]$ for any node n. By induction on the height of node n, for any word w, we prove that $\mathscr{M}(n)$ accepts w iff $Val_w(n) = \mathbf{t}$. Note that $\mathscr{M}(n)$ is start-free since each $\mathscr{M}(n)$ is either Acc or it begins with $\mathscr{L}(n)$ or $\mathscr{R}(n)$, which are start-free by the previous lemma. Below are the proofs of four cases, the rest are similar.

- Let $n = \lceil\lceil A\rceil\rceil$. Let $Intv_w(n) = [i, j]$. Then,
  $Val_w(n) = \mathbf{t}$
  iff $\forall k : i \le k \le j : w[k] \in A$
  iff $\forall k : i \le k : w[k] \notin A$ implies $k \notin Intv_w(n)$

iff not $(\exists k : i \leq k : w[k] \notin A$ and $rwithin(n,k))$
iff the $\mathscr{C}^+(n,\overline{A})$ automaton rejects starting from (w,i). (by Lemma 3)
iff $\mathscr{M}(n)$ accepts w.

- Let n match a Boolean expression. The result holds since the smaller automata are start-free.
- Let n match $n_1 F_a n_2$. Let $Intv_w(n) = [i,j]$. Then,
  $Val_w(n) = \mathbf{t}$
  iff $w,[i,j] \models (Subf(n_1))F_a(Subf(n_2))$
  iff $\exists k : i \leq k \leq j$ s.t. $w[k] = a$ and $a \notin w[i:k)$ and $Val_w(n_1) = \mathbf{t}$
  and $Val_w(n_2) = \mathbf{t}$ (giving $Intv_w(n_1) = [i,k]$ and $Intv_w(n_2) = [k,j]$)
  iff $\exists k$ s.t. $rwithin(n,k)$ and $w[k] = a$ and $a \notin w[i:k)$ and
  $\mathscr{M}(n_1)$ accepts w and $\mathscr{M}(n_2)$ accepts w (by induction hypothesis)
  iff $\mathscr{C}^+(n,a)$ accepts (w,i) (by Lemma 3) and $\mathscr{M}(n_1), \mathscr{M}(n_2)$ accept w.
  iff $\mathscr{M}(n)$ accepts w
- Let n match $\oplus n_1$. Let $intv(n) = [i,j]$ Then,
  $Val_w(n) = \mathbf{t}$
  iff $i+1 \leq j$ and $w,[i+1,j] \models (Subf(n_1))$
  iff $i+1 \leq j$ and $\mathscr{M}(n_1)$ accepts w (By induction hypothesis)
  iff $rwithin(n,i+1)$ and $\mathscr{M}(n_1)$ accepts w
  iff $\mathscr{C}^+(n,\xrightarrow{1})$ accepts (w,i) and $\mathscr{M}(n_1)$ accepts w
  iff $\mathscr{M}(n)$ accepts w

The number of nodes in the syntax tree of a formula is linear in its size $|D|$. At each node, at most $O(|D|)$ states to the automaton are added before recursively translating sub-nodes. Hence, the the number of states of $\mathscr{M}(root)$ is $O(|D|^2)$. □

The complexities of membership and satisfiability problem for the logic *UITL* were analysed in Theorem 1. Here we give alternate upperbounds on these complexities which are obtained using the formula automaton construction. Note that even when automaton based procedures have higher complexities, in practice, they are very amenable to implementation.

Corollary 1. *Membership of a word w in the language a formula D is in* DTIME$(|w| \times |D|^2)$. *The satisfiability of D can be checked in* NSPACE$(|D|^2 \log |D|)$.

Proof. Since the number of states of $\mathscr{M}(root)$ in the theorem above is $O(|D|^2)$, whether a word model w satisfies D can be checked in time $O(|w| \times |D|^2)$ by simulating the *po2dfa*.

We can also check satisfiability of D by reducing the *po2dfa* of size $O(|D|^2)$ to a one-way DFA of size $O((|D|^2)^{|D|^2})$ using the standard 2DFA to 1DFA reduction. The emptiness of this one-way DFA is contained in nondeterministic $\log((|D|^2)^{|D|^2})$ space, i.e.NSPACE$(|D|^2 \log |D|)$. □

5 From automata to formulae

Fix a *po2dfa* M. We give the construction of a formula exactly specifying the language of M. Consider a progress transition e of M. For simplicity, we assume that e is not labelled by the endmarkers $\lhd$ or $\rhd$. We construct a formula $\psi(e)$ such that the following lemma holds. Its proof is by induction on the partial order.

Lemma 5. *$w \models \psi(e)$ iff there exists a partial run of M on w (starting at position 1) and ending with the e transition.* $\square$

The formula $\psi(e) = \bigvee_{i \in I(e)} \xi_i$ consists of finitely many disjoint disjuncts where each ξ_i is a *pointed simple formula*. Such a formula does not use boolean operators and has a pointer to one of its subformulas. For example, see $\psi(e_b)$ in Example 6. Such $\psi(e)$ defines a class of words with a unique factorization [12]. For convenience a pointed simple formula is represented as (T,n) with syntax tree T and pointer node n.

Fix a progress transition e with $\delta(p,c) = q$ such that the incoming progress transitions into p are $e_1, \ldots, e_k$. Also assume that $A = \{a \in \Sigma \mid \delta(p,a) = p\}$ are the letters on which the automaton loops in state p. Inductively assume that $\psi(e_i)$ has been constructed. Then, we define $\psi(e) = \vee\{\mathit{Extend}(\xi,e) \mid \xi \in \psi(e_i), 1 \leq i \leq k\}$. Here $\mathit{Extend}(\xi,e)$ extends the partial runs satisfying ξ to their extensions ending with e. We now define $\mathit{Extend}(\xi,e)$.

An execution ending with one of the e_i can be extended with finitely many steps involving letters of A and then taking the transition e. Moreover the head moves backwards iff $p \in Q_r$ (except at the last step where it moves in the direction of q). To take the direction into account, let $\mathit{Extend}(\xi,e) = \mathit{rExtend}(\xi,e)$ if $p \in Q_l$ and $\mathit{Extend}(\xi,e) = \ell\mathit{Extend}(\xi,e)$ if $p \in Q_r$. We define these below.

Let the inorder traversal of T be $n_1, n_2, \ldots, n_r$ (thus n_1 is the leftmost leaf and n_r is the rightmost leaf) and $n = n_i$. It is easy to see that nodes n_1 and n_2 in inorder traversal nodes are adjacent iff $\mathit{Intv}_w(n_1)$ and $\mathit{Intv}(n_2)$ are adjacent. Then, $\mathit{rExtend}(T,n_i) \overset{\text{def}}{=} \mathit{rChange}(T,n_i)$ if $n_i = n_r$ (last node in inorder), and $\mathit{rExtend}(T,n_i) \overset{\text{def}}{=} \mathit{rChange}(T,n_i) \vee \mathit{rExtend}(\mathit{rSkip}(T,n_i),n_{i+1})$ otherwise.

The function $\mathit{rChange}(T,n_i)$ modifies (T,n_i) by propagating a subalphabet (corresponding to the selfloop of the state). If n_i is labelled with any of the four atomic formulas $\lceil\lceil B\rceil\rceil$, or $\lceil B\rceil\rceil$, (or $\lceil\lceil B\rceil$ or $\lceil B\rceil$), and $c \in B$, then the corresponding leaf node n_i is replaced by the subtree corresponding to $\oplus(\lceil\lceil A \cap B\rceil F_a \lceil B\rceil\rceil)$ (or $\oplus(\lceil\lceil A \cap B\rceil F_a \lceil B\rceil)$, respectively). The new pointer points to the subformula to the left of F_a if $q \in Q_r$ and to the right of F_a otherwise." If $c \notin B$, then $\mathit{rChange}(T,n_i) = \bot$. If n_i is an F_c or L_c node, then the parse tree remains unchanged. However, if n_i is labelled F_b or L_b, for some $b \neq c$, then $\mathit{rChange}(T,n_i) = \bot$.

The function $\mathit{rSkip}(T,n_i)$ alters (T,n_i) as follows. If n_i is labelled with any of the atomic formulas $\lceil\lceil B\rceil\rceil$, $\lceil B\rceil\rceil$, $\lceil\lceil B\rceil$ or $\lceil B\rceil$, then this node is replaced by $\lceil A \cap B\rceil$. If n_i is a F_b or L_b node, then $\mathit{rSkip}(T,n_i) = T$ if $b \in A$ and $\mathit{rSkip}(T,n_i) = \bot$ if $b \notin A$.

The base case of finding formula $\psi(e)$ for an outgoing transition e from the initial state $s \in Q_l$ is $\psi(e) \overset{\text{def}}{=} \mathit{rExtend}((\lceil\lceil\Sigma\rceil, 1), e)$.

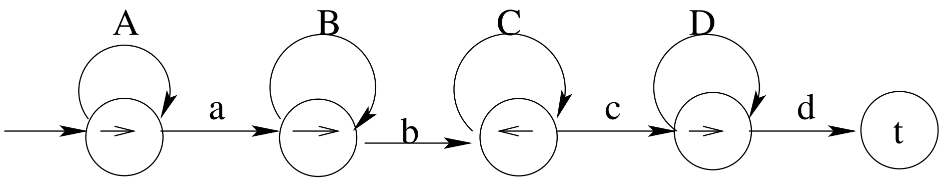

Fig. 3 A simple automaton with reversals

Example 6. Consider the automaton given in Figure 3. In this automaton, we have the conditions $a \notin A, b \notin B, c \notin C, d \notin D$, required for determinism, and we assume that $c \in B \cap A$ and $a, b \in C$. Let e_a, e_b, e_c, e_d be the edges labelled with a, b, c, d, respectively.

For convenience, a pointed simple formula T, n is denoted by underlining the subformula of node n.

- $\psi(e_a) \;=\; rExtend(\underline{\lceil\lceil\Sigma\rceil}, e_a) \;=\; \lceil\lceil A\rceil F_a \underline{\lceil\Sigma\rceil\rceil}$.
- $\psi(e_b) \;=\; rExtend(\underline{\psi(e_a)}, e_b) \;=\; \lceil\lceil A\rceil F_a(\oplus(\underline{\lceil\lceil B\rceil} F_b \lceil\Sigma\rceil\rceil))$.
- $\psi(e_c) \;=\; \ell Extend(\psi(e_b), e_c) \;=\; \ell Extend(\lceil\lceil A\rceil F_a(\oplus(\underline{\lceil\lceil B\rceil} F_b \lceil\Sigma\rceil\rceil)), e_c)$
  $= \;\; \lceil\lceil A\rceil F_a(\oplus(\ominus(\lceil\lceil B\rceil L_c \underline{\lceil B\cap C\rceil\rceil}) F_b \lceil\Sigma\rceil\rceil)) \;\; \vee \;\; \ell Extend(\lceil\lceil A\rceil \underline{F_a}(\oplus(\ominus(\lceil\lceil B\cap C\rceil) F_b \lceil\Sigma\rceil\rceil)), e_c)$
  $= \;\; \lceil\lceil A\rceil F_a(\oplus(\ominus(\lceil\lceil B\rceil L_c \underline{\lceil B\cap C\rceil\rceil}) F_b \lceil\Sigma\rceil\rceil)) \;\; \vee \;\; \ell Change(\lceil\lceil A\rceil \underline{F_a}(\oplus(\ominus(\lceil\lceil B\cap C\rceil) F_b \lceil\Sigma\rceil\rceil))) \;\vee\; \ell Extend(\underline{\lceil\lceil A\rceil} F_a(\oplus(\ominus(\lceil\lceil B\cap C\rceil) F_b \lceil\Sigma\rceil\rceil)), e_c)$
  (The above step follows from the assumption that $a \in C$)
  $= \;\; \lceil\lceil A\rceil F_a(\oplus(\ominus(\lceil\lceil B\rceil L_c \underline{\lceil B\cap C\rceil\rceil}) F_b \lceil\Sigma\rceil\rceil)) \;\; \vee \;\; false \;\; \vee \;\; \ominus(\lceil\lceil A\rceil L_c \lceil A\cap C\rceil\rceil) F_a(\oplus(\ominus(\lceil\lceil B\cap C\rceil) F_b \lceil\Sigma\rceil\rceil))$
  (The above step follows from the assumptions that $a \neq c$ and $c \in A$)

- $\psi(e_d)$ may be similarly worked out.

Theorem 3. *Given an n-pass po2dfa M, there is a formula in $R_n \cup L_n$ of size exponential in the number of its transitions, which defines the language accepted by M.*

Proof. Let E be the set of transitions leading into the accepting state t. Define the formula $F(M) \overset{\text{def}}{=} \bigvee_{e\in E} \psi(e)$. Then by the previous lemma M accepts the language $L(F(M))$. From the definition of *Extend* we see that the alternation between F_a and L_a modalities takes place only when the automaton changes direction, hence the constructed formula is in $R_n \cup L_n$.

For any transition e let $depth(e)$ denote the length of the longest progress path from the start state to e. By examining the construction, it can be seen that in $\psi(e) \;=\; \bigvee_{i\in I(e)} \xi_i$ the size of each ξ_i is linear in $depth(e)$. However, each $Extend(T, e)$ gives rise to up to $|depth(e)|$ disjuncts. Hence the number of disjuncts $|I(e)|$ can be exponential in $depth(e)$ as also the size of the $F(M)$. $\square$

This also shows that unambiguous polynomials over the piecewise testable languages [16] are matched by deterministic and co-deterministic products, a result independently obtained by Kufleitner and Weil [8]. From the main theorem of [8], we get the corollary that there is an $FO^2[<]$ formula with n quantifier alternations for the language of M above.

References

1. M. Bojańczyk, C. David, A. Muscholl, T. Schwentick and L. Segoufin. Two-variable logic on words with data, *Proc. LICS*, Seattle, 2006, 7–16.
2. M. Bojańczyk. Two-way unary temporal logic over trees, *Proc. LICS*, Wrocław, 2007.
3. B. Borchert and P. Tesson. The atnext/atprevious hierarchy on the starfree languages, *Report* WSI-2004-11 (U. Tübingen, 2004).
4. V. Diekert, P. Gastin and M. Kufleitner. A survey on small fragments of first-order logic over finite words, *Int. J. Found. Comp. Sci.*, to appear.
5. K. Etessami, M.Y. Vardi and T. Wilke. First-order logic with two variables and unary temporal logic, *Inf. Comput.* 179, 2002, 279–295.
6. S.N. Krishna and P.K. Pandya. Modal strength reduction in quantified discrete duration calculus, *Proc. FSTTCS*, Hyderabad (R. Ramanujam and S. Sen, eds.), *LNCS* 3821, 2005, 444–456.
7. M. Kufleitner. Polynomials, fragments of temporal logic and the variety DA over traces, *Theoret. Comp. Sci.* 376, 2007, 89–100.
8. M. Kufleitner and P. Weil. On language and logical hierarchies within FO^2 on words, in preparation.
9. C. Löding and W. Thomas. Alternating automata and logics over infinite words, *Proc. IFIP TCS*, Sendai (J. van Leeuwen, O. Watanabe, M. Hagiya, P.D. Mosses, T. Ito, eds.), *LNCS* 1872, 2000, 521–535.
10. M. Mortimer. On language with two variables, *Zeit. Math. Log. Grund. Math.* 21, 1975, 135–140.
11. B.C. Moszkowski and Z. Manna. Reasoning in interval temporal logic, *Proc. Logics of programs*, Pittsburgh (E.M. Clarke and D. Kozen, eds.), *LNCS* 164, 1983, 371–382.
12. M.-P. Schützenberger. Sur le produit de concaténation non ambigu, *Semigroup Forum* 13, 1976, 47–75.
13. R.L. Schwartz, P.M. Melliar-Smith and F.H. Vogt. An interval-based temporal logic, *Proc. Logics of programs*, Pittsburgh (E.M. Clarke and D. Kozen, eds.), *LNCS* 164, 1983, 443–457.
14. T. Schwentick, D. Thérien and H. Vollmer. Partially-ordered two-way automata: a new characterization of DA, *Proc. DLT '01*, Vienna (W. Kuich, G. Rozenberg and A. Salomaa, eds.), *LNCS* 2295, 2002, 239–250.
15. S.S. Shah. FO^2 and related logics, Master's thesis (TIFR, 2007).
16. I. Simon. Piecewise testable events, *Proc. GI Conf. Autom. Theory and Formal Lang.*, Kaiserslautern (H. Barkhage, ed.), *LNCS* 33, 1975, 214–222.
17. L.J. Stockmeyer and A.R. Meyer. Word problems requiring exponential time, *Proc. STOC*, Austin, 1973, 1–9.
18. P. Tesson and D. Thérien. Diamonds are forever: the variety DA, *Semigroups, algorithms, automata and languages* (G.M.S. Gomes, P.V. Silva and J.-E. Pin, eds.), (World Scientific, 2002), 475–500.
19. D. Thérien and T. Wilke. Over words, two variables are as powerful as one quantifier alternation: $FO^2 = \Sigma_2 \cap \Pi_2$, *Proc. STOC*, Dallas, 1998, 41–47.
20. P. Weis and N. Immerman. Structure theorem and strict alternation hierarchy for FO^2 on words, *Proc. CSL*, Lausanne (J. Duparc and T. Henzinger, eds.), *LNCS* 4646, 2007, 343–357.