POTL: A First-Order Complete Temporal Logic for Operator Precedence Languages

Michele Chiari¹, Dino Mandrioli¹, and Matteo Pradella¹,²

¹ Politecnico di Milano, P.zza L. Da Vinci, 32, 20133 Milano, Italy
{name.surname}@polimi.it
² IEIIT, Consiglio Nazionale delle Ricerche

Abstract. The problem of model checking procedural programs has fostered much research towards the definition of temporal logics for reasoning on context-free structures. The most notable of such results are temporal logics on Nested Words, such as CaReT and NWTL. Recently, we introduced OPTL, based on the class of Operator Precedence Languages (OPL), more powerful than Nested Words. In this paper, we introduce the new OPL-based logic POTL, prove its FO-completeness over finite words, and provide a model checking procedure for it. POTL improves on NWTL by enabling the formulation of requirements involving pre/post-conditions, stack inspection, and others in the presence of exception-like constructs. It improves on OPTL by being FO-complete, and by expressing more easily stack inspection and function-local properties.

Keywords: Linear Temporal Logic, Operator-Precedence Languages, Model Checking, First-Order Completeness, Visibly Pushdown Languages, Input-Driven Languages

1 Introduction

Model checking is one of the most successful techniques for the verification of software programs. It consists in the exhaustive verification of the mathematical model of a program against a specification of its desired behavior. The kind of properties that can be proved in this way depends both on the formalism employed to model the program, and on the one used to express the specification. The initial and most classical frameworks consist in the use of Finite State Automata (generally Büchi automata) for the model, and temporal logics such as LTL, CTL and CTL* for the specification. The success of such logics is due to their ease in reasoning about linear or branching sequences of events over time, by expressing liveness and safety properties, their conciseness with respect to automata, and the complexity of their model checking.

From the point of view of formal language theory, these kinds of temporal logics limit their set of expressible properties to the First-Order Logic (FOL) definable fragment of regular languages. This is quite restrictive when compared with the most popular abstract models of procedural programs, such as Pushdown Systems, Boolean Programs [8], and Recursive State Machines [6]. All such stack-based formalisms show behaviors which are expressible by means of context-free languages, rather than regular. To expand the expressive power of specification languages in the same direction, temporal logics
based on Visibly Pushdown Languages (VPL)\cite{4} –a.k.a. Input-Driven Languages\cite{28}– were introduced. Such logics, namely CaRet\cite{7} and the FO-complete NWTL, model the execution trace of a procedural program as a Nested Word\cite{5}, consisting in a linear ordering augmented with a one-to-one matching relation between function calls and returns. This enables requirement specifications to include Hoare-style pre/post-conditions, stack-inspection properties, and more.

VPL too have their limitations. In particular, they are a language class slightly more general than Parenthesis Languages\cite{27}, which constrains the matching relation they introduce to be exclusively one-to-one. This hinders their suitability to model processes in which a single event must be put in relation with multiple ones. Unfortunately, computer programs often present such behaviors: exceptions, continuations, and context-switches in real-time operating systems are single events that cause the termination (or re-instantiation) of multiple functions on the stack. To be able to reason about such behaviors, we propose the development of temporal logics based on Operator Precedence Languages (OPL). OPL were initially introduced with the purpose of efficient parsing\cite{18}, a field in which they continue to offer useful applications\cite{9}. They are capable of expressing arithmetic expressions, and other constructs whose context-free structure is not immediately visible. Indeed, the generality of the structure of their abstract syntax trees is much greater than that of VPL, which are strictly included in OPL\cite{15}. Nevertheless, they retain the same closure properties that make regular languages and VPL suitable for automata-theoretic model checking: OPL are closed under Boolean operations, concatenation, Kleene *, and language emptiness and inclusion are decidable\cite{23}. Moreover, they have been characterized by means of Monadic Second Order Logic.

The first step in this direction was the introduction of OPTL\cite{14}. This linear-time temporal logic, for which a model checking procedure has been given on both finite and $\omega$-words, enables reasoning on procedural programs with exceptions, expressing properties regarding the possibility of a function to be terminated by and exception, or to throw one, under certain conditions, or such as pre/post-conditions, which in the presence of exceptions can be seen as Exception Safety\cite{11}, and also function-local properties. In general, since NWTL can be translated into OPTL linearly, the latter is capable of expressing all properties of NWTL, and many more.

One of the characterizing features of linear-time temporal logics is their equivalence to FOL on their respective algebraic structure. This was the reason for introducing NWTL, since it was not possible to deduce the position of CaRet in this respect\cite{2}. This is also our motivation for presenting Precedence Oriented Temporal Logic (POTL). POTL redefines the semantics of OPTL to be much closer to the “essence” of OPL, i.e. precedence relations. In this paper, we prove the FO-completeness of POTL over finite Operator Precedence (OP) words, and we conjecture the extensibility of this proof to $\omega$-OP words by means of compositional arguments. The greater theoretical expressive power benefits POTL also in practice: with it, it is easier to express stack inspection properties in the presence of uncaught exceptions, as well as certain kinds of function-frame local properties. We conjecture some of such properties are not expressible at all in OPTL, although proving the “strict containment” of OPTL in POTL seems to be arduous, as was that of CaRet in NWTL. Nevertheless, the FO-completeness of POTL and the expressibility of OPTL in FOL allow us to conclude that POTL is at least as
expressive as OPTL. We also give a tableaux-construction procedure for model checking POTL, which yields automata of size at most singly exponential in formula length, and is thus not asymptotically greater that that of LTL and NWTL.

**Related Work.** Model checking of regular properties against context-free program models has been thoroughly studied [19,16,21,17,6,11]. After some early attempts at achieving higher expressive power [10,22], a breakthrough came with the introduction of VPL [4] and Nested Words [5], and the logics based on them: CaRet [7], NWTL and others [2]. A $\mu$-calculus based on VPL extends model checking to branching-time semantics in [3], while [13] introduces a temporal logic capturing the whole class of VPL. Timed extensions of CaRet are given in [12].

**Organization.** Background on OPL is given in Section 2. In Section 3 we present the syntax and semantics of POTL, and show its use for expressing program specifications by means of examples. In Section 4 we prove the FO-completeness of POTL, and we describe its model checking procedure in Section 5. In Section 6 we conclude by delineating research steps subsequent to this work. Most technical details and proofs are postponed to several appropriately referred appendices.

## 2 Operator Precedence Languages

Operator Precedence Languages (OPL) are usually defined through their generating grammars [18]; in this paper, however, we characterize them through their accepting automata [23] which are the natural way to state equivalence properties with logic characterization. We assume some familiarity with classical language theory concepts such as context-free grammar, parsing, shift-reduce algorithm, syntax tree [20].

Let $\Sigma = \{a_1, \ldots, a_n\}$ be an alphabet. The empty string is denoted $\varepsilon$. We use a special symbol # not in $\Sigma$ to mark the beginning and the end of any string. An **operator precedence matrix** (OPM) $M$ over an alphabet $\Sigma$ is a partial function $(\Sigma \cup \{#\})^2 \rightarrow \{\prec, \preceq, \succ\}$, that, for each ordered pair $(a, b)$, defines the **precedence relation** $(PR) M_{a,b}$ holding between $a$ and $b$; if the function is total we say that $M$ is **complete**. We call the pair $(\Sigma, M)$ an operator precedence alphabet. Relations $\prec, \preceq, \succ$ are respectively named **yields precedence, equal in precedence, and takes precedence**. By convention, the initial # can only yield precedence, and other symbols can only take precedence on the ending #. If $M_{a,b} = \pi$, where $\pi \in \{\prec, \preceq, \succ\}$, we write $a \pi b$. For $u, v \in \Sigma^+$ we write $u \pi v$ if $u = xa$ and $v = by$ with $a \pi b$.

**Definition 1.** An operator precedence automaton (OPA) is a tuple $A = (\Sigma, M, Q, I, F, \delta)$ where: $(\Sigma, M)$ is an operator precedence alphabet, $Q$ is a finite set of states (disjoint from $\Sigma$), $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, $\delta \subseteq Q \times (\Sigma \cup Q) \times Q$ is the transition relation, which is the union of three disjoint relations:

$$
\delta_{\text{shift}} \subseteq Q \times \Sigma \times Q, \quad \delta_{\text{push}} \subseteq Q \times \Sigma \times Q, \quad \delta_{\text{pop}} \subseteq Q \times Q \times Q.
$$

An **OPA** is deterministic iff $I$ is a singleton, and all three components of $\delta$ are possibly partial functions.

To define the semantics of the automaton, we need some new notations. We use letters $p, q, p_i, q_i, \ldots$ to denote states in $Q$. We will sometimes use $q_0 \xrightarrow{a} q_1$ for
changing its input symbol only to establish the pair of states consisting of the current state of the automaton and the symbol on top of the stack, and update the state of the automaton according to the transition relations.

\[ (q_0, a, q_1) \in \delta_{\text{shift}} \cup \delta_{\text{push}}, q_0 \xrightarrow{d_i} q_1 \text{ for } (q_0, q_2, q_1) \in \delta_{\text{pop}}, \text{ and } q_0 \xrightarrow{w} q_1, \text{ if the automaton can read } w \in \Sigma^* \text{ going from } q_0 \text{ to } q_1. \]

Let \( \Gamma \) be \( \Sigma \times Q \) and let \( \Gamma' = \Gamma \cup \{ \bot \} \) be the stack alphabet; we denote symbols in \( \Gamma' \) as \([a, q]\) or \(\bot\). We set \(smb([a, q]) = a\), \(smb(\bot) = \#\), and \(st([a, q]) = q\). Given a stack content \( \gamma = \gamma_n \ldots \gamma_1 \bot \), with \(\gamma_i \in \gamma\), \(n \geq 0\), we set \(smb(\gamma) = smb(\gamma_n)\) if \(n \geq 1\), \(smb(\gamma) = \#\) if \(n = 0\).

A configuration of an OPA is a triple \(c = (w, q, \gamma)\), where \(w \in \Sigma^*\#\), \(q \in Q\), and \(\gamma \in \Gamma^*\bot\). A computation or run of the automaton is a finite sequence \(c_0 \vdash c_1 \ldots \vdash c_n\) of moves or transitions \(c_i \vdash c_{i+1}\); there are three kinds of moves, depending on the PR between the symbol on top of the stack and the next symbol to read:

- **push move**: if \(smb(\gamma) < a\) then \(\langle ax, q, \gamma \rangle \vdash \langle x, q, [a, p]\gamma \rangle\), with \((p, a, q) \in \delta_{\text{push}}\);
- **shift move**: if \(a \geq b\) then \(\langle bx, q, [a, p]\gamma \rangle \vdash \langle x, r, [b, p]\gamma \rangle\), with \((q, b, r) \in \delta_{\text{shift}}\);
- **pop move**: if \(a > b\) then \(\langle bx, q, [a, p]\gamma \rangle \vdash \langle bx, r, \gamma \rangle\), with \((q, p, r) \in \delta_{\text{pop}}\).

Shift and pop moves are never performed when the stack contains only \(\bot\). Push and shift moves update the current state of the automaton according to the transition relations \(\delta_{\text{push}}\) and \(\delta_{\text{shift}}\), respectively: push moves put a new element on top of the stack consisting of the input symbol together with the current state of the automaton, whereas shift moves update the top element of the stack by changing its input symbol only. Pop moves remove the element on top of the stack, and update the state of the automaton according to \(\delta_{\text{pop}}\) on the basis of the pair of states consisting of the current state of the automaton and the state of the removed stack symbol. They do not consume the input symbol, which is used only to establish the \(\geq\) relation, remaining available for the next move. The automaton accepts the language \(L(\mathcal{A}) = \{x \in \Sigma^* \mid \langle x\#, q_l, \bot \rangle \vdash^* \langle \#, q_F, \bot \rangle, q_l \in I, q_F \in F\} \).

**Definition 2.** A simple chain is a string \(c_0c_1c_2 \ldots c_{\ell-1}\), written as \(\gamma^c_0c_1c_2 \ldots c_{\ell-1}\), such that: \(c_0, c_{\ell+1} \in \Sigma \cup \{\#\}\), \(c_i \in \Sigma\) for every \(i = 1, 2, \ldots \ell \) (\(\ell \geq 1\)), and \(c_0 < c_1 \leq c_2 \ldots c_{\ell-1} \leq c_{\ell+1}\). A composed chain is a string \(c_0s_0c_1s_1c_2 \ldots c_{\ell}s_{\ell}c_{\ell+1}\), where \(c_0[1]c_1c_2 \ldots c_{\ell}^{[\ell+1]}\) is a simple chain, and \(s_i \in \Sigma^*\) is the empty string or is such that \(c_i[s_i]^{[\ell+1]}\) is a chain (simple or composed), for every \(i = 0, 1, \ldots \ell \) (\(\ell \geq 1\)). Such a composed chain will be written as \(\gamma^c_i[s_0c_1s_1c_2 \ldots c_{\ell}s_{\ell}]^{[\ell+1]}\). The pair made of the first and the last symbols of a chain is called its context.

A finite word \(w\) over \(\Sigma\) is compatible with an OPM \(M\) iff for each pair of letters \(c, d\), consecutive in \(w\), \(M_{c,d}\) is defined and, for each substring \(x\) of \(#w\#\) which is a chain of the form \(\gamma^c_i[\gamma_i^d]\), \(M_{ab}\) is defined. E.g., the word call han call call call thr thr ret of Fig. 1 is compatible with \(M_{\text{call}}\). In the same figure all the resulting chains are reported,
e.g. \( \text{call}[\text{call}]^{\text{thr}} \), \( \text{han}[\text{thr}]^{\text{ret}} \) are simple chains, while \( \text{han}[[[[\text{call}[\text{call}]][\text{call}]][\text{thr}][\text{thr}]][\text{thr}]][\text{thr}]^{\text{ret}} \), \( \text{call}[\text{call}[\text{call}]]^{\text{thr}} \) are composed chains.

Let \( \mathcal{A} \) be an OPA. We call a support for the simple chain \( c_0[c_1 c_2 \ldots c_{\ell}]^{c_{\ell+1}} \) any path in \( \mathcal{A} \) of the form \( q_0 \xrightarrow{c_1} q_1 \xrightarrow{s_1} q_1 \xrightarrow{c_2} \ldots \xrightarrow{s_{\ell-1}} q_{\ell-1} \xrightarrow{c_\ell} q_\ell \xrightarrow{q_\ell} q_{\ell+1} \), where the arrow labeled \( c_1 \) corresponds to a push move whereas the remaining ones denote shift moves. The label of the last (and only) pop is exactly \( q_0 \), i.e. the first state of the path; this pop is executed because of relation \( c_\ell \geq c_{\ell+1} \).

We call a support for the composed chain \( c_0[s_0 c_1 s_1 c_2 \ldots c_{\ell} s_\ell]^{c_{\ell+1}} \) any path in \( \mathcal{A} \) of the form \( q_0 \xrightarrow{s_0} q'_0 \xrightarrow{c_1} q_1 \xrightarrow{s_1} q'_1 \xrightarrow{c_2} \ldots \xrightarrow{s_{\ell-1}} q_{\ell-1} \xrightarrow{s_\ell} q'_\ell \xrightarrow{q_\ell} q_{\ell+1} \), where, for every \( i = 0, 1, \ldots, \ell \); if \( s_i \neq \epsilon \), then \( q_i \xrightarrow{s_i} q'_i \) is a support for the chain \( c_i[s_i]^{c_i+1} \), else \( q'_i = q_i \).

Chains fully determine the structure of any automaton over \((\Sigma, M)\). If the automaton performs the computation \( (s, b, q_i, [a, q_j, \gamma]) \xrightarrow{a^*} (b, q_k, \gamma) \) then \( a^*[s]^b \) is necessarily a chain over \((\Sigma, M)\) and there exists a support like the one above with \( s = s_0 c_1 \ldots c_{\ell} s_\ell \) and \( q_{\ell+1} = q_k \). This corresponds to the parsing of the string \( sa_1 \ldots c_{\ell} s_\ell \) within the context \( a, b \), which contains all information needed to build the subtree whose frontier is that string.

Consider the OPA \( \mathcal{A}(\Sigma, M) = (\Sigma, M, \{q\}, \{q\}, \delta_{\text{max}}, \omega_{\text{max}}) \) where \( \delta_{\text{max}}(q, q) = q \) and \( \delta_{\text{max}}(q, c) = q, \forall c \in \Sigma \). We call it the OP Max-Automaton over \((\Sigma, M)\). For a max-automaton, each chain has a support. Since there is a chain \#[s]# for any string \( s \) compatible with \( M \), a string is accepted by \( \mathcal{A}(\Sigma, M) \) if it is compatible with \( M \). If \( M \) is complete, each string is accepted by \( \mathcal{A}(\Sigma, M) \), which defines the universal language \( \Sigma^* \) by assigning to any string the (unique) structure compatible with the OPM. With \( M_{\text{call}} \) of Fig. [1] if we take e.g. the string \( \text{ret call han} \), it is accepted by the max-automaton with structure \#([[\text{ret}][\text{call}][\text{han}]][\#].

In conclusion, given an OP alphabet, the OPM \( M \) assigns a structure to any compatible string in \( \Sigma^* \); unlike parentheses languages such a structure is not visible in the string, and must be built by means of a non-trivial parsing algorithm. An OPA defined on the OP alphabet selects an appropriate subset within the “universe” of strings compatible with \( M \). For a more complete description of the OPL family and of its relations with other CFL we refer the reader to [24].

3 POTL: Syntax and Semantics

Given a finite set of atomic propositions \( AP \), the syntax of POTL is defined as follows:

\[
\varphi \equiv a \mid \neg \varphi \mid \varphi \lor \varphi \mid \Box^\Pi \varphi \mid \Diamond^\Pi \varphi \mid \chi^\Pi_F \varphi \mid \chi^\Pi_T \varphi \mid \varphi U^\Pi \varphi \mid \varphi S^\Pi \varphi
\mid \Box^\mu_H \varphi \mid \Diamond^\mu_H \varphi \mid \varphi U^\mu_H \varphi \mid \varphi S^\mu_H \varphi
\]

where \( a \in AP, \Pi \) is a non-empty subset of \{\( \prec, \preceq, \succ \)\}, and \( \mu \in \{\langle, \rangle\} \).

We now informally illustrate its semantics by showing how it can be used to express properties on program execution traces, such as the one of Fig. [2] In the following, given two word positions \( i \) and \( j \) and a PR \( \pi \), we write \( i \pi j \) to say that the respective terminal symbols are in the \( \pi \) relation. The precedence next and back operators \( \bigcirc^\Pi \) and \( \boxdot^\Pi \) behave like their LTL counterparts, except they are true only if the current (resp. preceding)
Fig. 2. An example of execution trace, according to $M_{\text{call}}$ (Fig. 1). Chains are highlighted by arrows joining their context; structural labels are typeset in bold, while other atomic propositions are shown below them. First, procedure $p_a$ is called (pos. 1), and it installs an exception handler in pos. 2. Then, three nested procedures are called, and the innermost one ($p_d$) throws a sequence of exceptions, which are all caught by the handler. Finally, $p_a$ returns, uninstalling the handler.

position is in one of the PR in $\Pi$ with the next (resp. current) one. For example, with the OPM of Fig. 1 we can write $\circ^{\circ} \text{call}$ to say that the next position must be a call, and $\circ^\ast \text{call}$ to say that the current position is a return of an empty function frame. In Fig. 2 in pos. 2 $\circ^\ast p_b$ holds, but $\circ^\ast p_d$ does not.

The chain next and back operators $\chi_{\Pi}^n$ and $\chi_{\Pi}^b$ are similar to their precedence counterparts, except they jump between contexts of the same chain, instead of moving along the linear successor relation. $\chi_{\Pi}^n \psi$ is only true in left chain contexts, if $\psi$ holds in one of the corresponding right contexts. There can be more than one of the latter, but only those such that a relation in $\Pi$ holds among them are considered. $\chi_{\Pi}^n \psi$ is true in a position $i$ if it is the left context of a chain such that $\psi$ holds in its right context $j$, and $i$ and $j$ are in the $\preceq$ or $\succeq$ relation. By the definition of a composed chain $a[i_0, i_1, \ldots, i_n, i_{n+1}]$, this only happens if $j$ is the right context of the outermost chain starting in $i$, i.e. if $i$ is $c_i^0$ and $j$ is $c_i^{n+1}$, for $0 \leq k \leq n-1$, or $i$ is $c_i^0$ and $j$ is $b$. Conversely, $\chi_{\Pi}^b \psi$ considers the right contexts of the inner chains starting in $i$. For example, if we have $a[i_0^0 = x_0^1 c_0^1 \ldots j^0, a[i_1^0 = x_1^2 c_0^2 \ldots j^1]$, and so on, $\chi_{\Pi}^n \psi$ holds in $a$ if $\psi$ holds in either of $c_i^0, c_i^1, c_i^2$, and so on. So, in pos. 2 of Fig. 2 $\chi_{\Pi}^n p_a$ holds, meaning that the handler is uninstalled by the return statement of procedure $p_a$, and $\chi_{\Pi}^b t_1$, meaning that it catches an exception of type $t_1$. Note that if $\Pi \subseteq \{\preceq, \succeq\}$, the position considered by $\chi_{\Pi}^n$ is uniquely identified, while $\chi_{\Pi}^b$ existentially quantifies over (possibly) multiple positions.

The past operator $\chi_{\Pi}^p$ behaves symmetrically in this respect, i.e. $\prec$ takes the part of $\succeq$ in uniquely identifying a past position, the latter ranging over multiple ones.

The summary until $\psi \, U^{\Pi} \theta$ (resp. since $\psi \, S^{\Pi} \theta$) operator is obtained by inductively applying the precedence and chain next (resp. back) operators. It holds in a position in which either $\theta$ holds, or $\psi$ holds together with $\circ^H (\psi \, U^{\Pi} \theta)$ (resp. $\circ^H (\psi \, S^{\Pi} \theta)$) or $\chi_{\Pi}^n (\psi \, U^{\Pi} \theta)$ (resp. $\chi_{\Pi}^n (\psi \, S^{\Pi} \theta)$). In practice, it is an until operator that considers paths that can move not only between consecutive positions, but also between positions that are the contexts of a chain, skipping the chain body between them. With the OPM of Fig. 1 this means skipping function bodies. The relations in $\Pi$ constrain the movement of such paths. With $\preceq$, they can move between positions in the same simple chain body, i.e. between $c_i^j, c_i^{j+1}$, and so on. With $\prec$, they go down in the nested chain structure, so they can move, for example, from $a$ to $c_i^0, c_i^1$, and so on. With $\succeq$, they go upwards, e.g. from $c_i^0$ to $b$. Formula $\top \, U^{\geq} \, \text{thr}$ is true in positions contained in the frame of a function that is terminated by an exception. It is true in pos. 3 of Fig. 2 because of path 3-6,
and false in pos. 2, because no path can enter the chain whose context are positions 2 and 9. Formula $\top U^\leftarrow thr$ is true in all positions whose function frame contains throw statements, but that are not directly terminated by one of them, such as the one in pos. 1.

A single position may be the left or right context of multiple chains. The operators seen so far cannot keep this fact into account, since they “forget” about a left context when they jump to the right one. Thus, we introduce the **hierarchical** next and back operators. The **yield-precedence** hierarchical next (resp. back), $\circ_{\text{H}}^\right$ (resp. $\Theta_{\text{H}}^\right$), is true iff the current position $j$ is the right context of a chain whose left context is $i$, and $\psi$ holds in the next (resp. previous) pos. $j'$ that is the right context of $i$, with $i < j$, $j'$. Thus, if $a$ is in pos. $i$, $j$ and $j'$ can be any $c_{1}^{0}, c_{0}^{-1}$. So, $\circ_{\text{H}}^\right$ holds in pos. 6 of Fig. 2 and $\Theta_{\text{H}}^\right$ t1 in 7. The hierarchical until and since operators are defined by iterating these next and back operators. Their **take-precedence** counterparts behave symmetrically, i.e. considering multiple positions that are the left context of chains sharing their right context.

**POTL**'s formal semantics is based on the word structure $\langle U, M_{\text{P}(\text{AP})}, P \rangle$, where

- $U = \{0, 1, \ldots, n, n + 1\}$, with $n \in \mathbb{N}$ is a set of word positions;
- $M_{\text{P}(\text{AP})}$ is an operator precedence matrix on $\mathcal{P}(\text{AP})$;
- $P : U \rightarrow \mathcal{P}(\text{AP})$ is a function associating each word position in $U$ with the set of atomic propositions that hold in that position, with $P(0) = P(n + 1) = \{\#\}$.

For convenience, we consider a partitioning of AP into a set of normal proposition labels (in round font), and **structural labels** (SL, in italics). The latter define the OP structure of the word: $M_{\text{P}(\text{AP})}$ is only defined for subsets of AP containing exactly one SL, so that given two structural labels $l_1, l_2$, for any $a, a', b, b' \in \mathcal{P}(\text{AP})$ s.t. $l_1 \in a, a'$ and $l_2 \in b, b'$ we have $M_{\text{P}(\text{AP})}(a, b) = M_{\text{P}(\text{AP})}(a', b')$. This way, it is possible to define an OPM on the entire $\mathcal{P}(\text{AP})$ by only giving the relations between SL, as we did for $M_{\text{call}}$.

We define the chain relation $\chi \subseteq U \times U$ so that $\chi(i, j)$ holds between two positions $i < j - 1$ if $i$ and $j$ form the context of a chain. In case of composed chains, this relation is not one-to-one: there may be positions where multiple chains start or end. Given $i, j \in U$, the chain relation has the following properties:

1. It never crosses itself: if $\chi(i, j)$ and $\chi(h, k)$, for any $h, k \in U$, then we have $i < h < j \implies k \leq j$ and $i < k < j \implies i \leq h$.
2. If $\chi(i, j)$, then $i < i + 1$ and $j - 1 > j$.
3. There exists at most one single position $h$ s.t. $\chi(h, j)$ and $h < j$ or $h \equiv j$; for any $k$ s.t. $\chi(k, j) \text{ and } k > j$ we have $h > h$.
4. There exists at most one single position $h$ s.t. $\chi(i, h)$ and $i > h$ or $i \equiv h$; for any $k$ s.t. $\chi(i, k) \text{ and } i < k$ we have $k < h$.

The truth of POTL formulas is defined with respect to a single word position. Let $w$ be an OP word, and $a \in AP$. Then, for any position $i \in U$ of $w$, we have $\langle w, i \rangle \models a$ if $a \in P(i)$. Operators such as $\land$ and $\neg$ have the usual semantics from propositional logic.

For the **precedence** next and back operators, we have $\langle w, i \rangle \models \circ_{\Pi}^\right \phi$ iff $\pi (i + 1)$ for some $\pi \in \Pi$, and $\langle w, i + 1 \rangle \models \phi$. $\langle w, i \rangle \models \Theta_{\Pi}^\right \phi$ iff $(i - 1) \pi i$ for some $\pi \in \Pi$, and $\langle w, i - 1 \rangle \models \phi$.

For the **chain** next and back operators, we have $\langle w, i \rangle \models \chi_{\Pi}^\right \phi$ iff there exists a position $j > i$ such that $\chi(i, j)$, $i \pi j$ for some $\pi \in \Pi$, and $\langle w, j \rangle \models \phi$. $\langle w, i \rangle \models \chi_{\Pi}^\left \phi$
iff there exists a position \( j < i \) such that \( \chi(j, i), j \sim i \) for some \( \pi \in \Pi \), and \( (w, j) \models \varphi \).

In Fig. 2, \( \chi_P^{\text{ret}} \) is true in call positions whose procedure is terminated by an exception shown by an inner procedure (e.g. pos. 3 and 4). \( \chi_P^{\text{call}} \) is true in thr statements that terminate at least one procedure, such as the one in pos. 6. \( \chi_P^{\text{ret}} \) is true in call positions whose procedure terminates normally, and not because of an uncaught exception.

The hierarchical next\( /\) back operators \( \ominus_H^\mu \) and \( \ominus_H^\mu \), with \( \mu \in \{\prec, \succ\} \), move back and forth between positions in the chain relation with the same one. They are defined as:

\[
- (w, i) \models \ominus_H^\mu \varphi \text{ iff there exist a position } h < i \text{ s.t. } \chi(h, i) \text{ and } h < i \text{ and a position } j = \min\{k \mid i < k \land \chi(h, k) \land h < k\} \text{ and } (w, j) \models \varphi;
- (w, i) \models \ominus_H^\mu \varphi \text{ iff there exist a position } h < i \text{ s.t. } \chi(h, i) \text{ and } h < i \text{ and a position } j = \max\{k \mid i < k \land \chi(h, k) \land h < k\} \text{ and } (w, j) \models \varphi;
- (w, i) \models \ominus_H^\mu \varphi \text{ iff there exist a position } h > i \text{ s.t. } \chi(i, h) \text{ and } i > h \text{ and a position } j = \min\{k \mid i < k \land \chi(h, k) \land k > h\} \text{ and } (w, j) \models \varphi;
- (w, i) \models \ominus_H^\mu \varphi \text{ iff there exist a position } h > i \text{ s.t. } \chi(i, h) \text{ and } i > h \text{ and a position } j = \max\{k \mid i < k \land \chi(h, k) \land k > h\} \text{ and } (w, j) \models \varphi.
\]

For example, in pos. 3 \( \ominus_H^\prec p_c \) holds, because both pos. 3 and 4 are in the chain relation with 6. Similarly, in pos. 4 \( \ominus_H^\prec p_c \) holds. In pos. 7, both \( \ominus_H^\succ t_1 \) and \( \ominus_H^\succ t_3 \) hold.

POTL has different kinds of until and since operators. They express properties on paths, which are sequences of positions obtained by iterating the different kinds of next and back operators. In general, a path of length \( n \in \mathbb{N} \) between \( i,j \in U \) is a sequence of positions \( i = i_1 < i_2 < \cdots < i_n = j \). The until operator on a set of paths \( \Gamma \) is defined as follows: for any word \( w \) and position \( i \in U \), and for any two POTL formulas \( \varphi \) and \( \psi \), \( (w, i) \models \varphi \mathbf{U}_w^\Gamma \psi \) iff there exist a position \( j \in U \), \( j \geq i \), and a path \( i_1 < i_2 < \cdots < i_n \) between \( i \) and \( j \) in \( \Gamma \) such that \( (w, i_k) \models \varphi \) for any \( 1 \leq k < n \), and \( (w, i_n) \models \psi \). The since operator is defined symmetrically. Note that, depending on the set \( \Gamma \), a path from \( i \) to \( j \) may not exist. We define all kinds of until\( /\) since operators by associating them with different sets of paths.

**Summary Operators.** We define Operator Precedence Summary Paths (OPSP) as follows. Given an OP word \( w \), a non-empty set \( \Pi \subseteq \{\prec, \sim, \succ\} \), and two positions \( i \leq j \) in \( w \), the OPSP between \( i \) and \( j \), if it exists, is a sequence of positions \( i = i_1 < i_2 < \cdots < i_n = j \) such that, for each \( 1 \leq p < n \),

\[
i_{p+1} = \begin{cases} 
  k & \text{ iff } k = \max\{h \mid h \leq j \land \chi(i_p, h) \land \bigvee_{\pi \in \Pi} i \sim k\}; \\
  i_p + 1 & \text{ iff } \bigvee_{\pi \in \Pi} i_p \pi (i_p + 1).
\end{cases}
\]

The Operator Precedence Summary (OPS) until and since operators \( \mathbf{U}_w^\Pi \) and \( \mathbf{S}_w^\Pi \) are based on the set of OPSP starting in the position in which they are evaluated. In Fig. 2 (\( \text{han } \vee \text{ thr} \) \( \mathbf{U}_w^\Pi t_2 \) holds in pos. 2 because of path 2-7, thr \( \mathbf{S}_w^\Pi p_c \) holds in pos. 7 because of path 4-6-7. Moreover, we define two kinds of hierarchical paths.

**Yield-Precedence.** The yield-precedence hierarchical path (YPHP) between \( i \) and \( j \) is a sequence of positions \( i = i_1 < i_2 < \cdots < i_n = j \) such that there exists a position \( h < i \) such that for each \( 1 \leq p \leq n \) we have \( \chi(h, i_{p'}) \) and \( h < i_p \), and for each \( i < q < n \) there exists no position \( k \) such that \( i_q < k < i_{q+1} \) and \( \chi(h, k) \). The until and since operators based on the set of yield-precedence hierarchical paths starting in the position in which
they are evaluated are denoted as $\mathcal{U}_H^i$ and $\mathcal{S}_H^i$. E.g., $\text{thr}\, \mathcal{U}_H^i\, t_3$ holds in pos. 6 because of path 6-7-8. Similarly, $\text{thr}\, \mathcal{S}_H^i\, t_1$ holds in pos. 8 and is witnessed by the same path.

**Take-Precendence.** The take-precedence hierarchical path (TPHP) between $i$ and $j$ is a sequence of positions $i = i_1 < i_2 < \cdots < i_n = j$ such that there exists a position $h > j$ such that for each $1 \leq p \leq n$ we have $\chi(i_p, h)$ and $i_p \triangleright h$, and for each $1 \leq q < n$ there exists no position $k$ such that $i_q < k < i_{q+1}$ and $k \triangleright h$. The until and since operators based on the set of take-precedence hierarchical paths starting in the position in which they are evaluated are denoted as $\mathcal{U}_H^i$ and $\mathcal{S}_H^i$. In Fig. 2, $\text{call}\, \mathcal{U}_H^i\, p_c$ holds in pos. 3, and $\text{call}\, \mathcal{S}_H^i\, p_h$ in pos. 4, both because of path 3-4.

The above operators enjoy the following expansion laws, proved in Appendix [A].

\[
\varphi \mathcal{U}^i \psi \equiv \psi \lor \left( \varphi \land \left( \mathcal{H} \mathcal{U}^i \psi \right) \right)
\]
\[
\varphi \mathcal{S}^i \psi \equiv \psi \lor \left( \varphi \land \left( \mathcal{H} \mathcal{S}^i \psi \right) \right)
\]
\[
\varphi \mathcal{U}_H^i \psi \equiv \left( \varphi \land \chi^\varphi \right) \lor \left( \varphi \land \mathcal{H} \mathcal{U}_H^i \psi \right)
\]
\[
\varphi \mathcal{S}_H^i \psi \equiv \left( \varphi \land \chi^\varphi \right) \lor \left( \varphi \land \mathcal{H} \mathcal{S}_H^i \psi \right)
\]

POTL can express all stack-trace related properties of OPTL. As we mentioned in the introduction, and formalize in Corollary [2] and Appendix [C.3], OPTL $\subseteq$ POTL. E.g., $\square [\text{han} \Rightarrow \chi^\varphi_h \text{ret}]$, where $\square \psi$ is a shortcut for $\neg (\tau \mathcal{U}^{\varphi=\tau} \neg \psi)$, holds if all exception handlers are properly uninstalled by a return statement. Formula $\square [\text{thr} \Rightarrow \neg \chi^\varphi_h \text{p}_b]$ is false if $\text{p}_b$ is terminated by an exception. Formula $\neg \chi^\varphi_h (\text{thr} \land \chi^\varphi_h \text{call})$ is true in handles that do not catch any thr statement that terminates a procedure. We give a few more examples:

\[
\square[\text{call} \land p_A \Rightarrow \neg \chi^\varphi_h \text{thr} \land \neg \chi^\varphi_h \text{thr}]
\]
\[
\square[\text{call} \land p \land (\neg \chi^\varphi_h \text{thr} \lor \chi^\varphi_h \text{thr}) \Rightarrow \neg \chi^\varphi_h \text{thr} \lor \chi^\varphi_h \text{thr}]
\]
\[
\square[\text{call} \land p_A \land (\neg \text{ret} \mathcal{U}_C = \text{ret}) \Rightarrow \chi^\varphi_h \text{thr}]
\]
\[
\square[\{ p_C \land (\neg \chi^\varphi_h \text{thr}) \Rightarrow \neg p_A \mathcal{S}_H^i \text{p}_b \}]
\]
\[
\square[\{ \text{call} \land p_B \land (\neg \mathcal{S}_H^i (p_A \land \text{call}) \Rightarrow \mathcal{U}_C (\neg \chi^\varphi_h \text{thr}) \}]
\]

(7) means procedure $p_A$ is never interrupted by an uncaught exception. (8) means that if precondition $\rho$ holds when a procedure is called, then postcondition $\theta$ must hold if that procedure is terminated by an exception. If $\theta$ is a class invariant asserting that a class instance is in a valid state, this formula expresses weak exception safety [1]. If $\rho_0$ is a formula expressing a particular state of the class instance, this formula expresses strong exception safety. (9) means if a procedure $p_A$ or its subprocedures write to a variable $x$, they are terminated by an exception. It can be used to check enforcement of data access permissions. (10) means that whenever the stack frame of a procedure $p_C$ is unwound by an uncaught exception, $p_B$ is present in the backtrace, and it is also unwound. Moreover, the frame of $p_A$ cannot be unwound before the one of $p_B$. Unlike previous properties, we conjecture this one cannot be expressed in OPTL. (11) means
whenever \( p_B \) is executed, at least one instance of \( p_A \) must be on stack, otherwise \( p_B \) or a subfunction throw an exception.

4 First-Order Completeness

To obtain FO-completeness for POTL, we give a translation of Conditional XPath (CXPath) \([25]\), a logic on trees, into POTL on OP words. A translation into FOL is given in Appendix B. From CXPath being equivalent to FOL on trees \([26]\), we derive a FO-completeness result for POTL. To do so, we give an isomorphism between OP words and (a subset of) unranked ordered trees, the algebraic structures on which CXPath is based. First, we show how to translate OP words into trees, and then the reverse.

An unranked ordered tree is a tuple \( T = \langle S, R_\downarrow, R_\Rightarrow, L \rangle \). Each node is a sequence of child numbers, representing the path from the root to it. \( S \) is a finite set of finite sequences of natural numbers closed under the prefix operation, and for any sequence \( s \in S \), if \( s \cdot k \in S \), \( k \in \mathbb{N} \), then either \( k = 0 \) or \( s \cdot (k-1) \in S \) (by \( \cdot \) we denote concatenation). \( R_\downarrow \) and \( R_\Rightarrow \) are two binary relations called the descendant and following sibling relation, respectively. For \( s, t \in S \), \( sR_\downarrow t \) iff \( t \) is any child of \( s \) \((t = s \cdot k, k \in \mathbb{N}, \text{i.e. } t \text{ is the } k\text{-th child of } s \)\), and \( sR_\Rightarrow t \) iff \( t \) is the immediate sibling to the right of \( s \) \((s = r \cdot h \text{ and } t = r \cdot (h+1))\), for \( r \in S \) and \( h \in \mathbb{N} \). Finally, \( L : S \rightarrow \mathcal{P}(AP) \) is a function that maps each node to its label, i.e. the set of atomic propositions holding in it. We denote as \( \mathcal{T} \) the set of all unranked ordered trees.

Given an OP word \( w = \langle U, M_{P(A P)}, P \rangle \), it is possible to build an unranked ordered tree \( T_w = \langle S_w, R_\downarrow, R_\Rightarrow, L_w \rangle \in \mathcal{T} \) with labels in \( \mathcal{P}(AP) \) isomorphic to \( w \). To do so, we define a translation function \( \tau : U \rightarrow S_w \), which maps positions of \( w \) into nodes of \( T_w \).
- \( \tau(0) = 0 \): position 0 is the root node.
- Given any position \( i \in U \), if \( i \equiv i + 1 \), then \( \tau(i + 1) = \tau(i) \cdot 0 \) is the only child of \( i \).
- If \( i > i + 1 \), then \( i \) has no children.
- If \( i < i + 1 \), then the leftmost child of \( i \) is \( i + 1 \) \( (\tau(i + 1) = \tau(i) \cdot 0) \).
- If \( j_1 < j_2 < \cdots < j_n \) is the largest set of positions such that \( \chi(i, j_k) \) and either \( i < j_k \) or \( i = j_k \) for \( 1 \leq k \leq n \), then \( \tau(j_k) = \tau(i) \cdot k \).

In general, \( i \) is in the \( < \) relation with all of its children, except possibly the rightmost one, with which \( i \) may be in the \( = \) relation. Note that this way, every position \( i \) in \( w \) appears in the tree exactly once. Indeed, if the position preceding \( i \) is in the \( = \) or \( < \) relation with it, then \( i \) is one of its children. If \( (i - 1) > i \), then at least a chain ends in \( i \). In particular, consider position \( j \) s.t. \( \chi(j, i) \), and for no \( j' < j \) we have \( \chi(j', i) \); necessarily either \( j \not\equiv i \) or \( j < i \), or \( i \) would be the right context of another chain whose body contains \( j \).

So, \( i \) is a child of \( j \). Finally, \( L_w(\tau(i)) = P(i) \), so each node in \( T_w \) is labeled with the set of atomic propositions that hold in the corresponding word position. We denote as \( T_v = \tau(w) \) the tree obtained by applying \( \tau \) to every position of an OP word \( w \). Fig. 3 shows the translation of the word of Fig. 2 into an unranked ordered tree.

As for the other way of the isomorphism, notice that we are considering only a subset of unranked ordered trees. In fact, we only consider trees whose node labels are compatible with a given OPM \( M_{P(AP)} \). In order to define the notion of OPM compatibility for trees, we need to introduce the right context candidate (Rcc) of a node. Given a tree \( T \) and a node \( s \in T \), the Rcc of \( s \) is denoted Rcc(\( s \)). If \( r \) is the leftmost right sibling of \( s \), then Rcc(\( s \)) = 0. Otherwise, Rcc(\( s \)) = Rcc(\( p \)), where \( p \) is the parent of \( s \).

We denote the set of trees compatible with an OPM \( M \) as \( T_M \). A tree \( T \) is in \( T_M \) iff the following properties hold. The root node is labeled with \( \# \), and it has at most two children, the rightmost one being labeled with \( \# \). No other node is labeled with \( \# \). In the following we write \( s \equiv s' \) meaning that \( L(s) \equiv L(s') \), for any \( s, s' \in S \) and \( \equiv \in \{ <, =, > \} \).

For any node \( s \in T \), let \( r \in T \) be the rightmost child of \( s \). Then either \( s < r \) or \( s > r \). For any child \( s' \in T \) s.t. \( s' \) is a (left) sibling of \( r \), we have \( s < s' \). If \( s \) has no child \( s' \) such that \( s' > s \), then \( s > Rcc(s) \), if the latter exists. Note that Rcc(\( s \)) always exists for all nodes not labeled with \( \# \), because the root always has a second child.

Given a tree \( T \in T_M \) with labels on \( P(AP) \), it is possible to build an OP word \( w_T \) isomorphic to \( T \). We define function \( \tau^{−1}_{AP} : S \rightarrow P(AP)^∗ \), which maps a tree node to the substring corresponding to the subtree rooted in it. For any node \( s \in T \), let \( a = L(s) \) be its label, and let \( c_0, c_1 \ldots c_n \) be its children, if any. Then \( \tau^{−1}_{AP}(s) \) is defined as \( \tau^{−1}_{AP}(s) = a \) if \( s \) has no children, and \( \tau^{−1}_{AP}(s) = a \cdot \tau^{−1}_{AP}(c_0) \cdot \tau^{−1}_{AP}(c_1) \cdot \tau^{−1}_{AP}(c_n) \) otherwise.

The string obtained in this way is a valid OP word. To show this, we need to prove by induction that for any tree node \( s \), \( \tau^{−1}_{AP}(s) \) is of the form \( a_0x_0a_1x_1 \ldots a_nx_n \), with \( n \geq 0 \), and such that for \( 0 \leq k < n, a_k = a_{k+1} \) and either \( x_k = e \) or \( x_k = [a_{k+1}]^{u_{k+1}} \). In the following, we denote as first(\( x \)) the first position of a string \( x \), and as last(\( x \)) the last one. Indeed, for each \( 0 \leq i < n \) we have \( a < \text{first}(\tau^{−1}_{AP}(c_i)) \), and the rightmost leaf \( f_i \) of the tree rooted in \( c_i \) is such that Rcc(\( f_i \)) = \( c_{i+1} \), and last(\( \tau^{−1}_{AP}(c_i) \)) > first(\( \tau^{−1}_{AP}(c_{i+1}) \)). So, \( a^{[\text{first}(\tau^{−1}_{AP}(c_i))]^{\text{last}(\tau^{−1}_{AP}(c_{i+1}))}} \). As for \( \tau^{−1}_{AP}(c_n) \), if \( a < c_n \) then \( \tau^{−1}_{AP}(s) = a_0x_0(a_0 < \text{first}(x_0)) \), with \( a_0 = a \) and \( x_0 = \tau^{−1}_{AP}(c_0) \cdot \tau^{−1}_{AP}(c_1) \cdot \tau^{−1}_{AP}(c_n). \) If \( a \equiv c_n \), consider that, by hypothesis, \( \tau^{−1}_{AP}(c_n) \) is of the form \( a_1x_1a_2x_2 \ldots a_nx_n \). So \( \tau^{−1}_{AP}(s) = a_0x_0a_1x_1a_2x_2 \ldots a_nx_n, \) with \( a_0 = a \) and \( x_0 = \tau^{−1}_{AP}(c_0) \cdot \tau^{−1}_{AP}(c_1) \cdot \tau^{−1}_{AP}(c_n−1) \).
The root of $T$ is labeled with $\#$, and its first child has the form $a_0 x_0 a_1 x_1 \ldots a_n x_n$, as described above. The second child $c_b$ of the root is labeled with $\#$. Consider the tree node corresponding to first($x_n$): its rightmost leaf $f$ is such that Rcc($f$) = $c_b$, and last($x_n$) $\gg \#$.

So $#a_0 x_0 a_1 x_1 \ldots a_n x_n#$ is a finite OP word.

Function $\tau^{-1} : S \rightarrow U$ can be derived from $\tau_{AP}^{-1}$. From the existence of $\tau^{-1}$ follows

**Lemma 1.** Given an OP word $w$ and the tree $T_w = \tau(w)$, function $\tau$ is an isomorphism between positions of $w$ and nodes of $T_w$.

Consequently,

**Proposition 1.** Let $M_{P(\text{AP})}$ be an OPM on $P(\text{AP})$. For any FO formula $\varphi(x)$ on OP words compatible with $M_{P(\text{AP})}$, there exists a FO formula $\varphi'(x)$ on trees in $T_{M_{P(\text{AP})}}$, such that for any OP word $w$ and position $i$ in it, $w \models \varphi(i)$ iff $T_w \models \varphi'(\tau(i))$, with $T_w = \tau(w)$.

We now give the full translation of the logic $\mathcal{X}_{\text{un}}$ from [25] into POTL. The syntax of $\mathcal{X}_{\text{un}}$ formulas is

\[ \varphi = p \mid T \mid \neg \varphi \mid \varphi \land \varphi \mid \rho(\varphi, \psi) \]

with $a \in AP$ and $\rho \in \{\emptyset, \downarrow, \Rightarrow, \Leftarrow\}$. The semantics of propositional operators is the usual one, while $\rho(\varphi, \psi)$ is a strict until/since operator on the child and sibling relations. Let $T \in T$ be a tree with nodes in $S$. For any $r, s \in S$, let $R^t, R^s$ be s.t. $r R^t s$ iff $s R_i r$, and $r R^s s$ iff $s R_i s$. We denote as $R^t_{\rho}$ the transitive (but not reflexive) closure of relation $R^t$. For $s \in S$, $(T, s) \models \rho(\varphi, \psi)$ iff there exists a node $t \in S$ s.t. $r R^t s$ and $(T, t) \models \psi$, and for any $r \in S$ s.t. $s R^t s$ and $r R^t s$ we have $(T, r) \models \varphi$. $X_{\text{un}}$ was proved to be equivalent to FOL on finite unranked ordered trees in [26]. This result is valid for any labeling of tree nodes, and so is on OPM-compatible trees as well.

**Theorem 1.** Let $M_{P(\text{AP})}$ be an OPM on AP. For any FO formula $\varphi(x)$ on trees in $T_{M_{P(\text{AP})}}$, there exists a $\mathcal{X}_{\text{un}}$ formula $\varphi'$ such that, for any $T \in T_{M_{P(\text{AP})}}$ and node $t \in T$, we have $T \models \varphi(t)$ iff $(T, t) \models \varphi'$ [26].

We define function $\iota_X$, which translates any $\mathcal{X}_{\text{un}}$ formula $\varphi$ into a POTL formula $\varphi'$ holds on a tree $T$ iff $\iota_X(\varphi')$ holds on the isomorphic word $w_T$. For the purely propositional operators, $\iota_X$ is defined as the identity. Otherwise, $\iota_X$ is defined by means of the following equivalences.

\[
\iota_X(\emptyset, (\varphi, \psi)) \equiv \bigcirc^{H\varphi} (\varphi' \mathcal{U}^{H\psi}) \vee \bigcirc^{F\varphi} (\varphi' \mathcal{U}^{F\psi}) \\
\iota_X(\emptyset, (\varphi, \psi)) \equiv \bigcirc^{\varphi} (\varphi' S^{\psi}) \vee \bigcirc^{\psi} (\varphi' S^{\varphi'})
\]

(12)

(13)

\[
\iota_X(\Rightarrow, (\varphi, \psi)) \equiv \bigcirc^{H\varphi} (\varphi' \mathcal{U}^{H\psi}) \vee (\neg \bigcirc^{H\varphi} (\top \mathcal{U}^{H\psi}) \land \bigcirc^{H\varphi} \chi^{H\varphi'}(\varphi')) \vee (\bigcirc^{H\varphi} \chi^{H\varphi'}(\varphi') \land \neg \bigcirc^{H\varphi} (\top S^{H\psi}) \neg \varphi')
\]

(14)

(15)

(16)

(17)
employed for NWTL in [2]. The full translation can be found in Appendix B. From this, analogous. The full proofs are in Appendix C. 2. For
be the parent of
is captured by the disjunction of four cases in the corresponding word
satisfied by a path starting in the first sibling after
which paths do not start from the current node, but from its immediate children.

\[ \phi = \neg s.t. \] formula, \( \psi \), – – (15):

\[ \text{Thus, the appropriate translation. It must be preceded by } \circ \chi \text{ and } \chi \text{ due to the strict semantics of } X_{\text{unil}}, \]
in which paths do not start from the current node, but from its immediate children.

\[ \text{\Rightarrow (} \phi, \psi \text{) is an until on children of a node. It holds on a tree node } s = \tau(i) \text{ iff it is satisfied by a path starting in the first sibling after } s \text{ and ending in } t = \tau(j) \text{ s.t. } s R_{\phi}^{\circ} t. \text{ This is captured by the disjunction of four cases in the corresponding word } w. \text{ Let } r = \tau(h) \]
be the parent of \( s \), and \( p = \tau(m) \) and \( q = \tau(n) \) be the left- and rightmost children of \( r \).

- \( \text{(13): } s \neq p \text{ and } r < t. \text{ Then, all positions } j' \text{ in } w \text{ corresponding to nodes in the path are s.t. } \chi(h, j') \text{ and } h < j', \text{ so formula (13) is a valid translation.} \)
- \( \text{(15): } s \neq p, t = q \text{ and } s \neq q. \text{ In } w, \phi' \text{ must hold in all positions } j' > i \text{ s.t. } \chi(h, j') \text{ and } h < j', \text{ and this is captured by } \neg \circ \chi \text{ in } \psi \text{. Moreover, } \psi' \text{ holds in } n, \text{ the only position s.t. } \chi(h, n) \text{ and } h = n, \text{ which is equivalent to } \chi \text{ holding in } i. \)
- \( \text{(16): } s = p \text{ and } r < t. \text{ In } w, i = h + 1, \text{ so if } \circ \text{ is evaluated in } i, \text{ its argument holds in } h. \text{ The first sibling node after } s \text{ corresponds in } w \text{ to } m', \text{ the leftmost position s.t. } \chi(h, m') \text{ and } h < m'. \text{ If } n \text{ is s.t. } \chi(h, n) \text{ and } h = n, \text{ the path starting from } m' \text{ ends before } n. \text{ So, there exists a position } j' \text{ s.t. } \chi(h, j') \text{, } h < j' \text{ in which } \psi' \text{ holds, while } \phi' \text{ holds in all positions } j'' \text{ s.t. } m' \leq j'' < j', \chi(h, j'') \text{ and } h < j''. \text{ So } \chi \text{ must hold in } h. \)
- \( \text{(17): } s = p, t = q \text{ and } s \neq q. \text{ In this case, } \phi' \text{ holds in all positions } j' \text{ s.t. } \chi(h, j') \text{ and } h < j', \text{ while } \psi' \text{ holds in position } n \text{ s.t. } \chi(h, n) \text{ and } h = n. \text{ This is equivalent to } \chi \text{ holding in } h. \)

It is possible to express all POTL operators in FOL, by following the semantics described in Section 3. The translation of OPS until/since operators is similar to the one employed for NWTL in [2]. The full translation can be found in Appendix B. From this, and Lemma 2 together with Theorem 1 we derive

Theorem 2. POTL is equivalent to FO with one free variable on finite OP words.

From the results above follows that
Corollary 1. The set of operators $\circ_1, \circ_1^\Pi, \chi_F^\Pi, \chi_p^\Pi, \mathcal{U}_p^\varepsilon, \mathcal{S}_p^{\varepsilon^\pm}, \circ_1^c, \circ_1^c, \mathcal{U}_p^c, \mathcal{S}_p^c$, where $\Pi \subseteq \{\leq, \preceq\}$, is expressively complete on OP words.

The semantics of OPTL can be expressed in FOL similarly to POTL, hence

**Corollary 2.** $\text{OPTL} \subseteq \text{POTL}$ over finite OP words.

5 Model Checking

We present an automata-theoretic model checking procedure for POTL based on OPA. Its correctness is proved in Appendix D. Given an OP alphabet $(\mathcal{P}(\text{AP}), M_{\Phi}(\text{AP}))$, where $\text{AP}$ is a finite set of atomic propositions, and a formula $\varphi$, let $\mathcal{A}_\varphi = (\mathcal{P}(\text{AP}), M_{\Phi}(\text{AP}), Q, I, F, \delta)$ be an OPA. The construction of $\mathcal{A}_\varphi$ resembles the classical one for LTL and the ones for NWTL and OPTL, diverging from them significantly when dealing with temporal obligations that involve positions in the chain relation. We first introduce $\text{Cl}(\varphi)$, the closure of $\varphi$, containing all subformulas of $\varphi$, plus a few auxiliary operators.

Initially, $\text{Cl}(\varphi)$ is the smallest set such that

1. $\varphi \in \text{Cl}(\varphi)$,
2. $\text{AP} \subseteq \text{Cl}(\varphi)$,
3. if $\psi \in \text{Cl}(\varphi)$ and $\psi \neq \neg \theta$, then $\neg \psi \in \text{Cl}(\varphi)$;
4. if $\neg \psi \in \text{Cl}(\varphi)$, then $\psi \in \text{Cl}(\varphi)$;
5. if any of $\psi \land \theta$ or $\psi \lor \theta$ is in $\text{Cl}(\varphi)$, then $\psi \in \text{Cl}(\varphi)$ and $\theta \in \text{Cl}(\varphi)$;
6. if any of the unary temporal operators (such as $\circ_1^\Pi, \chi_F^\Pi, ...$) is in $\text{Cl}(\varphi)$, and $\psi$ is its argument, then $\psi \in \text{Cl}(\varphi)$;
7. if any of the until- and since-like operators is in $\text{Cl}(\varphi)$, and $\psi$ and $\theta$ are its operands, then $\psi, \theta \in \text{Cl}(\varphi)$.

The set $\text{Atoms}(\varphi)$ contains all consistent subsets of $\text{Cl}(\varphi)$, i.e. all $\Phi \subseteq \text{Cl}(\varphi)$ such that

1. for every $\psi \in \text{Cl}(\varphi)$, $\psi \in \Phi$ iff $\neg \psi \notin \Phi$;
2. if $\psi \land \theta \in \Phi$, then $\psi \in \Phi$ and $\theta \in \Phi$;
3. if $\psi \lor \theta \in \Phi$, then $\psi \in \Phi$ or $\theta \in \Phi$, or both.

The consistency constraints on $\text{Atoms}(\varphi)$ will be augmented incrementally in the following sections, for each operator.

The set of states of $\mathcal{A}_\varphi$ is $Q = \text{Atoms}(\varphi)^2$, and its elements, which we denote with Greek capital letters, are of the form $\Phi = (\Phi_c, \Phi_p)$, where $\Phi_c$ is the set of formulas that hold in the current position, and $\Phi_p$ is the set of temporal obligations. The latter keep track of arguments of temporal operators that must be satisfied after a chain body, skipping it. The way they do so depends on the transition relation $\delta$, which we also define incrementally in the next sections. Each word position is associated to an automaton state. So, for $(\Phi, a, \Psi) \in \delta_{\text{push/shift}}$, with $\Phi \in \text{Atoms}(\varphi)^2$ and $a \in \mathcal{P}(\text{AP})$, we have $\Phi_c \cap \text{AP} = a$ (by $\Phi_c \cap \text{AP}$ we mean the set of atomic propositions in $\Phi_c$). Pop moves do not read input symbols, and the automaton remains stuck at the same time instant when performing them: for any $(\Phi, \Theta, \Psi) \in \delta_{\text{pop}}$ we impose $\Phi_c \subseteq \Psi_c$, and $\Phi_c \cap \text{AP} = \Psi_c \cap \text{AP}$. The initial set $I$ only contains states of the form $(\Phi_c, \emptyset)$, with $\varphi \in \Phi_c$, and the final set $F$ only states of the form $(\Psi_c, \emptyset)$, s.t. $\Psi_c \cap \text{AP} = \{\#\}$ and $\Psi_c$ contains no future operators.
Precedence Next/Back Operators. Let $\circ \Pi \psi \in \text{Cl}(\varphi)$, with $\Pi \subseteq \{\prec, \approx, \succ\}$: then $\psi \in \text{Cl}(\varphi)$. Moreover, let $(\Phi, a, \Psi) \in \delta_{\text{shift}} \cup \delta_{\text{push}}$, with $\Phi, \Psi \in \text{Atoms}(\varphi)^2$, $a \in \mathcal{P}(AP)$, and let $b = \Psi_c \cap AP$: we have $\circ \Pi \psi \in \Phi_c$ iff $\psi \in \Psi_c$ and there exists a PR $\pi \in \Pi$ such that $a \pi b$. The constraints introduced for the $\circ \Pi$ operator are symmetric.

Chain Next/Back Operators. The semantics of $\chi_F^L \psi$, with $\Pi \subseteq \{\prec, \approx, \succ\}$, depends on the PR in $\Pi$. We describe the automaton construction for singleton values of $\Pi$ only. If $\Pi$ is not a singleton, then for each $\pi \in \Pi$ we add $\chi_F^\pi \in \text{Cl}(\varphi)$, and for each $\Phi \in \text{Atoms}(\varphi)^2$ we impose that if $\chi_F^\pi \psi \in \Phi_c$, then $\chi_F^\pi \psi \in \Phi_c$ for some $\pi' \in \Pi$.

First, we need to add into $\text{Cl}(\varphi)$ the auxiliary symbol $\chi_L$, which forces the previous position to be the left context of a chain, by imposing the current one to be the first position of a chain body. If the current state of the OPA is $\Phi \in \text{Atoms}(\varphi)^2$, and $\chi_L \in \Phi_p$, then the next transition (i.e. the one reading the current position) must be a push. This is formalized by the following rules: if $(\Phi, a, \Psi) \in \delta_{\text{shift}}$ or $(\Phi, \Theta, \Psi) \in \delta_{\text{pop}}$, for any $\Phi, \Theta, \Psi$ and $a$, then $\chi_L \notin \Phi_p$. If $(\Phi, a, \Psi) \in \delta_{\text{push}}$, we may have $\chi_L \in \Phi_c$ and possibly $\chi_L \notin \Psi$.

The satisfaction of $\chi_F^\pi \psi$ is ensured by the following constraints on $\delta$.

1. Let $(\Phi, a, \Psi) \in \delta_{\text{push}/\text{shift}}$: then $\chi_F^\pi \psi \in \Phi_c$ iff $\chi_F^\pi \psi, \chi_L \in \Psi_p$;
2. let $(\Phi, \Theta, \Psi) \in \delta_{\text{pop}}$: then $\chi_F^\pi \psi \notin \Phi_p$, and $\chi_F^\pi \psi \in \Theta_p$ iff $\chi_F^\pi \psi \in \Psi_p$;
3. let $(\Phi, a, \Psi) \in \delta_{\text{shift}}$: then $\chi_F^\pi \psi \in \Phi_p$ iff $\psi \in \Phi_c$.

The rules for $\chi_F^\pi \psi$ only differ in $\psi$ being enforced by a pop transition, triggered by the $\succ$ relation between the left and right contexts of the chain on which $\chi_F^\pi \psi$ holds.

1. Let $(\Phi, a, \Psi) \in \delta_{\text{push}/\text{shift}}$: then $\chi_F^\pi \psi \in \Phi_c$ iff $\chi_F^\pi \psi, \chi_L \in \Psi_p$;
2. let $(\Phi, \Theta, \Psi) \in \delta_{\text{pop}}$: then $\chi_F^\pi \psi \in \Theta_p$ iff $\chi_F^\pi \psi \in \Psi_p$, and $\chi_F^\pi \psi \in \Phi_p$ iff $\psi \in \Phi_c$;
3. let $(\Phi, a, \Psi) \in \delta_{\text{shift}}$: then $\chi_F^\pi \psi \notin \Phi_p$.

The constraints for $\chi_F^\pi \psi$ are the following.

1. Let $(\Phi, a, \Psi) \in \delta_{\text{push}/\text{shift}}$: then $\chi_F^\pi \psi \in \Phi_c$ iff $\chi_F^\pi \psi, \chi_L \in \Psi_p$;
2. let $(\Phi, \Theta, \Psi) \in \delta_{\text{pop}}$: then $\chi_F^\pi \psi \in \Theta_p$ iff $\chi_L \in \Psi_p$, and either (a) $\chi_F^\pi \psi \in \Psi_p$ or (b) $\psi \in \Phi_c$.

We illustrate how the construction works for $\chi_F^\pi \psi$ with the example of Fig. 4. The OPA starts in state $\Phi^0$, with $\chi_F^\pi \text{ret} \in \Phi_c^0$. Since $\# \prec \text{call}$, call is read by a push move, resulting in state $\Phi^1$. By rule $\square$, we have $\chi_F^\pi \text{ret}, \chi_L \in \Phi_p^1$. Since call $\prec \text{han}$, the next move is a push (step 2-3), consistently with the presence of $\chi_L$ in $\Phi^1_p$. Thus, the pending obligation for $\chi_F^\pi \text{ret}$ is stored onto the stack in $\Phi^1$. The OPA, then, reads the body of the chain between call and ret normally, until the stack symbol containing $\Phi^1$ is popped, in step 7-8. By rule $\square$, the temporal obligation is resumed in the next state $\Phi^4$, so $\chi_F^\pi \text{ret} \in \Phi^4_p$. Finally, ret is read by a shift which, by rule $\square$, may occur only if ret $\in \Phi^2$, ensuring the satisfaction of $\chi_F^\pi \text{ret}$, and bringing the OPA to the final state $\Phi^5$. Rule $\square$ fulfills the temporal obligation contained in $\Phi^5_p$, by preventing computations in which ret $\notin \Phi^5$, from continuing. Note that, had the next transition been a pop (e.g. because there was no ret and call $\succ \#$), the run would have been blocked by rule $\square$, preventing the OPA from reaching an accepting state, and from emptying the stack. The constructions for past operators, given in Appendix D, and for $\chi_F^\pi$ and $\chi_F^\pi$ work similarly.
For the hierarchical operators, we do not give an explicit OPA construction, but we rely on a translation into other POTL operands. For each hierarchical operator η in ϕ, we add a propositional symbol q_{(η)}^ϕ. The <η>-hierarchical operators consider the right contexts of chains sharing the same left context. To distinguish such positions, the following formula, evaluated in one of the right contexts, asserts that q_{(η)}^ϕ holds in the unique left context of the same chain, only.

\[ γ_{L, η} ≡ X_{ϕ}^\left< q_{(η)}^ϕ \right> ∧ \bigcirc_{<η>} (\square_ϕ \neg q_{(η)}^ϕ) ∧ \bigcirc_{<η>} (\square_ϕ \neg q_{(η)}^ϕ) \]

where \( \square_ϕ \equiv \neg (\top \mathcal{U}^{<\left< ϕ \right>}) \land \equiv \) is symmetric. This allows us to define the following equivalences for <η> operators, and symmetric ones for their >η> counterparts.

\[ \bigcirc_{<η>} ϕ ≡ γ_{L, \bigcirc_{<η>} ϕ} ∧ \bigcirc_{<η>} ((\neg X_{ϕ}^\left< q_{(η)}^ϕ \right> \mathcal{U}^{<\left< ϕ \right>}) (X_{ϕ}^\left< q_{(η)}^ϕ \right> ∧ ϕ)) \]
\[ \bigcirc_{<η>} ϕ ≡ γ_{L, \bigcirc_{<η>} ϕ} ∧ \bigcirc_{<η>} ((\neg X_{ϕ}^\left< q_{(η)}^ϕ \right> \mathcal{U}^{<\left< ϕ \right>}) (X_{ϕ}^\left< q_{(η)}^ϕ \right> ∧ ϕ)) \]

\[ ψ U_{<η>} \theta ≡ γ_{L, ψ U_{<η>} \theta} ∧ (X_{ϕ}^\left< q_{(η)}^ϕ \right> \mathcal{U}^{<\left< ϕ \right>}) (X_{ϕ}^\left< q_{(η)}^ϕ \right> ∧ ϕ) \]
\[ ψ S_{<η>} \theta ≡ γ_{L, ψ S_{<η>} \theta} ∧ (X_{ϕ}^\left< q_{(η)}^ϕ \right> \mathcal{U}^{<\left< ϕ \right>}) (X_{ϕ}^\left< q_{(η)}^ϕ \right> ∧ ϕ) \]

**Complexity.** The set Cl(ϕ) is linear in |ϕ|, the length of ϕ. Atoms(ϕ) has size at most \( 2^{\| Cl(ϕ) \|} = 2^{O(|ϕ|)} \), and the size of the set of states is the square of that. Moreover, the use of the equivalences for the hierarchical operators causes only a linear increase in the length of ϕ. Therefore,

**Theorem 3.** Given a POTL formula ϕ, it is possible to build an OPA accepting the language denoted by ϕ with at most \( 2^{O(|ϕ|)} \) states.

### 6 Conclusions

We introduced the temporal logic POTL by extensively redefining the semantics of OPTL. We proved its FO-completeness on finite OP words, and we gave an automata-
Theoretic model checking procedure for it. The next natural research step is the extension of such results to \( \omega \)-words, which, for model checking, may follow the approach sketched in [14] for OPTL. Whether POTL is strictly more expressive than OPTL also remains an open problem, although we conjecture OPTL is not FO-complete. A direct explanation of the completeness result of Corollary 1 also remains to be given. An important future step is the implementation of a model checker for POTL, to prove that OP languages and logics are suitable in practice to program verification.

References

1. Abrahams, D.: Exception-Safety in Generic Components. In: Generic Programming. pp. 69–79. Springer (2000). [https://doi.org/10.1007/3-540-39953-4_45]
2. Alur, R., Arenas, M., Barceló, P., Etessami, K., Immerman, N., Libkin, L.: First-order and temporal logics for nested words. LMCS 4(4) (2008). [https://doi.org/10.2168/LMCS-4(4:11)2008]
3. Alur, R., Chaudhuri, S., Madhusudan, P.: Software model checking using languages of nested trees. ACM Trans. Program. Lang. Syst. 33(5), 15:1–15:45 (2011). [https://doi.org/10.1145/2039346.2039347]
4. Alur, R., Madhusudan, P.: Visibly Pushdown Languages. In: ACM STOC (2004)
5. Alur, R., Madhusudan, P.: Adding nesting structure to words. JACM 56(3) (2009). [https://doi.org/10.1145/1516512.1516518]
6. Alur, R., Benedikt, M., Etessami, K., Godefroid, P., Reps, T., Yannakakis, M.: Analysis of recursive state machines. ACM Trans. Program. Lang. Syst. 27(4), 786–818 (2005). [https://doi.org/10.1145/1075382.1075387]
7. Alur, R., Etessami, K., Madhusudan, P.: A temporal logic of nested calls and returns. In: TACAS 2004. pp. 467–481. Springer (2004). [https://doi.org/10.1007/978-3-540-24730-2_35]
8. Ball, T., Rajamani, S.K.: Bebop: A symbolic model checker for boolean programs. In: SPIN Model Checking and Software Verification. pp. 113–130. Springer (2000). [https://doi.org/10.1007/10722468_7]
9. Barenghi, A., Crespi Reghizzi, S., Mandrioli, D., Panella, F., Pradella, M.: Parallel parsing made practical. Sci. Comput. Program. 112, 195–226 (2015). [https://doi.org/10.1016/j.scico.2015.09.002]
10. Bouajjani, A., Esparza, J., Habermehl, P.: On the verification problem of non-regular properties for nonregular processes. In: LICS 95. pp. 123–133 (1995). [https://doi.org/10.1109/LICS.1995.523250]
11. Bouajjani, A., Enea, C., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: CONCUR ’97. pp. 135–150. Springer (1997). [https://doi.org/10.1007/3-540-63141-0_10]
12. Bozzelli, L., Murano, A., Peron, A.: Timed context-free temporal logics. In: GandALF 2018. EPTCS, vol. 277, pp. 235–249. Open Publishing Association (2018). [https://doi.org/10.4204/EPTCS.277.17]
13. Bozzelli, L., Sánchez, C.: Visibly linear temporal logic. In: Automated Reasoning. pp. 418–433. Springer (2014). [https://doi.org/10.1007/978-3-319-08587-6_33]
14. Chiari, M., Mandrioli, D., Pradella, M.: Temporal logic and model checking for operator precedence languages. In: GandALF 2018. EPTCS, vol. 277, pp. 161–175. Open Publishing Association (2018). [https://doi.org/10.4204/EPTCS.277.12]
15. Crespi Reghizzi, S., Mandrioli, D.: Operator Precedence and the Visibly Pushdown Property. JCSS 78(6), 1837–1867 (2012). [https://doi.org/10.1016/j.jcss.2011.12.006]
16. Esparza, J., Hansel, D., Rossmanith, P., Schwoon, S.: Efficient algorithms for model checking pushdown systems. In: CAV 2000. LNCS, vol. 1855, pp. 232–247. Springer (2000)
17. Esparza, J., Kučera, A., Schwoon, S.: Model checking LTL with regular valuations for pushdown systems. Information and Computation 186(2), 355–376 (2003). https://doi.org/10.1016/S0890-5401(03)90013-9
18. Floyd, R.W.: Syntactic Analysis and Operator Precedence. JACM 10(3), 316–333 (1963). https://doi.org/10.1145/321172.321179
19. Godefroid, P., Yannakakis, M.: Analysis of boolean programs. In: TACAS 2013, pp. 214–229. Springer (2013). https://doi.org/10.1007/978-3-642-36742-7_16
20. Grune, D., Jacobs, C.J.: Parsing techniques: a practical guide. Springer, New York (2008). https://doi.org/10.1007/10.1007/978-0-387-68954-8
21. Kupferman, O., Piterman, N., Vardi, M.Y.: Model Checking Linear Properties of Prefix-Recognizable Systems. In: CAV 2002. LNCS, vol. 2404, pp. 371–385. Springer (2002). https://doi.org/10.1007/3-540-45657-0_31
22. Kupferman, O., Piterman, N., Vardi, M.Y.: Pushdown Specifications. In: LPAR 2002. LNCS, vol. 2514, pp. 262–277. Springer (2002). https://doi.org/10.1007/10.1007/3-540-36078-6_18
23. Lonati, V., Mandrioli, D., Panella, F., Pradella, M.: Operator precedence languages: Their automata-theoretic and logic characterization. SIAM J. Comput. 44(4), 1026–1088 (2015). https://doi.org/10.1137/140978818
24. Mandrioli, D., Pradella, M.: Generalizing input-driven languages: Theoretical and practical benefits. Computer Science Review 27, 61–87 (2018). https://doi.org/10.1016/j.cosrev.2017.12.001
25. Marx, M.: Conditional XPath, the first order complete XPath dialect. In: PODS ’04. p. 13. ACM Press, New York, USA (2004). https://doi.org/10.1145/1055558.1055562
26. Marx, M.: Conditional XPath. ACM Transactions on Database Systems 30(4), 929–959 (dec 2005). https://doi.org/10.1145/1114244.1114247
27. McNaughton, R.: Parenthesis Grammars. JACM 14(3), 490–500 (1967). https://doi.org/10.1145/321406.321411
28. Mehlhorn, K.: Pebbling mountain ranges and its application of DCFL-recognition. In: ICALP ’80. LNCS, vol. 85, pp. 422–435 (1980)
A Omitted Proofs: Semantics of POTL

In the following Lemma, we prove a few properties of the chain relation.

**Lemma 3 (Properties of the χ relation.).** Given an OP word w and positions i, j, h, k in it, the following properties hold.

1. If χ(i, j) and χ(h, k), then we have i < h < j ⇒ k ≤ j and i < k < j ⇒ i ≤ h.
2. If χ(i, j), then i < i + 1 and j − 1 > j.
3. Given j, there exists at most one single position i s.t. χ(i, j) and i < j or i = j; for any i′ s.t. χ(i′, j) and i′ > j we have i′ > i.
4. Given j, there exists at most one single position j s.t. χ(i, j) and i > j or i = j; for any j′ s.t. χ(i, j′) and i < j we have j′ < j.

**Proof.** In the following, we denote by c_p the character labeling word position p, and by writing \(c_{-1} = [x_0c_0x_1 \ldots x_nc_{n+1}]c_{n+1}\) we imply \(c_{-1}\) and \(c_{n+1}\) are the context of a simple or composed chain, in which either \(x_p = ε\), or \(c_{p-1}[x_p]^c\) is a chain, for each p.

1. Suppose, by contradiction, that χ(i, j), χ(h, k), and i < h < j, but k > j. Consider the case in which χ(i, j) is the innermost chain whose body contains h, so it is of the form \(c_{i} = x_0c_0 \ldots x_pc_p\ldots c_{n}x_{n+1}\)^c_j or \(c_{i} = x_0c_0 \ldots c HCI_{n+1}\)^c_j. By the definition of chain, we have either \(c_{h} \neq c_p\) or \(c_{h} > c_{j}\), respectively.
   Since χ(h, k), this chain must be of the form \(c_{h} = x_p x_{p+1} \ldots x_j x_{j+1}\)^c_k or \(c_{h} = x_p c_{h} \ldots c_{j} x_{j+1}\)^c_k, implying \(c_{h} \neq c_j\) or \(c_{h} < c_{j}\), respectively. This means there is a conflict in the OPM, contradicting the hypothesis that w is an OP word.
   In case χ(i, j) is not the innermost chain whose body contains h, we can reach the same contradiction by inductively considering the chain between i and j containing h in its body. Moreover, it is possible to reach a symmetric contradiction with the hypothesis χ(i, j), χ(h, k), and i < h < j, but i > h.
2. Trivially follows from the definition of chain.
3. Suppose, by contradiction, there exists a position h ≠ i, and w.l.o.g., h < i, s.t. χ(h, j) and h < j. Since i < j, by the definition of chain, j must be part of the body of another composed chain whose left context is i. So, w contains a structure of the form \(c_{i} = x_0c_0 \ldots c_{i} x_{i} c_{i+1}\)^c_j where \(|x_0| ≥ 1\), \(c_{i} = x_0 x_1 \ldots x_j x_{j+1}\), and \(k > j\) is such that χ(i, k). This contradicts the hypothesis that χ(h, j) and h < i, because such a chain would cross χ(i, k), contradicting property (1).
   Similarly, if χ(h, j), χ(i, j), h ≥ j, and h < i, then w contains a structure \(c_{i} = x_0c_0 \ldots c_{i} x_{i} c_{i+1}\)^c_j, with \(|x_i| ≥ 1\) and \(c_{i} = x_0 x_1 \ldots x_j x_{j+1}\). By the definition of chain, we have j > i, which contradicts the hypothesis that either i < j or i = j. This proves that i is unique.
   For the second part of the property, suppose there exists a position i′ s.t. χ(i′, j) and i′ > j, but i′ < i (the case i′ = i is trivial). The only way of having both χ(i′, j) and χ(i, j) in this case is χ(i′, j) and i′ > j, which contradicts the hypothesis that i > j, which contradicts the hypothesis that i < j or i = j.
4. The proof is symmetric to the previous one.

In the rest of this section, we prove the expansion laws of the until and since operators.
Lemma 4. Given a word $w$ on an OP alphabet $(\mathcal{P}(AP), M_{P(AP)})$, two POTL formulas $\varphi$ and $\psi$, and a non-empty set $\Pi \subseteq \{<, =, >\}$, for any position $i \in w$ the following equivalence holds:

$$\varphi \mathcal{U}^I \psi \equiv \psi \lor \left( \varphi \land \left( \circ^I (\varphi \mathcal{U}^I \psi) \lor \chi^I_F(\varphi \mathcal{U}^I \psi) \right) \right).$$

Proof. $[\Rightarrow]$ Suppose $\varphi \mathcal{U}^I \psi$ holds in $i$. If $\psi$ holds in $i$, the equivalence is trivially verified. Otherwise, $\varphi \mathcal{U}^I \psi$ is verified by an OPSP $i = i_0 < i_1 < \cdots < i_n = j$ with $n \geq 1$, s.t. $(w,i_p) \models \varphi$ for $0 \leq p < n$ and $(w,i_n) \models \psi$. Note that, by the definition of OPSP, any suffix of such a path is also an OPSP ending in $j$. Consider position $i$: $\varphi$ holds in it, and it can be either

- $i_1 = i + 1$. Then there exists $\pi \in \Pi$ s.t. $i \pi (i + 1)$, and path $i_1 < i_2 < \cdots < i_n = j$ is the OPSP between $i_1$ and $j$, and $\varphi$ holds in all $i_p$ with $1 \leq p < n$, and $\psi$ in $j_n$. So, $\varphi \mathcal{U}^I \psi$ holds in $i_1$, and $\circ^I(\varphi \mathcal{U}^I \psi)$ holds in $i$.

- $i_1 > i + 1$. Then, $\chi(i,i_1)$, and there exists $\pi \in \Pi$ s.t. $i \pi i_1$. Since $i_1 < i_2 < \cdots < i_n = j$ is the OPSP from $i_1$ to $j$, $\varphi \mathcal{U}^I \psi$ holds in $i_1$, and so does $\chi^I_F(\varphi \mathcal{U}^I \psi)$ in $i$.

$[\Leftarrow]$ Suppose $\psi \lor \left( \varphi \land \left( \circ^I (\varphi \mathcal{U}^I \psi) \lor \chi^I_F(\varphi \mathcal{U}^I \psi) \right) \right)$ holds in $i$. The case $(w,i) \models \psi$ is trivial. Suppose $\psi$ does not hold in $i$. Then $\varphi$ holds in $i$, and either:

- $\circ^I(\varphi \mathcal{U}^I \psi)$ holds in $i$. Then, we have $i \pi (i + 1)$, $\pi \in \Pi$, and there is an OPSP $i + 1 = i_1 < i_2 < \cdots < i_n = j$, with $\varphi$ holding in all $i_p$ with $1 \leq p < n$, and $\psi$ in $i_n$. If $i$ is not the left context of any chain, then $i = i_0 < i_1 < i_2 < \cdots < i_n$ is an OPSP satisfying $\varphi \mathcal{U}^I \psi$ in $i$. Otherwise, let $k = \min\{h \mid \chi(i,h)\}$. Since $i$ is the left context of a chain, $\pi \in \Pi$, or $\circ^I(\varphi \mathcal{U}^I \psi)$ would not be true in $i$.

Suppose $k > j$. This is always the case if $\not\exists \Pi$, because then there is no position $h \leq i$ s.t. $\chi(h,i_p)$ for any $1 \leq p \leq n$. So, adding $i$ to the OPSP generates another OPSP, because there is no position $h$ s.t. $\chi(i,h)$ with $h \leq j$, and the successor of $i$ in the path can only be $i_1 = i + 1$.

Suppose $k \leq j$. Let $k' = \max\{h \mid h \leq j \land \chi(i,h) \land \lor_{\pi \in \Pi} i \pi k\}$. Since $i_1 > i$, and chains cannot cross each other, there exists a value $q$, $1 \leq q \leq n$, s.t. $i_q = k'$. The path $i = i_0 < i_q < \cdots < i_n = j$ is an OPSP by definition, and $\varphi$ holds both in $i$ and $i_q$. So, this path makes $\varphi \mathcal{U}^I \psi$ true in $i$.

- $\chi^I_F(\varphi \mathcal{U}^I \psi)$ holds in $i$. Then, there exists a position $k$ s.t. $\chi(i,k)$ and $i \pi k$ with $\pi \in \Pi$, and $\varphi \mathcal{U}^I \psi$ holds in $k$, because of an OPSP $k = i_1 < i_2 < \cdots < i_q = j$. If $k = \max\{h \mid h \leq j \land \chi(i,h) \land \lor_{\pi \in \Pi} i \pi k\}$, then $i = i_0 < i_1 < i_2 < \cdots < i_n$ is an OPSP by definition, and since $\varphi$ holds in $i$, $\varphi \mathcal{U}^I \psi$ is satisfied in it. Otherwise, let $k' = \max\{h \mid h \leq j \land \chi(i,h) \land \lor_{\pi \in \Pi} i \pi k\}$. Since $i_1 > i$ and chains cannot cross, there exists a value $q$, $1 < q \leq n$, s.t. $i_q = k'$, $i_q < i_{q+1} < \cdots < i_n = j$ is an OPSP, so $\varphi \mathcal{U}^I \psi$ holds in $i_q$ as well. The path $i < i_q < \cdots < i_n$ is an OPSP, and $\varphi \mathcal{U}^I \psi$ holds in $i$.

The proof for the OPS since operator is analogous.

Lemma 5. Given a word $w$ on an OP alphabet $(\mathcal{P}(AP), M_{P(AP)})$, and two POTL formulas $\varphi$ and $\psi$, for any position $i \in w$ the following equivalence holds:

$$\varphi \mathcal{H}_i \psi \equiv (\psi \land \chi^P_F) \lor (\varphi \land \circ^P(\varphi \mathcal{H}_i \psi)).$$
B.1 Precedence and Chain Next/Back

A three-variable property, which holds in regular words. For any pair of FO variables, the path \( i_1 < \cdots < i_n \), and only propositions in \( \text{AP} \) hold in \( i_1 \), and so does \( \chi_p \). Otherwise, the path \( i_1 < \cdots < i_n \) is also a YPHP, so \( \varphi \mathcal{U}_H \psi \) is true in \( i_1 \). Therefore, \( \bigcirc_H^\ast(\varphi \mathcal{U}_H^\ast \psi) \) holds in \( i \), and so does \( \varphi \).

\[ \forall i \in \mathbb{N}, \varphi \mathcal{U}_H^\ast \psi \text{ holds in } i. \]

The proofs for the other hierarchical operators are analogous.

### B First Order Semantics

We now show that POTL can be expressed with First Order logic equipped with monadic relations for atomic propositions, a total order on positions, and the chain relation between pairs of positions. We define below the translation function \( \nu \), such that for any POTL formula \( \varphi \), word \( w \) and position \( x, (w, x) \models \nu(\varphi) \) iff \( (w, x) \models \varphi \). The translation for propositional operators is trivial.

For temporal operators, we first need to define a few auxiliary formulae. We define the successor relation as the FO formula

\[ \text{succ}(x, y) \equiv x < y \land \exists z (x < z \land z < y). \]

In the following, \( \pi \in \{<, \doteq, >\} \) and \( \Pi \subseteq \{<, \doteq, >\} \). The PR between positions can be expressed by means of propositional combinations of monadic atomic relations only. Given a set of atomic propositions \( a \subseteq \text{AP} \), we define formula \( \sigma_a(x) \), stating that all and only propositions in \( a \) hold in position \( x \), as follows:

\[ \sigma_a(x) \equiv \bigwedge_{p \in a} p(x) \land \bigwedge_{p \in \text{AP} \setminus a} \neg p(x) \]

For any pair of FO variables \( x, y \) and \( \pi \in \{<, \doteq, >\} \), we can build formula

\[ x \pi y \equiv \bigvee_{a, b \subseteq \text{AP} \mid \pi \cap ab} (\sigma_a(x) \land \sigma_b(y)). \]

The following translations make use of the three FO variables \( x, y, z \), only. This, in addition to a FO-completeness result for POTL, proves that FO on OP words retains the three-variable property, which holds in regular words.

#### B.1 Precedence and Chain Next/Back

\[ \nu \circ H^\ast(\varphi) \equiv \exists y \left( \text{succ}(x, y) \land \bigvee_{\pi \in \Pi} (x \pi y) \land \exists x (x = y \land \nu(\varphi)) \right) \]
Finally, and above.

\[
\nu_{\varphi \psi \pi}(x) \equiv \exists y \left( \text{succ}(y, x) \land \bigvee_{\pi \in \Pi} (y \land (x \land \exists x (x = y \land \nu_{\varphi}(x)))) \right)
\]

Note that \text{succ}(y, x) can be obtained by exchanging \(x\) and \(y\) in the definition of \text{succ}(x, y) above.

\[
\nu_{\chi \psi \pi}(x) \equiv \exists y (x < y \land \chi(x, y) \land \bigvee_{\pi \in \Pi} (y \land \exists x (x = y \land \nu_{\psi}(x))))
\]

\[
\nu_{\gamma \psi \pi}(x) \equiv \exists y (y < x \land \gamma(y, x) \land \bigvee_{\pi \in \Pi} (y \land \exists x (x = y \land \nu_{\psi}(x))))
\]

**B.2 Operator Precedence Summary Until/Since**

The translation for the OPS until operator can be obtained by noting that, given two positions \(x\) and \(y\), the OPS path between them, if it exists, is the one that skips all chain bodies entirely contained between them, among those whose contexts are in a relation in \(\Pi\). The fact that a position \(z\) is part of such path can be expressed with formula \(\neg \gamma(x, y, z)\) as follows:

\[
\gamma(x, y, z) \equiv \gamma_L(x, z) \land \gamma_R(y, z)
\]

\[
\gamma_L(x, z) \equiv \exists y \left( x \leq y \land (y < z \land \exists x (z < x \land \chi(y, x) \land \bigvee_{\pi \in \Pi} (y \land \exists x (x = y \land \nu_{\psi}(x)))) \right)
\]

\[
\gamma_R(y, z) \equiv \exists x \left( z < x \land x \leq y \land \exists y (y < z \land \chi(y, x) \land \bigvee_{\pi \in \Pi} (y \land \exists x (x = y \land \nu_{\psi}(x)))) \right)
\]

\(\gamma(x, y, z)\) is true if and only if \(z\) is not part of the OPS path between \(x\) and \(y\), while \(x \leq z \leq y\). In particular, \(\gamma_L(x, z)\) asserts that \(z\) is part of the body of a chain whose left context is after \(x\), and \(\gamma_R(y, z)\) states that \(z\) is part of the body of a chain whose right context is before \(y\). Only chains whose contexts are in a relation in \(\Pi\) are considered. Since chain bodies cannot cross, either the two chain bodies are actually the same one, or one of them is a sub-chain nested into the other. In both cases, \(z\) is part of a chain body entirely contained between \(x\) and \(y\), and is thus not part of the path.

Moreover, for such a path to exist, each one of its positions must be in one of the admitted PR with the next one. Formula

\[
\delta(y, z) \equiv \exists x (z < x \land x \leq y \land \bigvee_{\pi \in \Pi} (z \land \chi(x, z) \land \neg \gamma(z, y, x) \land (\text{succ}(z, x) \lor \chi(z, x)))
\]

asserts this for each position \(z\), with the path ending in \(y\). (Note that by exchanging \(x\) and \(z\) in the definition of \(\gamma(x, y, z)\) above, one can obtain \(\gamma(z, y, x)\) without using any additional variable.) Finally, \(\varphi \psi \Pi \chi\) can be translated as follows:

\[
\nu_{\varphi \psi \Pi \chi}(x) \equiv \exists y (x \leq y \land \exists x (x = y \land \nu_{\chi}(x)) \land \forall z (z \leq x \land \exists y (y < z \land \neg \gamma(x, y, z) \iff \exists x (x = z \land \nu_{\chi}(x)) \land \delta(y, z)))
\]
The translation for the OPS since operator is similar:

$$\nu_{\phi, \delta_{\phi}}(x) \equiv \exists y \left( y \leq x \land \exists x (x = y \land \nu_{\phi}(x)) \land \forall z \left( y < z \land z \leq x \land \neg \gamma(y, x, z) \implies \exists x (z = x \land \nu_{\phi}(x) \land \delta(x, z)) \right) \right)$$

### B.3 Hierarchical Next/Back

Finally, the translations for hierarchical operators.

$$\nu_{\bigcirc_{\bigtriangledown_{\bigtriangledown}}^{\chi_{\gamma}}} \equiv \exists y \left( y < x \land \chi(y, x) \land y < x \land \exists z \left( z < x \land \chi(y, z) \land y < z \land \exists x (x = z \land \nu_{\phi}(x)) \land \forall z \left( z < y \land y < z \implies \forall z (\chi(z, x) \land z < x \implies \neg \chi(z, y)) \right) \right) \right)$$

$$\nu_{\bigtriangledown_{\bigcirc_{\bigtriangledown}}} \equiv \exists y \left( y < x \land \chi(y, x) \land y < x \land \exists z \left( z < x \land \chi(y, z) \land y < z \land \exists x (x = z \land \nu_{\phi}(x)) \land \forall z \left( z < y \land y < z \implies \forall z (\chi(z, x) \land z < x \implies \neg \chi(z, y)) \right) \right) \right)$$

$$\nu_{\bigtriangledown_{\bigtriangledown_{\bigcirc}}} \equiv \exists y \left( x < y \land \chi(x, y) \land x > y \land \exists z \left( x < z \land \chi(x, y) \land x > y \land \exists x (x = z \land \nu_{\phi}(x)) \land \forall z \left( x < y \land y < z \implies \forall z (\chi(x, z) \land x > z \implies \neg \chi(y, z)) \right) \right) \right)$$

$$\nu_{\bigtriangledown_{\bigcirc_{\bigtriangledown}}} \equiv \exists y \left( x < y \land \chi(x, y) \land x > y \land \exists z \left( z < x \land \chi(x, y) \land z > y \land \exists x (x = z \land \nu_{\phi}(x)) \land \forall z \left( z < y \land y < z \implies \forall z (\chi(x, z) \land x > z \implies \neg \chi(y, z)) \right) \right) \right)$$
B.4 Hierarchical Until/Since

$$\nu_\psi U_\psi(x) \equiv \exists z (z < x \land z < x \land \chi(z, x) \land$$
$$\exists y (x \leq y \land \chi(z, y) \land z < y \land \exists x (x = y \land \nu_\psi(x))) \land$$
$$\forall z (x \leq z \land z < y \land \exists y (y < x \land y < x \land \chi(y, x) \land \chi(y, z))$$
$$\implies \exists x (x = z \land \nu_\psi(x)))$$

$$\nu_\psi S_\psi(x) \equiv \exists z (z < x \land z < x \land \chi(z, x) \land$$
$$\exists y (y \leq x \land \chi(z, y) \land z < y \land \exists x (x = y \land \nu_\psi(x))) \land$$
$$\forall z (y < z \land z < x \land \exists y (y < x \land y < x \land \chi(y, x) \land \chi(y, z))$$
$$\implies \exists x (x = z \land \nu_\psi(x)))$$

$$\nu_\psi U_\psi(x) \equiv \exists x (x < z \land x < z \land \chi(x, z) \land$$
$$\exists y (x \leq y \land \chi(y, x) \land x > y \land \exists x (x = y \land \nu_\psi(x))) \land$$
$$\forall z (x \leq z \land z < y \land \exists y (x < y \land x < y \land \chi(x, y) \land \chi(z, y))$$
$$\implies \exists x (x = z \land \nu_\psi(x)))$$

$$\nu_\psi S_\psi(x) \equiv \exists x (x < z \land x < z \land \chi(x, z) \land$$
$$\exists y (y \leq x \land \chi(y, x) \land y > z \land \exists x (x = y \land \nu_\psi(x))) \land$$
$$\forall z (y < z \land z < x \land \exists y (x < y \land x < y \land \chi(x, y) \land \chi(z, y))$$
$$\implies \exists x (x = z \land \nu_\psi(x)))$$

C Omitted Proofs: Conditional XPath Translation

C.1 Completeness of CXPath on OPM-compatible trees

First, we give an argument for Theorem [1] by proving a more general result.

Lemma 6. Let $M$ be the set of algebraic structures with common signature $\sigma$, let $\mathcal{L}$ be a logic formalism that is FO-complete on $M$, and let $N$ be FO-definable subset of $M$. Then, $\mathcal{L}$ is also FO-complete on $N$. 
Proof. Since $N$ is FO-definable, there exists a FO formula $\varphi_N$ such that, for any $M \in \mathcal{M}$, we have $M \models \varphi_N$ iff $M \in N$. Thus, any FO formula $\psi$ on $N$ is equivalent to $\psi \land \varphi_N$ on $\mathcal{M}$.

Since $\mathcal{L}$ is FO-complete, there exists an $\mathcal{L}$-formula $\Phi$ such that, for any $M \in \mathcal{M}$, $M \models \Phi$ iff $M \models \psi \land \varphi_N$. Therefore, since $N \subseteq \mathcal{M}$, we also have $N \models \Phi$ iff $N \models \psi \land \varphi_N$ for any $N \in \mathcal{N}$. By construction, $\varphi_N \equiv \top$ on any $N \in \mathcal{N}$, and thus $N \models \Phi$ iff $N \models \psi$.

In our case, $\mathcal{M}$ is the set of all unranked ordered trees $T$, while $\mathcal{N}$ is $\mathcal{T}_{\mathcal{M}_{\mathcal{P}(\mathcal{AP})}}$, for a given OPM $\mathcal{M}_{\mathcal{P}(\mathcal{AP})}$, $\mathcal{L}$ is the logic CXPath, proved to be FO-complete in [26]. We only need to show that the set $\mathcal{T}_{\mathcal{M}_{\mathcal{P}(\mathcal{AP})}}$ is FO-definable.

Note that the actual signature of $\mathcal{T}$ differs from the one reported in Section 4 in the fact that the transitive and reflexive closures of the $R_\parallel$ and $R_\equiv$ relations are used (denoted resp. $R_\parallel^*$ and $R_\equiv^*$). We also use $R_\equiv^*$ to denote the transitive closure of $R_\parallel$. Moreover, the signature contains monadic predicates for propositional symbols, instead of the labeling function $L$. First, we define the following FO formula on $\mathcal{T}$, which is true iff a node $y$ is the right context candidate of another node $x$:

$$\text{Rcc}(x, y) \equiv xR_\equiv y \lor (\neg \exists z(xR_\equiv z) \land \exists z(zR_\parallel^* x \land zR_\equiv y) \land \forall y(zR_\parallel^* y \land yR_\equiv^* x \implies \neg \exists x(yR_\equiv(x))))$$

We also express sets of atomic propositions and PR as detailed at the beginning of Section 3 and we define the following shortcuts:

- leftmost($x$) $\equiv \neg \exists y(yR_\equiv x)$
- rightmost($x$) $\equiv \neg \exists y(xR_\equiv y)$

The following formula $\varphi_{\mathcal{T}_{\mathcal{M}_{\mathcal{P}(\mathcal{AP})}}}$ defines the set $\mathcal{T}_{\mathcal{M}_{\mathcal{P}(\mathcal{AP})}}$:

$$\varphi_{\mathcal{T}_{\mathcal{M}_{\mathcal{P}(\mathcal{AP})}}} \equiv \forall x \left( (\neg \exists y(yR_\parallel x) \implies (\sigma_\parallel(x) \land \exists y(xR_\parallel y \land \text{rightmost}(y) \land \sigma_\parallel(y) \land (\text{leftmost}(y) \lor \exists z(zR_\equiv y \land \text{leftmost}(z) \land \neg \#(z)))))) \land (\exists y(yR_\parallel x) \implies \neg \#(x)) \land (\forall y(xR_\parallel y \land \text{rightmost}(y) \implies x < y \lor x \equiv y) \land \forall y(xR_\equiv y \land \neg \text{rightmost}(y) \implies x < y) \land \neg \exists y(xR_\parallel y \land x \equiv y) \implies \forall y(yR_{\text{occ}}(x, y) \implies x > y) \right)$$

The first two lines say that the root is labeled with $\#$ and it has at most two children, the rightmost one labeled with $\#$. The third line states that no other position is labeled with $\#$. The remaining lines describe the PR among sets of labels of each node, as described in Section 4.

C.2 POTL Translation of CXPath

We now proceed with the proof of the translation of CXPath into POTL, with one lemma for each translated operator.
Lemma 7. Given a set of atomic propositions AP, and OPM $M_{\forall \langle \phi, \psi \rangle}$, for every $\chi_{\text{until}}$ formula $\downarrow (\phi, \psi)$, and for any OP word $w$ based on $M_{\forall \langle \phi, \psi \rangle}$ and position $i$ in $w$, we have

$$(T_w, \tau(i)) \models \downarrow (\phi, \psi) \iff (w, i) \models \iota_X(\downarrow (\phi, \psi)).$$

$T_w \in T_{M_{\forall \langle \phi, \psi \rangle}}$ is the tree obtained by applying function $\tau$ to every position in $w$, such that for any position $j$ in $w (T_w, \tau(i')) \models \varphi \iff (w, i') \models \iota_X(\varphi)$, and likewise for $\psi$.

Proof. $[\Rightarrow]$ Suppose $(T_w, \tau(i)) \models \downarrow (\phi, \psi)$. Let $s = \tau(j)$ be the first tree node of the path witnessing $\downarrow (\phi, \psi)$, and $r$ s.t. $rR \subseteq s$ be its parent.

We shall now inductively prove that $\varphi' \mathcal{U}^{\leq} \psi'$ holds in $j$. If $s$ is the last node of the path, then $\psi'$ holds in $j$ and so does, trivially, $\varphi' \mathcal{U}^{\leq} \psi'$. Otherwise, consider any node $t = \tau(k)$ of the path, except the last one, and suppose $\varphi' \mathcal{U}^{\leq} \psi'$ holds in $k'$ s.t. $t' = \tau(k')$ is the next node in the path. If $t'$ is the leftmost child of $t$, then $k' = k + 1$ and either $k < k'$ or $k = k'$. In both cases, $\mathcal{O}^{\leq}(\varphi' \mathcal{U}^{\leq} \psi')$ holds in $k$. If $t'$ is not the leftmost child, then $\chi(k, k')$ and $k < k'$ or $k = k'$. In both cases, $\chi_F^{\leq}(\varphi' \mathcal{U}^{\leq} \psi')$ holds in $k$. So, by expansion formula $\varphi' \mathcal{U}^{\leq} \psi' \equiv \psi' \lor \varphi' \land (\mathcal{O}^{\leq}(\varphi' \mathcal{U}^{\leq} \psi') \lor \chi_F^{\leq}(\varphi' \mathcal{U}^{\leq} \psi'))$, $\varphi' \mathcal{U}^{\leq} \psi'$ holds in $k$ and, by induction, also in $j$.

Suppose $s$ is the leftmost child of $r$: $j = i + 1$, and either $i < j$ or $i = j$, so $\mathcal{O}^{\leq}(\varphi' \mathcal{U}^{\leq} \psi')$ holds in $i$. Otherwise, $\chi(i, j)$ and either $i < j$ or $i = j$. In both cases, $\chi_F^{\leq}(\varphi' \mathcal{U}^{\leq} \psi')$ holds in $i$.

$[\Leftarrow]$. Suppose $\downarrow (\varphi', \psi')$ holds in $i$. If $\mathcal{O}^{\leq}(\varphi' \mathcal{U}^{\leq} \psi')$ holds in $i$, then $\varphi' \mathcal{U}^{\leq} \psi'$ holds in $j = i + 1$, and either $i < j$ or $i = j$; then $s = \tau(j)$ is the leftmost child of $\tau(i)$. If $\chi_F^{\leq}(\varphi' \mathcal{U}^{\leq} \psi')$ holds in $i$, then $\varphi' \mathcal{U}^{\leq} \psi'$ holds in $j$ s.t. $\chi(i, j)$ and either $i < j$ or $i = j$; $s = \tau(j)$ is a child of $\tau(i)$ in this case as well.

We shall now prove that if $\varphi' \mathcal{U}^{\leq} \psi'$ holds in a position $j$ s.t. $\tau(i)R \subseteq (\tau(j)$, then $\downarrow (\varphi, \psi) \models \tau(i)$. If $\varphi' \mathcal{U}^{\leq} \psi'$ holds in $j$, then there exists an OPSP from $j$ to $h$ s.t. $\tau(h) \models \psi'$ and $\varphi'$ holds in all positions $j \leq k < k$ of the path, and $(T_w, \tau(k)) \models \varphi$. In any such $k$, $\varphi' \mathcal{U}^{\leq} \psi' \equiv \psi' \lor (\varphi' \land (\mathcal{O}^{\leq}(\varphi' \mathcal{U}^{\leq} \psi') \lor \chi_F^{\leq}(\varphi' \mathcal{U}^{\leq} \psi'))) \models \varphi$. Since $\psi'$ does not hold in $k$, either $\mathcal{O}^{\leq}(\varphi' \mathcal{U}^{\leq} \psi')$ or $\chi_F^{\leq}(\varphi' \mathcal{U}^{\leq} \psi')$ hold in it. Therefore, the next position in the path is $k'$ s.t. either $k' = k + 1$ or $\chi(k, k')$, and either $k < k'$ or $k = k'$, and $(w, k') \models \varphi' \mathcal{U}^{\leq} \psi'$. Therefore, $\tau(k')$ is a child of $\tau(k)$. So, there is a sequence of nodes $s_0, s_1, \ldots, s_n$ in $T_w$ s.t. $\tau(i)R \subseteq s_0$, and $s_1, s_2, \ldots, s_n$ in $(T_w, s_{n-1}) \models \varphi$ for $0 \leq i < n$, and $(T_w, s_n) \models \psi$. This is a path making $\downarrow (\varphi, \psi)$ true in $\tau(i)$.

Lemma 8. Given a set of atomic propositions AP, and OPM $M_{\forall \langle \phi, \psi \rangle}$, for every $\chi_{\text{until}}$ formula $\downarrow (\phi, \psi)$, and for any OP word $w$ based on $M_{\forall \langle \phi, \psi \rangle}$ and position $i$ in $w$, we have

$$(T_w, \tau(i)) \models \downarrow (\phi, \psi) \iff (w, i) \models \iota_X(\downarrow (\phi, \psi)).$$

$T_w \in T_{M_{\forall \langle \phi, \psi \rangle}}$ is the tree obtained by applying function $\tau$ to every position in $w$, such that for any position $j$ in $w (T_w, \tau(i')) \models \varphi \iff (w, i') \models \iota_X(\varphi)$, and likewise for $\psi$.

Proof. The proof of this lemma is analogous to the one of Lemma 7 and is therefore omitted.
Lemma 9. Given a set of atomic propositions AP, and OPM $M_{P(AP)}$, for every $X_{until}$ formula $\Rightarrow (\varphi, \psi)$, and for any OP word $w$ based on $M_{P(AP)}$ and position $i$ in $w$, we have

$$(T_w, \tau(i)) \models (\varphi, \psi) \iff (w, i) \models \lambda_X(\Rightarrow (\varphi, \psi)).$$

$T_w \in T_{M_{P(AP)}}$ is the tree obtained by applying function $\tau$ to every position in $w$, such that for any position $j$ in $w$ $(T_w, \tau(j))$ $\models \varphi \iff (w, i') \models \lambda_X(\varphi)$, and likewise for $\psi$.

Proof. [$\Rightarrow$] Suppose $\Rightarrow (\varphi, \psi)$ holds in $s = \tau(i)$. Then, node $r = \tau(h)$ s.t. $rR_1s$ has at least two children, and $\Rightarrow (\varphi, \psi)$ is witnessed by a path starting in $t = \tau(j)$ s.t. $sR_\Rightarrow t$, and ending in $v = \tau(k)$. We have the following cases:

1. $s$ is not the leftmost child of $r$.
   (a) $h \leq k$. By the construction of $T_w$, for any node $t'$ in the path, there exists a position $j' \in w$ s.t. $t' = \tau(j')$, $\chi(h, j')$ and $h < j'$. The path made by such positions is a YPHP, and $\varphi' \Upsilon_H \psi'$ is true in $j$. Since $s$ is not the leftmost child of $r$, we have $\chi(h, i)$, and $h < i$, so (14) $(\bigcirc_H^p(\varphi' \Upsilon_H \psi'))$ holds in $i$.
   (b) $h \geq k$, so $s = \varphi$ holds in all siblings between $s$ and $v$ (excluded), and $\varphi'$ holds in the corresponding positions of $w$. All such positions, if any, are s.t. $\chi(h, j)$ and $h < j$, and they form a YPHP, so $\bigcirc_H^p(\varphi' \Upsilon_H \psi')$ never holds in $i$. Moreover, since $\psi$ holds in $v$, $\psi'$ holds in $k$. Note that $\chi^+_H$ in $i$ uniquely identifies position $h$, and $\chi_H^+$ evaluated in $h$ identifies $k$. So, (15) holds in $i$.

2. $s$ is the leftmost child of $r$. In this case, we have $i = h + 1$ and $h < i$ (if $h = i$, then $r$ would have only one child).
   (a) $h < k$. $\otimes^+$ evaluated in $i$ identifies position $h$. $\psi'$ holds in $k$, and $\otimes_H^+(\top \Sigma^+ H \sim \varphi')$ does not, because in all positions between $i$ and $k$ (excluded) corresponding to children of $r$, $\psi'$ holds. Note that all such positions form a YPHP, but $i$ is not part of it ($i = h + 1$, so $\sim \chi(h, i)$), and is not considered by $\top \Sigma^+ H \sim \varphi'$. So, (16) holds in $i$.
   (b) $h \geq k$, so $s = \varphi$ holds in $k$, and $\varphi$ holds in all children of $r$, except possibly the first ($i$) and the last one ($v$). These are exactly all positions s.t. $\chi(h, j)$ and $h < j$. Since $\varphi'$ holds in all of them by hypothesis, $\sim \chi^+_H \sim \psi'$ holds in $h$. Since $\psi$ holds in $v$, $\psi'$ holds in $k$, and $\chi^+_H \psi'$ in $h$. So, (17) holds in $i$.

[$\Leftarrow$] Suppose (14) $(\bigcirc_H^p(\varphi' \Upsilon_H \psi'))$ holds in a position $i$ in $w$. Then, there exists a position $h$ s.t. $\chi(h, i)$ and $h < i$, and a position $j$ s.t. $\chi(h, j)$ and $h < j$. Then the path starting in $t = \tau(j)$ and ending in $v = \tau(k)$ witnesses the truth of $\Rightarrow (\varphi, \psi)$ in $s$.

Suppose (15) $(\sim \bigcirc_H^+(\top \Sigma^+ H \sim \varphi') \land \chi^+_H(\varphi') \psi)$ holds in position $i$ in $w$. If $\chi^+_H(\varphi')$ holds in $i$, then there exists a position $h$ s.t. $\chi(h, i)$ and $h < i$, and a position $k$ s.t. $\chi(h, k)$ and $h = k$, and $\psi'$ holds in $k$. $v = \tau(k)$ is the rightmost child of $r = \tau(h)$, parent of $s = \tau(i)$. Moreover, if $\sim \bigcirc_H^+(\top \Sigma^+ H \sim \varphi')$ holds in $i$, then either:
\(\neg \imath \circ \c H \top \text{ holds, i.e. there is no position } j > i \text{ s.t. } \chi(h, j) \text{ and } h < j, \text{ so } v \text{ is the immediate right sibling of } s. \) In this case \(\Rightarrow (\varphi, \psi) \text{ holds in } s \) because \(\psi \text{ holds in } v.\)

\(\neg (\imath \circ \c H \top) \text{ holds in } j, \text{ the first position after } i \text{ s.t. } \text{chain}(h, j) \text{ and } h < j. \) This means \(\varphi' \text{ holds in all positions } j' \text{ s.t. } \text{chain}(h, j') \text{ and } h < j' \text{ greater or equal to } j.\)

Consequently, the tree nodes corresponding to these positions plus \(v = \tau(k)\) form a path witnessing \(\Rightarrow (\varphi, \psi), \text{ which holds in } s = \tau(i).\)

This corresponds to case 1.b

Suppose (16) \((\circ \c H \top (\psi' \land \neg \imath \circ \c H (\top \circ H \neg \varphi'))) \text{ holds in } i. \) Let \(h = i - 1, \text{ with } h < i \) (it exists because \(\circ \c H \top \text{ is true.}\) There exists a position \(k \text{ in which } \psi' \text{ holds, so }\)

\(\text{there is no position } j < k \text{ s.t. } \chi(h, j) \text{ and } h < j, \text{ so } v \text{ is the second child of } r = \tau(h), s = \tau(i) \text{ being the first one. So, } \Rightarrow (\varphi, \psi) \text{ trivially holds in } s \text{ because } \psi \text{ holds in the next sibling. Otherwise, let } j \text{ be the rightmost position lower than } k \text{ s.t. } \chi(h, j) \text{ and } h < j.\)

\(\neg (\imath \circ \c H \top) \text{ holds in } j, \text{ so } \varphi' \text{ holds in all positions } j' \text{ between } i \text{ and } k \text{ that are part of the hierarchical path, i.e. s.t. } \chi(h, j') \text{ and } h < j'. \) The corresponding tree nodes form a path ending in \(v = \tau(k)\) that witnesses the truth of \(\Rightarrow (\varphi, \psi) \text{ in } s \text{ case 2.a.} \)

If (17) \((\circ \c H \top (\chi' \land \neg \chi')) \text{ holds in } i, \) then let \(h = i - 1, h < i, \text{ and } S = \tau(i) \text{ is the leftmost child of } r = \tau(h). \) Since \(\chi' \land \neg \chi' \text{ holds in } h, \text{ there exists a position } k, \text{ s.t. } \chi(h, k) \text{ and } h \leq k, \text{ in which } \psi' \text{ holds. So, } \psi \text{ holds in } v = \tau(k), \text{ which is the rightmost child of } r, \text{ by construction. Moreover, in all positions s.t. } \chi(h, j) \text{ and } h < j, \psi' \text{ holds. }\)

Hence, \(\varphi \text{ holds in all corresponding nodes } t = \tau(j), \text{ which are all nodes between } s \text{ and } v, \text{ excluded. This, together with } \psi \text{ holding in } v, \text{ makes a path that verifies } \Rightarrow (\varphi, \psi) \text{ in } s \text{ case 2.b.} \)

**Lemma 10.** Given a set of atomic propositions AP, and OPM \(M_{\varphi(AP)} \), for every \(X_{\text{null}} \) formula \(\equiv (\varphi, \psi), \text{ and for any OP word } w \text{ based on } M_{\varphi(AP)} \text{ and position } i \text{ in } w, \text{ we have} \)

\[(T_w, \tau(i)) \models (\varphi, \psi) \iff (w, i) \models \iota_X(\equiv (\varphi, \psi)). \]

\(T_w \in T_{M_{\varphi(AP)}} \text{ is the tree obtained by applying function } \tau \text{ to every position in } w, \text{ such that for any position } j \text{ in } w \text{ (}T_w, \tau(i')) \models \varphi \iff (w, i') \models \iota_X(\varphi), \text{ and likewise for } \psi. \)

**Proof.** \([\Rightarrow] \) Suppose \(\equiv (\varphi, \psi) \text{ holds in } s = \tau(i). \) Then node \(r \text{ s.t. } rR_s \text{ has at least two children, and } \equiv (\varphi, \psi) \text{ is true because of a path starting in } t = \tau(j) \text{ s.t. } tR_s \text{ and ending in } v = \tau(k), \text{ s.t. } rR_v \text{ and } (T_w, v) \models \psi. \) We must distinguish between the following cases:

1. \(v \text{ is not the leftmost child of } r. \)
   
   (a) \(h < i. \) By construction, all nodes in the path correspond to positions \(j' \in w \text{ s.t. } \chi(h, j') \text{ and } h < j', \text{ so they form a YPHP. Hence, } \varphi' \circ H \psi' \text{ holds in } j, \) and (18) \((\circ H (\varphi' \circ H \psi')) \) holds in \(i. \)

   (b) \(h \geq i. \) In this case, \(s \text{ is the rightmost child of } r, \) and \(\chi(h, i). \) The path made of positions between \(k \) and \(j \) corresponding to nodes between \(v \) and \(t \) (included) form a YPHP. So \(\varphi' \circ H \psi' \text{ holds in } j, \text{ which is the rightmost position of any possible such YPHP: so } \neg \circ H \top \text{ also holds in } j. \) Hence, (19) \((\chi' (\neg \circ H \top \land \varphi' \circ H \psi')) \) holds in \(i. \)

2. \(v \text{ is the leftmost child of } r. \)
(a) $h < i$. In this case, $k = h + 1$ and $\psi'$ holds in $k$. So, $\bigcirc \psi'$ holds in $h$, and $\chi_F^\varepsilon(\bigcirc \psi')$ holds in $i$. Moreover, in all positions $j' \in w, k < j' < j$, corresponding to tree nodes, $\psi'$ holds. Such positions form a YPHP. So $\neg \bigcirc_H^\varepsilon(\bigcap S_H^\varepsilon \neg \psi')$ holds in $i$.

Note that this is also true if $s$ is the first right sibling of $v$. In conclusion, \ref{eq:20} holds in $i$.

(b) $h \geq i$. $\psi'$ holds in $k = h + 1$, so $\bigcirc \psi'$ holds in $h$. Since $\chi(h, i)$ and $h \geq i$, $\chi_F^\varepsilon(\bigcirc \psi')$ holds in $i$. Moreover, $\psi'$ holds in all children of $r$ except the first and last one, i.e. $\psi'$ holds in all positions $j'$ s.t. $\chi(h, j')$ and $h < j'$. So $\neg \chi_F^\varepsilon \neg \psi'$ holds in $h$, and \ref{eq:21} holds in $i$.

$[\Leftarrow]$ Suppose \ref{eq:13} ($\bigcirc_H^\varepsilon(\psi' S_H^\varepsilon \psi')$) holds in $i$. Then, there exists a position $h$ s.t. $\chi(h, i)$ and $h < i$, and a position $j < i$ s.t. $\chi(h, j)$ and $h < j$. Since $j \neq h + 1$ and $h < i$, the corresponding tree nodes are not the leftmost nor the rightmost one. So, this is case \ref{eq:1a} and $\Leftarrow (\varphi, \psi)$ holds in $s = \tau(i)$.

Suppose \ref{eq:19} ($\chi_F^\varepsilon(\bigcirc \psi' \bigcap \bigcirc H^\varepsilon \psi')$) holds in $i$. Then, there exists a position $h$ s.t. $\chi(h, i)$ and $h = i$. Moreover, at least a position $j'$ s.t. $\chi(h, j')$ and $h < j'$ exists. Let $j$ be the rightmost one, i.e. the only one in which $\neg \bigcirc H^\varepsilon \bigcirc H^\varepsilon$ holds. The corresponding tree node $i = \tau(j)$ is s.t. $\tau R_{\Rightarrow} s$, with $s = \tau(i)$. Since $\psi' S_H^\varepsilon \psi'$ holds in $j$, a YPHP starts from it, and $\psi$ and $\varphi$ hold in the tree nodes corresponding to, respectively, the first and all other positions in the path. This is case \ref{eq:2b} and $\Leftarrow (\varphi, \psi)$ holds in $s$.

Suppose \ref{eq:20} ($\chi_F^\varepsilon(\bigcirc \psi' \bigwedge \neg \bigcirc H^\varepsilon \psi')$) holds in $i$. Then, there exists a position $h$ s.t. $\chi(h, i)$ and $h < i$. $\psi'$ holds in $k = h + 1$, so $\psi$ holds in the leftmost child of $r = \tau(h)$. Moreover, $\psi'$ holds in all positions $j' < i$ s.t. $\chi(h, j')$ and $h < j'$, so $\varphi$ holds in all children of $r$ between $v = \tau(k)$ and $s = \tau(i)$, excluded. This corresponds to case \ref{eq:2a} and $\Leftarrow (\varphi, \psi)$ holds in $s$.

Finally, suppose \ref{eq:21} ($\chi_F^\varepsilon(\bigcirc \psi' \bigwedge \neg \chi_F^\varepsilon \neg \psi')$) holds in $i$. Then, there exists a position $h$ s.t. $\chi(h, i)$ and $h = i$. $\bigcirc \psi'$ holds in $h$, so $\psi$ holds in node $v = \tau(h + 1)$, which is the leftmost child of $r = \tau(h)$. Since $\neg \chi_F^\varepsilon \neg \psi'$ holds in $h$, $\psi'$ holds in all positions $j'$ s.t. $\chi(h, j')$ and $h < j$. So, $\psi$ holds in all children of $r$ except (possibly) the leftmost ($v$) and the rightmost ($s = \tau(i)$) ones. This is case \ref{eq:2b} and $\Leftarrow (\varphi, \psi)$ holds in $s$.

### C.3 POTL Translation of OPTL

As an alternative proof of Corollary \ref{eq:2} we provide a direct translation of OPTL into POTL. We define function $\nu$, which given an OPTL formula $\varphi$, yields a POTL formula $\nu(\varphi)$ such that, for any OP word $w$ and position $i$, we have $(w, i) \models \varphi$ iff $(w, i) \models \nu(\varphi)$. $\nu$ is defined as the identity for propositional operators. In the following, we use the abbreviations $\varphi' \equiv \nu(\varphi)$ and $\psi' \equiv \nu(\psi)$.

$$\nu(\bigcirc \varphi) \equiv \bigcirc < \varphi' \quad \nu(\neg \varphi) \equiv \neg \nu(\varphi) \quad \nu(\varphi \land \varphi') \equiv \nu(\varphi) \land \nu(\varphi') \quad \nu(\varphi \lor \varphi') \equiv \nu(\varphi) \lor \nu(\varphi')$$

The translation for LTL until and since is much more involved:

$$\nu(\varphi \mathcal{U} \psi) \equiv \psi' \lor (\varphi' \land \alpha(\psi')) \mathcal{U}^{\equiv} (\psi' \lor (\varphi' \land \beta(\varphi')))$$
where
\[
\begin{align*}
\alpha(\varphi') & \equiv \chi_F^{\bowtie} \top \implies \neg (\bowtie (\top U^{\bowtie} \neg \varphi') \lor \chi_P^{\bowtie} (\top U^{\bowtie} \neg \varphi')) \\
\beta(\varphi') & \equiv \chi_P^{\bowtie} \top \implies \neg (\bowtie H (\top U^{\bowtie} \neg \varphi') \lor \chi_P^{\bowtie} (\top U^{\bowtie} \neg \varphi'))
\end{align*}
\]

The main formula is the concatenation of two OP summary until operators. The first one only admits the \(\bowtie\) and \(\bowtie\) \(\bowtie\) PR: it can only go upwards in the chain structure. Whenever one of its paths contains the left context of a chain, either the path ends there or it continues with the right context of that chain. The second until only admits \(\bowtie\) and \(\bowtie\) \(\bowtie\). It can therefore go downwards in the chain structure, and whenever one of its paths contains a right chain context, it must contain the left context too.

When evaluated in the left context of a chain, subformula \(\alpha(\varphi)\) makes sure \(\varphi'\) holds in all positions of the body of the outermost (forward maximal) chain starting in that position (i.e. the one whose left and right contexts are in the \(\bowtie\) or \(\bowtie\) relation). Therefore, including it in the left side of the first until makes sure \(\varphi'\) holds in all chain bodies skipped by its paths.

Symmetrically, when evaluated in the right context of a chain, \(\beta(\varphi)\) makes sure \(\varphi'\) holds in all positions in the body of the outermost (backward-maximal) chain ending there. It is included in both sides of the second until, so that \(\varphi'\) holds in all chain bodies skipped by its paths.

The translation for the since operator is symmetric.

The translations of the summary operators changes depending on the allowed PR. The main difference between the semantics of OPS until in OPTL and POTL is that, in the former, PR are checked only on consecutive positions, and the path can follow all “maximal” chains, whose contexts are in the \(\bowtie\) or \(\bowtie\) relation. In POTL, the allowed PR must holds between all positions consecutive in the path, including contexts of the same chain, and the \(\bowtie\) relation is also allowed in this case. Since maximal chains have their contexts in the \(\bowtie\) or \(\bowtie\) relations, we have

\[
\nu(\varphi U^{\bowtie} \psi) \equiv \varphi' U^{\bowtie} \psi'.
\]

When only one of such relation is allowed, we must prevent the path from spanning consecutive positions in the wrong relation.

\[
\nu(\varphi U^{\bowtie} \psi) \equiv (\varphi' \land \neg \bowtie U \top) U^{\bowtie} \psi'
\]
\[
\nu(\varphi U^{\bowtie} \psi) \equiv (\varphi' \land \neg \bowtie U \top) U^{\bowtie} \psi'
\]

Things become more complicated when the \(\bowtie\) relation is also allowed.

\[
\nu(\varphi U^{\bowtie} \psi) \equiv \varphi' U^{\bowtie} [\psi' \lor (\varphi' \land \gamma(\varphi'))] U^{\bowtie} \psi'
\]
\[
\gamma(\varphi') \equiv \chi_F^{\bowtie} \top \implies [\bowtie H (\varphi' \land \square \varphi') \land \chi_F^{\bowtie} \bowtie \bowtie \varphi']
\]

where \(\bowtie H \theta \equiv \neg [\top U^{\bowtie} (\neg \theta)]\), and \(\square \theta \equiv \neg [\top U^{\bowtie} (\neg \theta)]\).

In this case, we must make up for the fact that OPSP in POTL can skip bodies of chain whose contexts are in the \(\bowtie\) relation, while OPTL cannot. In such cases, OPSP in OPTL continue by following the successor edge. So, we split an OPSP in a path that goes only upwards in the abstract syntax tree (the first until with \(\bowtie\) \(\bowtie\)), followed by one
that can go downwards. In the latter, \( \gamma(\varphi') \) must also hold. Let \( i \) be a position in an OPSP, and let \( j_p, 1 \leq p \leq n \), be all positions such that \( \chi(i, j_p) \) and \( i \ll j_p \). Suppose \( j_q, 1 \leq q \leq n \), is also part of the OPSP. Formula \( \gamma(\varphi') \), if evaluated in \( i \), enforces \( \varphi' \) to hold in all positions that the OPTL OPSP would span between \( i \) and \( j_q \).

Formally, let \( i \) be a position in an OPSP, and let \( j_p, 1 \leq p \leq n \), be all positions such that \( \chi(i, j_p) \) and \( i \ll j_p \). Suppose \( j_q, 1 \leq q \leq n \), is also part of the OPSP. Formula \( \gamma(\varphi') \), if evaluated in \( i \), enforces \( \varphi' \) to hold in all positions that the OPTL OPSP would span between \( i \) and \( j_q \).

For other PR combinations containing \( \ll \), it suffices to forbid consecutive positions in the wrong relation. For \( \nu(\varphi \mathcal{U}^{<} \psi) \) and \( \nu(\varphi \mathcal{U}^{<>} \psi) \), just substitute \( \varphi' \) with, respectively, \( \varphi' \wedge \neg \circ \wedge \top \) and \( \varphi' \wedge \neg \circ \wedge \top \) in \( \nu(\varphi \mathcal{U}^{<>} \psi) \).

The translations for the take precedence hierarchical until and since are symmetric.

### D Omitted Proofs: Model Checking

In this section, we prove the correctness of the model checking OPA construction of Section 5, and we give the construction for operators omitted in the main text.

**Theorem 4 (Correctness of Finite Model Checking).** Given a finite set of atomic propositions \( AP \), an OP alphabet \( \mathcal{P}(AP, M_{P(AP)}) \), a word \( w \) on it, and an POTL formula \( \varphi \), the automaton built according to the procedure in this section is such that we have:

\[
(w, 1) \models \varphi
\]

if and only if it performs at least one accepting computation on a word \( w' \) equal to \( w \), except for the presence of one more propositional symbol for each hierarchical operator in \( \varphi \).

**Proof.** In the following, we prove that all chain next/back operators hold in a position in \( w \) iff the OPA performs at least a computation that, after reading a subword of \( w \), leaves the OPA in a state not containing any pending obligation related to that instance of the operator (cf. Lemmas 11, 12, 13, 14). While the correctness of the precedence next/back operators is trivial, that of OPS until/since operators is due to the correctness of the respective expansion laws, proved in Lemma 4. Moreover, in Lemma 15, we prove the correctness of the equivalences for the hierarchical operators.

The results above allow us to prove that, by structural induction on the syntax of \( \varphi \), if the latter holds in position 1 of \( w \), there exists a word \( w' \) identical to \( w \), except for the propositional symbols needed for the hierarchical operators, such that the OPA performs at least a computation reaching the end of \( w \) in a state containing no future operators and no temporal obligations. By the definition of the set of final states \( F \), such a computation is accepting.
Conversely, suppose there exists a word $w'$ with the described features on which the OPA performs at least one accepting computation starting from a state containing $\varphi$. Then $\varphi$ holds in the first position of a word $w$ built by removing the propositional symbols introduced by equivalence formulas for hierarchical operators. Indeed, such a computation ends with an empty stack, and a state containing no future operators or temporal obligations which, by the lemmas listed above, implies all temporal obligations have been satisfied, and $w$ is a model for $\varphi$.

D.1 Chain Next/Back Operators

![Diagram of Chain Next/Back Operators]

**Fig. 5.** The structure of a generic OP word. Solid lines connect terminals that are the context of a chain, wavy lines are a placeholder for a subtree. A few word positions are shown below the corresponding terminals ($i$, $j$, ...). We have either $a \doteq d$ or $a \triangleright d$, and $a \triangleleft b_k$ for $1 \leq k \leq n$. For $1 \leq k \leq n$, either $b_k[u_k]\doteq^{b_{k+1}}$, or $u_k$ is of the form $v^k_0\doteq c^k_0 v^k_1 \cdots c^k_{m_k} v^k_{m_k}$, where $c^k_p = c^k_{p+1}$ for $0 \leq p < m_k$, $b_k = c^k_0$, and resp. $c^k_{m_k} > b_{k+1}$ and $v^k_{m_k} > d$ (cf. Fig. 6). Moreover, for each $0 \leq p < m_k$, either $v^k_{p+1} = \varepsilon$ or $c^k_p[v^k_{p+1}]c^k_{p+1}$; either $v^k_0 = \varepsilon$ or $b_k[v^k_0]\varepsilon$, and either $v^k_{m_k+1} = \varepsilon$ or $c^k_{m_k} \varepsilon^{b_{k+1}}$. $u_0$ has this latter form, except $v^0_0 = \varepsilon$ and $a < c^0_0$. The $\pi$ symbols are placeholders for PR, and they vary depending on the surrounding terminal characters.

**Lemma 11.** Given a finite set of atomic propositions $\mathcal{AP}$, an OP alphabet $(\mathcal{P}(\mathcal{AP}), M_{\mathcal{P}(\mathcal{AP})})$, a word $w = \#xyz\#$ on it, and a position $i = |x| + 1$ in $w$, we have

$$(w, i) \models \lambda_F \psi$$
if and only if the automaton built for $\chi_F^w \psi$ performs at least one computation that brings it from configuration $\langle yz, \Phi, a\gamma \rangle$ with $\chi_F^z \psi \in \Phi_c$ to a configuration $\langle z, \Phi^d, a'\gamma \rangle$ such that $\chi_F^z \psi \notin \Phi^d_p$, $|a| = 1$ and $|a'| = 1$ if first(y) is read by a shift move, $|a'| = 2$ if it is read by a push move.

Proof. In the following, we denote by $\Phi^a$ the state of the automaton before reading symbol $a$, so $\Phi^a \cap AP = a$, for any $a \subseteq AP$.

$[\Rightarrow]$ Suppose $\chi_F^w \psi$ holds in position $i$, corresponding to terminal symbol $a$. There exists a computation in which the OPA reaches configuration $\langle a \ldots z, \Phi^a, [f, \Phi^f] y \rangle$, where $\alpha = [f, \Phi^f]$, and $\chi_F^w \psi \in \Phi^a_c$. $a$ is read by either a push or a shift transition, leading the OPA to configuration $\langle c_0^0 \ldots z, \Phi^c_{c_0^0}, \delta \rangle$, with either $\delta = [a, \Phi^a][f, \Phi^f] y$ or $\delta = [a, \Phi^f] y$, respectively. Moreover, $\chi_F^w \psi \in \Phi^c_{c_0^0}$ and $\chi_L \in \Phi^c_{c_0^0}$ due to rule (1). Since $\chi_F^w \psi$ holds in $i$, $a$ is the left context of a chain, so the next transition is a push, satisfying the requirement for $\chi_L$. This also means $w$ has the form of Fig. 5, possibly with $n = 0$ (cf. the caption for notation). The new configuration is $\langle v_0^1 \ldots z, \Phi^v_{v_0^1}, [c_0^0, \Phi^c_{c_0^0}][f, \Phi^f] \delta \rangle$, with $\chi_F^w \psi \in \Phi^c_{c_0^0}$. Then, the computation goes on normally. Note that, when reading an inner chain body such as $v_0^1$, the automaton does not touch the stack containing $\Phi^c_{c_0^0}$, and other symbols in the body of the same simple chain, i.e. $c_0^1, c_0^2, \ldots$, are read with shift moves that update the topmost stack symbol with the new terminal, leaving state $\Phi^c_{c_0^0}$ untouched.

If $a$ is the left context of more than one chain (i.e. $n > 0$ in the figure), the OPA then reaches configuration $\langle b_1 \ldots z, \Phi^{b_1}, [c_0^0, \Phi^c_{c_0^0}][f, \Phi^f] \delta \rangle$. Since $c_0^0 > b_1$, the next transition is a pop. $\chi_F^w \psi \in \Phi^c_{c_0^0}$, so by rule (2), the automaton reaches configuration $\langle b_1 \ldots z, \Phi^{b_1}, \delta \rangle$ with $\chi_F^w \psi \in \Phi^{b_1}_{b_1}$. Then, since $a$ is contained in the topmost stack symbol and $a < b_1$, the next move is a push, leading to $\langle v_0^1 \ldots z, \Phi^v_{v_0^1}, [b_1, \Phi^{b_1}][\delta] \rangle$. Notice how $\chi_F^w \psi$ is again stored as a pending obligation in the topmost stack symbol. The OPA runs on in the same way for each terminal $b_p$, $1 \leq p \leq n$, until the automaton reaches configuration $\langle d \ldots z, \Phi^{d}, [c_{n-1}^m, \Phi^{c_{n-1}^m}][f, \Phi^f] \delta \rangle$ with $\chi_F^w \psi \in \Phi^{b_p}_{b_p}$. If $a$ was the left context of one chain only, this is the configuration reached after reading the body of such chain, with $n = 0$. Since $c_{n-1}^m \geq d$, a pop transition leads to $\langle d \ldots z, \Phi^{d}, \delta \rangle$, with $\chi_F^w \psi \in \Phi^{d}_{b_p}$, by rule (2). Note that there exists a computation in which $\chi_F^w \psi \notin \Phi^{d}_{b_p}$, so rule (2) applies. The fact that a computation with $\chi_F^w \psi \notin \Phi^{d}_{b_p}$ is blocked by rule (2) is correct, because this implies $\chi_F^w \psi$ holds in the position preceding $d$. This is wrong, because such a position is in the
relation with \( d \), and it cannot be the left context of a chain, so \( \chi_F^\psi \) must be false in it. Then, if \( \chi_F^\psi \) holds in \( i \), since \( a \) is the terminal in the topmost stack symbol, we must have \( a \) is the terminal in the topmost stack symbol, we must have \( a = \gamma \). So \( d \) is read by a push move, leading to \( \langle z, \Phi^d, \alpha', \gamma \rangle \) with \( \alpha' = [d, \Phi^d][f, \Phi^f] \) or \( \alpha' = [d, \Phi^d] \), depending on which kind of move previously read \( a \). Since \( \chi_F^\psi \) holds in \( i, \psi \) holds in \( j \) (the position corresponding to \( d \)), and \( \psi \in \Phi^c, \) satisfying rule \( 3 \). Finally, there exists a computation in which \( \chi_F^\psi \not\in \Phi^c \), satisfying the thesis statement.

\[ \Rightarrow \] Suppose the OPA starts from configuration \( \langle a \ldots z, \Phi^a, [f, \Phi^f] \rangle \), with \( \chi_F^\psi \in \Phi^c, a = [f, \Phi^f] \), and \( f < a \) (the case \( f \leq a \) is analogous). \( a \) is read by a push move in this case, which leads the OPA to configuration \( \langle c_0 \ldots z, \Phi^c, [a, \Phi^a][f, \Phi^f] \rangle \), with \( \chi_F^\psi, \chi_L \in \Phi^c \). Since \( \chi_L \in \Phi^c \), the next transition must be a push, so \( a = c_0 \), \( a \) is the left context of a chain and \( w \) is of the form of Fig. 5. The push move brings the OPA to configuration \( \langle c_0 \ldots z, \Phi^c, [a, \Phi^a][f, \Phi^f] \rangle \). Notice that the stack size is now \( |y| + 3 \). By the thesis, the automaton eventually reaches a configuration in which the stack size is \( |y| + 2 \). This can be achieved if \( [c_0, \Phi^c] \) is popped, so \( \alpha' = [a, \Phi^a][f, \Phi^f] \). In a generic word such as the one of Fig. 5 this happens only before reading \( b_p \), \( 1 \leq i \leq n \), or \( d \).

In both cases, let \( [c_{m_k}, b_{n_k}] \) be the popped stack symbol. We have \( \chi_F^\psi \in \Phi^b \). By rule \( 3 \), if \( \Phi^d \) is the destination state of the pop move, \( \chi_F^\psi \in \Phi^d \), which does not satisfy the thesis statement. If the next move is a push (such as when reading any \( b_p \), \( 1 \leq p \leq n \)), the stack length increases again, which also does not satisfy the thesis. If the next move is a pop, rule \( 3 \) blocks the computation. So, the next move must be a shift, updating symbol \( [a, \Phi^a] \) to \( [d, \Phi^d] \), where \( d \) is the just-read terminal symbol. This means the OPA reached the right context of the chain whose left context is \( i \) (i.e. \( a \)), and the two positions are in the \( \approx \) relation. By rule \( 3 \), \( \psi \) is part of the starting state of this move, so \( \psi \) holds in this position, satisfying \( \chi_F^\psi \) in \( i \). The state resulting from the shift move may not contain \( \chi_F^\psi \) as a pending obligation, thus satisfying the thesis.

The proof for the \( \chi_F^\psi \) operator changes only in the fact that \( \psi \) is enforced by pop transitions, instead of shifts. It is therefore omitted.

**Lemma 12.** Given a finite set of atomic propositions \( AP \), an OP alphabet \( \langle \mathcal{P}(AP), \mathcal{M}_{AP} \rangle \), a word \( w = \#xyz\# \) on it, and a position \( i = |x| + 1 \) in \( w \), we have

\[
(w, i) \models \chi_F^\psi
\]

if and only if the automaton built for \( \chi_F^\psi \) performs at least one computation that brings it from configuration \( \langle yz, \Phi, \alpha, \gamma \rangle \) with \( \chi_F^\psi \in \Phi^c \) to a configuration \( \langle z, \Phi', \alpha', \gamma \rangle \) such that \( \chi_F^\psi \not\in \Phi^c \), \( |\alpha| = 1 \) and \( |\alpha'| = 2 \) if first(\( y \)) is read by a shift move, \( |\alpha'| = 2 \) if it is read by a push move.

**Proof.** \( \Rightarrow \) Suppose \( \chi_F^\psi \) holds in position \( i \), corresponding to terminal \( a \). Then, \( a \) must be the left context of more than one chain, and the word being read is of the form of Fig. 5 with \( n \geq 1 \). Let us call \( b_p \), \( 1 \leq p \leq n \), the right contexts of those of these chains that are s.t. \( a < b_p \). There exists an index \( q \), \( 1 \leq q \leq n \), such that \( \psi \) holds in \( i_{b_q} \), the word position labeled with \( b_q \). Then, there exists a computation that reaches a configuration \( \langle a \ldots z, \Phi^a, [f, \Phi^f] \rangle \), where \( a = [f, \Phi^f] \) and \( \chi_F^\psi \in \Phi^c \). \( a \) is read by a
shift or a push transition, which leads the OPA to configuration \( \langle c_0^0 \ldots z, \Phi_F^0, \delta \rangle \), with \( \delta = \alpha'\gamma \), and either \( \alpha' = [a, \Phi^a][f, \Phi^f] \) or \( \alpha' = [a, \Phi^f] \), respectively. Due to rule (1), we have \( \chi_F^e \psi \in \Phi_F^0 \) and \( \chi_L \in \Phi_F^0 \). As a result, the next move must be a push, consistently with the hypothesis implying \( a \) is the left context of a chain. Then, starting with \( c_0^0 \), the OPA reads the body of the innermost chain whose left context is \( a \), until it reaches its right context \( b_1 \). In this process, the topmost stack symbol \( [c_m^0, \Phi^0] \) may be updated by shift transitions reading other terminals \( c_p^0 \), \( 1 \leq p \leq m_0 \), that are part of the same simple chain as \( c_0^0 \). However, it is never popped until \( b_1 \) is reached, since subchains cause the OPA to only push, pop and update new stack symbols, but not existing ones. So, the OPA reaches configuration \( \langle b_1 \ldots z, \Phi^{b_1}, [c_m^0, \Phi^0] \rangle \), with \( \chi_F^e \psi \in \Phi_F^0 \).

Suppose \( q \neq 1 \), so \( \psi \) does not hold in \( b_1 \). Since \( c_m^0 > b_1 \), the next transition is a pop. Due to rule (2), it leads the OPA to configuration \( \langle b_1 \ldots z, \Phi^{b_1}, \delta \rangle \) with \( \chi_F^e \psi \in \Phi_F^{b_1} \) and \( \chi_L \in \Phi_F^{b_1} \). The presence of \( \chi_L \) implies the next move is a push, a requirement that is satisfied because \( a < b_1 \). So, the OPA transitions to configuration \( \langle v_0^1 \ldots z, \Phi^v_1, [b_1, \Phi^{b_1}] \rangle \). The computation, then, goes on in the same way for each \( b_p \), \( 1 \leq p < q \). Before \( b_q \) is read, (and possibly \( q = 1 \)), the OPA is in configuration \( \langle b_q \ldots z, \Phi_{b_q}, [c_{m_q}^{-1}, \Phi_{b_q}] \rangle \), with \( \chi_F^e \psi \in \Phi_F^{b_q} \). Since \( c_{m_q}^{-1} > b_q \), a pop transition brings the OPA to \( \langle b_q \ldots z, \Phi^{b_q}, \delta \rangle \). Since by hypothesis \( \psi \in \Phi_F^{b_q} \), by rule (2) we just have \( \chi_L \in \Phi_F^{b_q} \). Since the topmost stack symbol contains \( a \), and \( a < b_q \), the next transition is a push, which satisfies the requirement of \( \chi_L \). Note that \( \chi_F^e \psi \notin \Phi_F^{b_q} \), and the current stack is \( \delta \), which satisfies the thesis statement.

\[ [\Leftarrow] \text{Suppose the automaton reaches configuration } \langle a \ldots z, \Phi^a, [f, \Phi^f] \gamma \rangle \text{, with } \chi_F^e \psi \in \Phi_F^e \text{. Again, } a \text{ must be read by either a push or a shift move. Since } \chi_L \text{ is inserted as a pending requirement into the state resulting from this move, the next transition must be a push, so } a \text{ is the left context of at least a chain. This chain has the form of Fig. 5. By rule (1), the OPA reaches configuration } \langle c_0^0 \ldots z, \Phi_F^0, \delta \rangle \text{, with } \chi_F^e \psi, \chi_L \in \Phi_F^0 \text{, and } \delta \text{ as in the } [\Rightarrow] \text{ part after reading } \gamma \text{. Let } [c_0^0, \Phi^0] \text{ be the stack symbol pushed with } c_0^0 \text{. The stack size at this time is greater by one w.r.t. what is required by the thesis statement, so } [c_0^0, \Phi^0] \text{ must be popped.} \]

This happens when the OPA reaches a symbol \( e \) s.t. the terminal symbol in the topmost stack symbol takes precedence from \( e \). \( e \) must be s.t. \( a < e \) (and \( e = b_1 \) in Fig. 5). Otherwise, suppose by contradiction that \( a > e \) or \( a = e \) (so \( e = d \) in Fig. 5) in which \( n = 0 \) and \( c_m^0 \) precedes \( d \). In this case, after popping \([c_m^0, \Phi^0]\), the automaton reaches configuration \( \langle dz, \Phi^{ad}, \delta' \rangle \). Since \( \chi_F^e \psi \in \Phi_F^{ad} \), by rule (2) we have \( \chi_F^e \psi \in \Phi_F^{ad} \), so this configuration does not satisfy the thesis statement. Moreover, \( \chi_L \in \Phi_F^{ad} \), which requires the next transition to be a push. But \( a \preceq d \) or \( a > d \), and \( a \) is the topmost stack symbol, so such a computation is blocked by \( \chi_L \), never reaching a configuration complying with the thesis statement.

So, \( e = b_1 \), and the OPA reaches configuration \( \langle b_1 \ldots z, \Phi^{b_1}, [c_m^0, \Phi^0] \rangle \). The subsequent pop move leads to \( \langle b_1 \ldots z, \Phi^{b_1}, \delta \rangle \). Suppose \( \psi \in \Phi_F^{b_1} \). Then, by rule (2) we only have \( \chi_L \in \Phi_F^{b_1} \), and \( \chi_F^e \psi \notin \Phi_F^{b_1} \). This configuration satisfies the thesis statement,
and since \(a < b_1\), \(a\) and \(b_1\) are the context of a chain, and \(\psi\) holds in \(b_1\), we can conclude that \(\chi^p_\varphi \psi\) holds in \(a\).

Otherwise, if \(\psi \notin \Phi^b_\varphi\), by rule (2) we have \(\chi^p_\varphi \psi, \chi^L \in \Phi^b_\varphi\). The next transition will therefore push the symbol \([b_1, \Phi^b_\varphi]\) onto the stack, again with \(\chi^p_\varphi \psi\) as a pending obligation in it. Then, the same reasoning done with \([c'_0, \Phi^c_0]\) (and its subsequent updates) can be repeated. The only way the thesis statement can be satisfied is by reading a position \(b_q\), s.t. \(a < b_q\), the terminal in the topmost stack symbol takes precedence from \(b_q\) (so \(a\) and \(b_q\) are the context of a chain), and \(\psi \in \Phi^b_{b_q}\), so \(\psi\) holds in \(b_q\). This implies \(\chi^p_\varphi \psi\) holds in \(a\).

We now give the construction for the chain back operators, and their proofs.

We need symbol \(\chi_R\), which lets the computation go on only if the previous transition was a pop, and hence the position associated with the current state is the right context of a chain. So, for any \((\Phi, a, \Psi) \in \delta_{\text{push/shift}}\), we have \(\chi_R \notin \Psi_p\).

We can now list the constraints on the transition relation for the \(\chi^p_\varphi \psi\) operator.

1. Let \((\Phi, a, \Psi) \in \delta_{\text{shift}}\): then \(\chi^p_\varphi \psi \in \Phi_c\) iff \(\chi^p_\varphi \psi, \chi_R \in \Phi_p\);
2. Let \((\Phi, a, \Psi) \in \delta_{\text{push}}\): then \(\chi^p_\varphi \psi \notin \Phi_c\);
3. Let \((\Phi, \Theta, \Psi) \in \delta_{\text{pop}}\): then \(\chi^p_\varphi \psi \in \Psi_p\) iff \(\chi^p_\varphi \psi \in \Theta_p\);
4. Let \((\Phi, a, \Psi) \in \delta_{\text{push/shift}}\): then \(\chi^p_\varphi \psi \in \Psi_p\) iff \(\psi \in \Phi_c\).

The constraints for \(\chi^p_\varphi \psi\) now follow.

1. Let \((\Phi, a, \Psi) \in \delta_{\text{push}}\): then \(\chi^p_\varphi \psi \in \Phi_c\) iff \(\chi^p_\varphi \psi, \chi_R \in \Phi_p\);
2. Let \((\Phi, a, \Psi) \in \delta_{\text{shift}}\): then \(\chi^p_\varphi \psi \notin \Phi_c\);
3. Let \((\Phi, \Theta, \Psi) \in \delta_{\text{pop}}\): then \(\chi^p_\varphi \psi \in \Psi_p\) iff \(\chi^p_\varphi \psi \in \Theta_p\);
4. Let \((\Phi, a, \Psi) \in \delta_{\text{push/shift}}\): then \(\chi^p_\varphi \psi \in \Psi_p\) iff \(\psi \in \Phi_c\).

Finally, we give the constraints for \(\chi^p_\varphi \psi\) below.

Let \((\Phi, a, \Psi) \in \delta_{\text{push/shift}}\):

1. \(\chi^p_\varphi \psi \notin \Psi_p\);
2. \(\chi^p_\varphi \psi \in \Phi_c\) iff \(\chi^p_\varphi \psi, \chi_R \in \Phi_p\);
3. then \(\chi^p_\varphi \psi, \chi_R \in \Psi_p\) implies \(\chi^p_\varphi \psi \in \Phi_p\);
4. \(\chi^p_\varphi \psi \in \Psi_p\) and \(\chi_R \notin \Psi_p\) iff either \(\chi^p_\varphi \psi \notin \Theta_c\) or \(\chi^p_\varphi \psi \in \Phi_p\).

**Lemma 13.** Given a finite set of atomic propositions \(AP\), an OP alphabet \((P(AP), M_{\Phi(AP)})\), a word \(w = #xyz#\) on it, and a position \(j = |xy|\) in \(w\), we have

\[
(w, j) \models \chi^p_\varphi \psi
\]

if and only if the automaton built for \(\chi^p_\varphi \psi\) performs at least one computation that brings it from configuration \(\langle yz, \Phi, \alpha y \rangle\) to a configuration \(\langle z, \Phi', \alpha' y \rangle\) such that \(|\alpha| = 1\), \(|\alpha'| = 1\) if first \(y\) is read by a shift move, \(|\alpha'| = 2\) if it is read by a push move, and \(\chi^p_\varphi \psi \in \Phi'_c\), where \(\Phi'_c\) is the state of the OPA before reading \(j\), the last position of \(y\).
Proof. \[\Rightarrow\] Suppose \(\chi^p_\psi\) holds in position \(i\), corresponding to terminal symbol \(a\). Then, there exists a position \(j\), labeled with terminal \(d\), s.t. \(\chi(i, j), a = d\), and \(\psi\) holds in \(i\). Since \(a\) and \(d\) are the context of a chain, \(\psi\) must be encountered, and the automaton must reach configuration \(\langle c, \Phi^a, [f, \Phi^d] \rangle\). By the inductive assumption, we have \(\psi \in \Phi^a\). \(a\) is read by a shift or push move, bringing the OPA to \(\langle \varepsilon_0 \ldots v, \Phi^0, \delta \rangle\), with \(\delta = a'\gamma\), and either \(a' = [a, \Phi^a][f, \Phi^d]\) or \(a' = [a, \Phi^a]\), respectively. Due to rule (4), we have \(\chi^p_\psi \in \Phi^0\). After reading \(c_0\), the OPA reaches configuration \(\langle \varepsilon_0 \ldots z, \Phi^0, [c_0, \Phi^0] \rangle\). Then, the automaton proceeds to read the rest of the body of chain \(\chi(i, j)\). If \(i\) is the left context of multiple chains, the stack symbol \([c_0, \Phi^0]\), containing \(\chi^p_\psi\) as a pending obligation, is popped before reaching \(d\). Let \(b_p, 1 \leq p \leq n\), be all labels of positions \(i_{b_p}\) s.t. \(\chi(i, i_{b_p})\) and \(a < b_p\). It can be proved inductively that, before reading any of such positions, the OPA is in a configuration \(\langle b_p \ldots z, \Phi^{b_p}[c_{m_{p-1}}, \Phi^{b_{p-1}}] \rangle\), with \(\chi^p_\psi \in \Phi^{b_{p-1}}\). Since \(c_{m_{p-1}} = b_p\), the next move is a pop, leading to a configuration \(\langle b_p \ldots z, \Phi^{b_{p-1}} \rangle\), with \(\chi^p_\psi \in \Phi^{b_{p-1}}\), due to rule (5). Then, \(b_p\) is read by a push move because \(a < b_p\), so \(\chi^p_\psi\) is again stored in the topmost stack symbol as a pending obligation, in a configuration \(\langle v_0 \ldots z, \Phi^{v_0}[b_p, \Phi^{b_{p-1}}] \rangle\). The stack symbol containing \(\chi^p_\psi\) is only popped in positions \(b_p\), or when reaching \(d\), since subchains only cause the OPA to push and pop new symbols.

So, configuration \(\langle d_z, \Phi^d, [c, \Phi^d] \rangle\) is reached, with \(\chi^p_\psi \in \Phi^{c_{b_n}}\) (note that \(d\) labels the last position of \(y\)). Due to rule (3), a pop move leads the OPA to \(\langle d_z, \Phi^{d, \delta} \rangle\), with \(\chi^p_\psi \in \Phi^{d, \delta}\). Then, since by hypothesis \(a = d\), and \(a\) is contained in the topmost stack symbol, \(d\) is read by a shift move. Since this transition is preceded by a pop, we have a computation in which \(\chi^p_\psi = \Phi^{d, \delta}\). So, by rule (1), since \(\chi^p_\psi, \chi^p_\psi = \Phi^{d, \delta}\), we have \(\chi^p_\psi = \Phi^{d, \delta}\), with the stack equal to \(\delta\), satisfying the thesis statement.

[\(\Leftarrow\)] Suppose that, while reading \(w\), a computation of the OPA arrives at a configuration \(\langle d_z, \Phi^d, \delta \rangle\), where \(d\) is the last character of \(y\), and \(\chi^p_\psi \in \Phi^{d, \delta}\). By rule (1), we have \(\chi^p_\psi, \chi^p_R = \Phi^{d, \delta}\). \(\chi^p_R \in \Phi^{d, \delta}\) requires the previous transition to be a pop, so \(d\) is the right context of a chain. Let \(a\) be its left context. By hypothesis, the computation proceeds reading \(d\), and by rule (2) it must be read by a shift transition. So, we have \(a = d\), and \(w\) must be of the form of Fig. 5. Going back to \(\langle d_z, \Phi^d, \delta \rangle\), consider the pop move leading to this configuration. It starts from configuration \(\langle d_z, \Phi^d, [c_{m_{n}}, \Phi^{b_n}] \delta \rangle\), and by rule (5) we have \(\chi^p_\psi = \Phi^{b_n}\).

Consider the move that pushed \(\Phi^{b_n}\) onto the stack. Suppose it was preceded by a pop move. Since \(\Phi^{b_n}\) is the target state of this transition, and \(\chi^p_\psi \in \Phi^{b_n}\), by rule (3) \(\chi^p_\psi\) must be contained as a pending obligation in the popped state as well. So, this obligation is propagated backwards every time the automaton encounters a position that is the left context of a chain, i.e. positions \(b_p\), \(1 \leq p \leq n\), in Fig. 5. In order to stop the propagation, a push of a state with \(\chi^p_\psi\) as a pending obligation, preceded by another push or shift move must be encountered. Such a transition pushes or updates the stack symbol under the one containing \(\chi^p_\psi\), which means the left context \(a\) s.t. \(a = d\) of a chain whose right context is \(d\) has been reached. In both cases, the target state of the push/shift transitions contains \(\chi^p_\psi\) as a pending obligation, so by rule (4) we have
ψ ∈ Φ_j. Hence, by the inductive assumption, ψ holds in position i (corresponding to a), we have i = j and χ(i, j), which implies χ_Pψ holds in j.

The proof of the model checking rules of χ_Pψ is similar to the one of Lemma 13 and is therefore omitted.

**Fig. 7.** The structure of a generic OP word. We have either a = d or a < d, and b_k > d for 1 ≤ k ≤ n. For 1 ≤ k ≤ n, we either have b_m+1[u_k]_0^m, or u_k is of the form v_k^0, v_k^1, ..., v_k^m, v_k^m+1, where c_p^k = c_p^k for 0 ≤ p < m_k, c_p^k = b_k, and resp. a < c_0 and b_{k+1} < c_0. Moreover, for each 0 ≤ p < m_k, either v_{p+1} = e or v_{p+1} = v_p^k, either v_{m_k+1} = e or v_{m_k+1} = b_k, and either v_{m_k+1} = e or b_{m_k+1} = e. a_0 has the same form, except v_{m_0} = e and v_{m_0} = d. The π symbols are placeholders for PR, and they vary depending on the surrounding terminal characters.

**Lemma 14.** Given a finite set of atomic propositions AP, an OP alphabet (P(AP), M_P(AP)), a word w = #xyz# on it, and a position j = |xy| in w, we have

\[(w, j) \models \chi_P^\psi\]

if and only if the automaton built for χ_P^\psi performs at least one computation that brings it from configuration ⟨yz, Φ, αγ⟩ to a configuration ⟨z, Φ', α'γ⟩ such that |α| = 1, |α'| = 1 if first(y) is read by a shift move, |α'| = 2 if it is read by a push move, and χ_P^\psi ∈ Φ', where Φ' is the state of the OPA before reading j, the last position of y.
Proof. \(\Rightarrow\) Suppose \(\chi_p^q \psi\) holds in position \(j\). Then, \(j\) is the right context of at least two chains, and the word \(w\) has the form of Fig. 7 with \(i\) being the left context of the outermost chain whose right context is \(j\). Let positions \(i_{bq}\), labeled with \(b_p, 1 \leq p \leq n\), be all other left contexts of chains sharing \(j\) as their right context. There exists a value \(q, 1 \leq q \leq n, \) s.t. \(\psi\) holds in \(i_{bq}\).

The OPA reads \(w\) normally, until it reaches \(b_q\), with configuration \(\langle b_q \ldots z, \Phi^q_b, [c^q_{ma}, \Phi^q_c]\[b_{q+1}, \Phi^{q+1}_c] \ldots \delta\rangle\), with \(\psi \in \Phi^q_c, \delta = \alpha'\gamma\), and either \(\alpha' = [a', \Phi^a][f, \Phi^f]\). if \(a\) (the label of \(i\)) was read by a push move, or \(\alpha' = [a, \Phi^a]\) if it was read by a shift. Note that if \(b_q\) is the only character in its rhs \((u_q = \varepsilon\) in Fig. 7\(] \[c^q_{ma}, \Phi^q_c]\) is not present on the stack. In this case, \(b_q\) is read by a push move instead of a shift. Suppose \(b_q\) is the left context of one or more chains, besides the one whose right context is \(j\). In Fig. 7\(] this means \(v^{q-1}_{0} \neq \varepsilon\). Consider the right context of the outermost of such chains: w.l.o.g., we call it \(v^{q-1}_0\) (it may as well be \(\psi^{q-1}_0\)). Since \(\psi\) holds in \(i_{bq}\), \(\chi_p^q \psi\) holds in \(v^{q-1}_0\). If, instead, \(v^{q-1}_0 = \varepsilon\), then \(v^{q-1}_0\) is the successor of \(b_q\), and \(\psi\) holds in it. In both cases, \(\chi_p^q \psi \lor \psi\) holds in \(v^{q-1}_0\). Since \(b_q < c^{q-1}_0\), the latter is read by a push transition, pushing stack symbol \([c^{q-1}_0, \Phi^{q-1}_c]\), with \(\chi_p^q \psi \lor \psi\) \(\in \Phi^{q-1}_c\). This symbol remains on stack until \(d\) is reached, although its terminal symbol may be updated. The computation then proceeds normally, until configuration \(\langle dz, \Phi^{(q-2)d}, [b_{q-1}, \Phi^{(q-1)d}] \ldots \delta\rangle\) is reached.

Since \(\chi_p^q \psi \lor \psi\) \(\in \Phi^{q-1}_c\), by rule 4, the OPA transitions to configuration \(\langle dz, \Phi^{(q-1)d}, [b_q, \Phi^{q}_d] \ldots \delta\rangle\) with \(\chi_p^q \psi \in \Phi^{(q-1)d}\) and \(\chi_R \notin \Phi^{(q-1)d}\). (Note that \(\chi_R\) may not be contained in a state even if the previous transition is a pop.) Then, by rule 4, all subsequent pop transitions propagate \(\chi_p^q \psi\) as a pending obligation in the OPA state, until configuration \(\langle dz, \Phi^{(n-1)d}, \delta\rangle\), with \(\chi_p^q \psi \in \Phi^{(n-1)d}\). Now, at least one of the computations transitions to \(\langle dz, \Phi^{(n)d}, \delta\rangle\), with \(\chi_p^q \psi \in \Phi^{(n)d}\) and \(\chi_R \in \Phi^{(n)}d\), according to rule 3. At this point, \(d\) is read with either a shift or a push transition. So, according to rule 2, \(\chi_p^q \psi \in \Phi^{(n)d}_c\), which satisfies the thesis statement.

\(\Leftarrow\) Suppose the automaton reaches a state \(\Phi^d = \Phi^{(n)d}\) s.t. \(\chi_p^q \psi \in \Phi^d_c\). For the computation to continue, we have \(\chi_R \in \Phi^d_p\). So, the transition leading to state \(\Phi^d_p\) must be a pop, and the related word position \(d\) is the right context of a chain. Let \(\Phi^d_j\) be the starting state of this transition. Since \(\chi_p^q \psi \in \Phi^d_j\), by rule 3, we have \(\chi_p^q \psi \in \Phi^d_p\). By rule 1, this transition must be preceded by another pop, so \(d\) is the right context of at least two chains, and the word being read is of the form of Fig. 7 with \(n \geq 1\).

So, before reading \(d\), the OPA performs a pop transition for each inner chain having \(d\) as a right context, i.e. those having \(b_p, 1 \leq p \leq n\), as left contexts in Fig. 7 plus one for the outermost chain (whose left context is \(a\)). By rule 4, \(\chi_p^q \psi\) is propagated backwards through such transitions from the one before \(d\) is read, to one in which \(\chi_p^q \psi \lor \psi\) is contained into the popped state. Note that the model checking rules for the other operators only introduce \(\chi_R\) in the last pop before \(d\), so they do not interfere with the application of rule 4.

By rule 1, for the computation to reach such pop transitions, the propagation of \(\chi_p^q \psi\) as a pending obligation must stop. So, the OPA must reach a configuration \(\langle dz,
[b_q, \Phi^q, \ldots \delta] with \chi^q \psi \lor \Theta^q \psi \in \Phi^q. Note that the following reasoning also applies to the case in which, in Fig. 7, u_q = e, by substituting b_q to e^q. The topmost stack symbol was pushed after configuration \langle e^q, \ldots, \delta \rangle. We have \text{for } q_{q-1} < e^q. If v^q_0 = e, and e^q_0 is in the position next to b_{q-1}, \Theta^q \psi holds, while if v^q_0 \neq e, since \text{there exist two positions between } 0 \leq \text{for } \exists q \in \text{for } \forall \in \Phi^q n \in \text{for } \bigwedge \text{for } \bigvee \text{for } \Rightarrow \text{for } \bigcap \text{for } \bigodot 
exists \text{for } \exists \text{for } \bigcirc \text{for } \bigotimes \text{for } \bigoplus \text{for } \bigodot \text{for } \bigotimes
onumber

D.2 Hierarchical Operators

We only prove equivalence (24), as the others are essentially analogous.

Lemma 15 (Equivalence (24)). Let w be an OP word based on an alphabet of atomic propositions \mathcal{P}(AP), and i a position in w, and let q_{\psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\}} identical to w, except q_{\psi \in \mathcal{P}(AP)} holds in position h < i s.t. \chi(h, i) and \chi(h, i).

Then, (w, i) \models \psi \mathcal{U}^w_1 \theta iff (w', i) \models \Upsilon(\psi, \theta), with

\Upsilon(\psi, \theta) \equiv \gamma_{\psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\}} \wedge (\chi^w \psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\}) \Rightarrow \psi) \mathcal{U}^w_1 \theta \wedge (\chi^w \psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\}) \wedge (\theta).

Proof. \([\Rightarrow] Suppose \psi \mathcal{U}^w_1 \theta holds in position i in word w. Then, by its semantics, there exists a YPHP \text{ for each } i_p, 0 \leq p \leq n, with n \geq 0, and a position h < i s.t. \chi(h, i) holds in i, and \theta holds in i. We show that \text{for each } i, By construction, in w', q_{\psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\}} holds in h only. So, q_{\psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\}} \wedge (\chi^w \psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\}) \Rightarrow (\theta).

For (\chi^w \psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\})\Rightarrow (\psi) \mathcal{U}^w_1 \theta \wedge (\theta) to hold in i, there must exist an OPSP between i = i_0 and i_n, allowing only the \text{for each } i, r < s, either \text{for each } i, r < s, or \chi(r, s). s.t. no OPSP can skip them. So, there exist no positions \text{for each } i, i < i' \leq i < s < i' < i, s.t. \chi(r, s') and either \text{for each } i, i < s < i', s.t. \chi(r, s') and \text{for each } i, r < s, r is the left context of a chain. Let k be the maximal (i.e. rightmost) position s.t. \chi(r, k).

There are three cases:

- \text{for each } i, i < k, is part of the body of the chain \chi(r, k). However, by hypothesis, \chi(h, i), and \text{for each } i, i < k. These two chains cross each other, which is impossible by the definition of chain.

- k = i_n. If \text{for each } i, i < i, then \text{for each } i, \chi \text{ for each } i, and \text{for each } i, r < i, with \text{for each } i, which is impossible because of property (3) of Lemma 3.

- \text{for each } i, i < k. If \text{for each } i, i < k, then \text{for each } i, k can be part of an OPSP reaching i_n. If \text{for each } i, \text{for each } i, then \text{for each } i, k is the first position of the body of another chain having r as its left context, which contradicts the assumption that \text{for each } i, k is maximal.

By hypothesis, q_{\psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\}} holds in h, so \chi^w \psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\} holds in all positions \text{for each } i_p, 0 \leq p \leq n, in the \text{for each } i, \chi^w \psi \in \mathcal{P}(AP) \cup \{q_{\psi \in \mathcal{P}(AP)}\} \text{ for each } i_p. Moreover, since \psi holds in
all \( i_q \), \( 0 \leq q \leq n \). \( \mathcal{V}_\phi q_w U_{\eta}^\theta \) \( \implies \) \( \psi \) holds in all positions in the OPSP between \( i_0 \) and \( i_n \).

[\( \Leftarrow \)] Suppose \( (w', i) \models \Upsilon(\psi, \theta) \). Then, \( \gamma_L \psi U_{\eta}^\theta \) holds in \( i \). This implies there exists a position \( h \) s.t. \( \chi(h, i) \) and \( h < i \), which is unique by Lemma 5. By \( \gamma_L \psi U_{\eta}^\theta \), \( \mathcal{V}_\psi q_w U_{\theta}^\theta \) holds in \( h \) and in no other position. Moreover, \( (\mathcal{V}_\phi q_w U_{\eta}^\theta) \implies \psi) \mathcal{U}^{\triangleright} (\mathcal{V}_\phi q_w U_{\eta}^\theta) \land \theta) \) holds in \( i \), so there exists an OPSP \( i = j_0 < j_1 < \cdots < j_m \). We show that there exists a sequence of indices \( 0 = p_0 < p_1 < \cdots < p_n = m \) s.t. \( j_{p_0}, j_{p_1}, \ldots, j_{p_n} \) is a YPHP satisfying \( \psi U_{\eta}^\theta \) in \( i \) in \( w \).

First, note that \( \theta \) holds in \( j_{p_n} \), and since \( h \) is the only position in which \( q_w U_{\eta}^\theta \) holds, we have \( \chi(h, j_{p_n}) \) and \( h < j_{p_n} \). So, \( j_{p_n} \) is the last position of a YPHP starting in \( i \).

For each position \( j \) s.t. \( i < j < j_{p_n} \), \( \chi(h, j) \) and \( h < j \), there exists an index \( 1 \leq q \leq n - 1 \) s.t. \( j_p = j \). Since all such positions \( j \) are between \( j_0 \) and \( j_m \), the OPSP could skip them only if they were part of the body of a chain, i.e. if there exist two positions \( j_0 \leq r < s \leq j_m \) s.t. \( \chi(j_0, j_m) \) and either \( r = s \) or \( r > s \). Such a chain would, however, cross with \( \chi(h, j) \), which contradicts the definition of chain.

Because \( q_w U_{\eta}^\theta \) only holds in \( h \), the fact that \( \mathcal{V}_\phi q_w U_{\eta}^\theta \) \( \implies \psi \) holds in all positions \( j_0, j_1, \ldots, j_{m-1} \) implies \( \psi \) holds in all of \( j_{p_0}, j_{p_1}, \ldots, j_{p_n} \). So, by construction of \( w', j_{p_0}, j_{p_1}, \ldots, j_{p_n} \) is a YPHP satisfying \( \psi U_{\eta}^\theta \) in \( i \) in \( w \).

We now give the equivalences for \( \triangleright \)-hierarchical operators. The following formula, when evaluated in the left context of a chain, forces symbol \( p_{\eta} \) in the right context. Note that if the left context is in the \( \triangleright \) relation with the right one, the latter is uniquely identified.

\[
\gamma_{R, \eta} \equiv \chi_F (q_{\eta}) \land (\circ <^{\triangleright} (\square \neg q_{\eta})) \land (\circ <^{\triangleright} (\square \neg q_{\eta}))
\]

We give the following equivalences for the take precedence hierarchical operators.

\[
\circ^{\triangleright} H \psi \equiv \gamma_{R, \circ^{\triangleright} H} \psi \land (\neg \chi_F (q_{\circ^{\triangleright} H} \psi) \mathcal{U}^{<^{\triangleright}} (\chi_F (q_{\circ^{\triangleright} H} \psi) \land \psi)) \tag{26}
\]

\[
\circ^{\triangleright} H \psi \equiv \gamma_{R, \circ^{\triangleright} H} \psi \land (\neg \chi_F (q_{\circ^{\triangleright} H} \psi) \mathcal{S}^{<^{\triangleright}} (\chi_F (q_{\circ^{\triangleright} H} \psi) \land \psi)) \tag{27}
\]

\[
\psi U_{H}^{\triangleright} \theta \equiv \gamma_{R, \psi U_{H}^{\triangleright}} \theta \land (\chi_F (q_{\psi U_{H}^{\triangleright}} \theta) \implies \psi) \mathcal{U}^{<^{\triangleright}} (\chi_F (q_{\psi U_{H}^{\triangleright}} \theta) \land \theta) \tag{28}
\]

\[
\psi S_{H}^{<^{\triangleright}} \theta \equiv \gamma_{R, \psi S_{H}^{<^{\triangleright}} \theta} \land (\chi_F (q_{\psi S_{H}^{<^{\triangleright}} \theta}) \implies \psi) \mathcal{S}^{<^{\triangleright}} (\chi_F (q_{\psi S_{H}^{<^{\triangleright}} \theta}) \land \theta) \tag{29}
\]