

Chapter 1

Variable-free reasoning on finite trees

PATRICK BLACKBURN^{*}, BERTRAND GAIFFE[†], MAARTEN MARX[‡]

ABSTRACT.

In this paper we examine three modal languages that have been proposed in the model theoretic syntax literature for describing finite ordered trees. We compare their expressive power, and then examine a key complexity-theoretic issue: how expensive it is to decide — given a theory specifying a certain class of trees — whether a formula describes a model? Our main result is that for the languages proposed by Blackburn *et al.* and Palm this problem is EXPTIME-complete.

1.1 Introduction

Model theoretic syntax is an uncompromisingly declarative approach to natural language syntax: grammatical theories are logical theories, and grammatical structures are their models. Perhaps the best known work in this tradition is that of James Rogers (for example [1]) in which grammatical theories are stated in monadic second-order logic. However other authors (in particular [2], [3] and [4]) use various kinds of *modal logic* (in essence, variable free formalisms for describing relational structures) to specify grammatical constraints. [5] contains some interesting linguistic examples and is a good introduction to (and motivation for) this approach.

In this paper we examine the modal languages proposed by Kracht, Palm, and Blackburn *et al.* for describing models based on finite trees. We compare their expressive power, and then examine a key complexity-theoretic issue: how expensive it is to decide — given a theory specifying a certain class of trees — whether a formula describes a model? Our main result is that for the languages of Blackburn *et al.* and Palm this problem is complete for the class of problems solvable in exponential time.

^{*}Languag et Dialogue, LORIA, Nancy, France; patrick@aplog.org.

[†]Languag et Dialogue, LORIA, Nancy, France; gaiffe@loria.fr

[‡]ILLIC, Universiteit van Amsterdam, The Netherlands; marx@science.uva.nl

1.2 The Languages \mathcal{L}_B , \mathcal{L}_P and \mathcal{L}_K

We first recall the definitions of three modal languages proposed in the model-theoretic syntax literature for specifying declarative constraints on ordered trees. We start with the strongest, proposed by Marcus Kracht in [19]. The language will be called \mathcal{L}_K (K for Kracht).

\mathcal{L}_K is a propositional modal language identical to Propositional Dynamic Logic (PDL) [17] over four basic programs \leftarrow , \rightarrow , \uparrow and \downarrow , which explore the left-sister, right-sister, mother-of and daughter-of relations. Recall that PDL has two sorts of expressions: programs and propositions. We suppose we have fixed a non-empty, finite or countably infinite, set of atomic symbols A whose elements are typically denoted by p . \mathcal{L}_K 's syntax is as follows, writing π for programs and ϕ for propositions:

$$\begin{aligned}\pi &::= \leftarrow | \rightarrow | \uparrow | \downarrow | \pi; \pi | \pi \cup \pi | \pi^* | ?\phi \\ \phi &::= p | \top | \neg\phi | \phi \wedge \phi | \langle \pi \rangle \phi.\end{aligned}$$

We sometimes write $\mathcal{L}_K(A)$ to emphasize the dependence on A . We employ the usual boolean abbreviations and use $[\pi]\phi$ for $\neg\langle \pi \rangle\neg\phi$.

We interpret $\mathcal{L}_K(A)$ on *finite ordered trees* whose nodes are *labeled* with symbols drawn from A . We assume that the reader is familiar with finite trees and such concepts as ‘daughter-of’, ‘mother-of’, ‘sister-of’, ‘root-node’, ‘terminal-node’, and so on. If a node has no sister to the immediate right we call it a last node, and if it has no sister to the immediate left we call it a first node. Note that the root node is both first and last. The root node will always be called *root*. A labeling of a finite tree associates a subset of A with each tree node.

Formally, we present finite ordered trees as tuples $\mathbf{T} = (T, R_{\rightarrow}, R_{\downarrow})$. Here T is the set of tree nodes and R_{\rightarrow} and R_{\downarrow} are the right-sister and daughter-of relations respectively. A pair $\mathfrak{M} = (\mathbf{T}, V)$, where \mathbf{T} is a finite tree and $V : A \rightarrow \text{Pow}(T)$, is called a *model*, and we say that V is a *labeling function* or a *valuation*. Given a model \mathfrak{M} , we simultaneously define a set of relations on $T \times T$ and the interpretation of the language $\mathcal{L}_K(A)$ on \mathfrak{M} :

$$\begin{aligned}R_{\uparrow} &= R_{\downarrow}^{-1} & R_{\pi \cup \pi'} &= R_{\pi} \cup R_{\pi'} \\ R_{\leftarrow} &= R_{\rightarrow}^{-1} & R_{\pi; \pi'} &= R_{\pi} \circ R_{\pi'} \\ R_{\pi^*} &= R_{\pi}^* & R_{?\phi} &= \{(t, t) \mid \mathfrak{M}, t \models \phi\}.\end{aligned}$$

$$\begin{aligned}\mathfrak{M}, t \models p &\text{ iff } t \in V(p), \text{ for all } p \in A \\ \mathfrak{M}, t \models \top &\text{ iff } t \in T \\ \mathfrak{M}, t \models \neg\phi &\text{ iff } \mathfrak{M}, t \not\models \phi\end{aligned}$$

$$\begin{aligned} \mathfrak{M}, t \models \phi \wedge \psi & \text{ iff } \mathfrak{M}, t \models \phi \text{ and } \mathfrak{M}, t \models \psi \\ \mathfrak{M}, t \models \langle \pi \rangle \phi & \text{ iff } \exists t' (tR_{\pi}t' \text{ and } \mathfrak{M}, t' \models \phi). \end{aligned}$$

If $\mathfrak{M}, t \models \phi$, then we say ϕ is *satisfied* in \mathfrak{M} at t . For any formula ϕ , if there is a model \mathfrak{M} such that $\mathfrak{M}, \text{root} \models \phi$, then we say that ϕ is *satisfiable*. For Γ a set of formulas, and ϕ a formula, we say that ϕ is a consequence of Γ (denoted by $\Gamma \models \phi$) if for every model in which Γ is satisfied at every node, ϕ is also satisfied at every node.

Below are two examples of such formulas: (3.2.1) says that every a node has a b and a c daughter, in that order, and no other daughters; and (3.2.2) says that every a node has a b first daughter followed by some number of c daughters, and no other daughters.

$$(1.2.1)\alpha \rightarrow \langle \downarrow \rangle (\neg \langle \leftarrow \rangle \top \wedge b \wedge \langle \rightarrow \rangle (c \wedge \neg \langle \rightarrow \rangle \top))$$

$$(1.2.2)\alpha \rightarrow \langle \downarrow \rangle (\neg \langle \leftarrow \rangle \top \wedge b \wedge \langle (\rightarrow; ?c)^* \rangle \neg \langle \rightarrow \rangle \top).$$

A final remark. Note that we could have generated the same language by taking \downarrow and \rightarrow as primitive programs and closing the set of programs under converses.

Two more languages The two other languages proposed in the literature only differ from \mathcal{L}_K in the programs they allow.

The language proposed by Blackburn, Meyer–Viol and de Rijke (?), here called \mathcal{L}_B , is the weakest. It contains only the four basic programs plus their *transitive* closures, denoted by a superscript $(\cdot)^+$. This language is precisely as expressive as the language generated by the following programs:

$$\pi ::= \leftarrow \mid \rightarrow \mid \uparrow \mid \downarrow \mid \pi^*.$$

To see this, note that for $\pi \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$, the transitive closure operator is expressible by, $\langle \pi^+ \rangle \phi \equiv \langle \pi \rangle \langle \pi^* \rangle \phi$. For the other direction, note that $\langle (\pi^*)^* \rangle \phi \equiv \langle \pi^* \rangle \phi$ and $\langle \pi^* \rangle \phi \equiv \phi \vee \langle \pi^+ \rangle \phi$

The language proposed by ?, here called \mathcal{L}_P , lies between \mathcal{L}_B and \mathcal{L}_K with respect to expressive power. It is generated by the following programs¹:

$$\pi ::= \leftarrow \mid \rightarrow \mid \uparrow \mid \downarrow \mid ?\phi; \pi \mid \pi^*.$$

Palm tried to designed his language to have exactly the expressive power required to reason about syntactical structures. At first glance, \mathcal{L}_P seems rather weak compared with Kracht's language, for it lacks the composition, union and test operator

¹Palm's conditional paths π_{ϕ} are denoted here as $?\phi; \pi$.

constructors. However note that when these are applied outside of the scope of the Kleene star they are definable as follows: $\langle \pi; \pi' \rangle \phi \equiv \langle \pi \rangle \langle \pi' \rangle \phi$, $\langle \pi \cup \pi' \rangle \phi \equiv \langle \pi \rangle \phi \vee \langle \pi' \rangle \phi$, and $\langle ?\psi \rangle \phi \equiv \psi \wedge \phi$. Palm claims that “The resulting ‘tense’ fragment of PDL holds sufficient expressivity to handle the linguistic demands on tree constraints”.

Palm calls his language *Propositional Tense Logic for Finite Trees*, making an analogy with branching time logic. In branching time logic, $\langle \downarrow^* \rangle \phi$ and $\langle \uparrow^* \rangle \phi$ are called *sometimes in the future* ϕ and *sometimes in the past* ϕ , respectively. But besides these unary operators, branching time logic standardly makes use of the binary *until* and *since* connectives. *Until* is defined as: $\mathfrak{M}, t \models U(\phi, \psi)$ iff there exists a time t' in the future of t with $\mathfrak{M}, t' \models \phi$ and for all time points t'' in between t and t' it holds that $\mathfrak{M}, t'' \models \psi$. *Since* has an analogous definition, but toward the past.

In fact, Palm’s choice of the name *Tense Logic* is apt, for as we shall now see \mathcal{L}_P is nothing but the simplest language \mathcal{L}_B with four additional *until* operators defined as follows. For $\pi \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$, $\mathfrak{M}, t \models U_\pi(\phi, \psi)$ iff there exists a t' such that $tR_{\pi^*}t'$ and $\mathfrak{M}, t' \models \phi$ and for all t'' such that $tR_{\pi^*}t''R_\pi t'$ it holds that $\mathfrak{M}, t'' \models \psi$. $U_\pi(\phi, \psi)$ is a very natural operation. For instance, the program `while ϕ do π` is expressed by $U_\pi(\neg\phi, \phi)$.

Theorem 1.2.1. *The language \mathcal{L}_P is precisely as expressive as the language \mathcal{L}_B with the additional four until programs.*

Proof: For one direction, note that for $\pi \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$, $U_\pi(\phi, \psi) \equiv \langle \langle ?\psi; \pi \rangle^* \rangle \phi$, and the right hand side is a Palm formula. For the other direction, we use induction on the complexity of the programs in \mathcal{L}_P formulas. Inside this proof \mathcal{L}_{until} denotes the language \mathcal{L}_B with the additional four U_π programs.

Consider the formula $\langle \pi \rangle \phi$. There are three cases. If π is a basic program, then $\langle \pi \rangle \phi \in \mathcal{L}_{until}$. In the second case, π is of the form $? \psi; P$, for P a program. If P is a basic program, then $\langle \pi \rangle \phi = \langle ? \psi; P \rangle \phi$ is equivalent to $\psi \wedge \langle P \rangle \phi$ which is in \mathcal{L}_{until} . If P itself is of the form $? \theta; P'$, then $\langle \pi \rangle \phi = \langle ? \psi; ? \theta; P' \rangle \phi$ which is equivalent to $\langle ?(\psi \wedge \theta); P' \rangle \phi$. Now P' is of smaller complexity than P , whence by inductive hypothesis, the last formula is equivalent to a formula in \mathcal{L}_{until} . Finally if P is of the form Q^* , then $\langle \pi \rangle \phi = \langle ? \psi; Q^* \rangle \phi$ which is equivalent to $\psi \wedge \langle Q^* \rangle \phi$, which by IH then is equivalent to a formula in \mathcal{L}_{until} . In the third and last case, π is of the form P^* . If P is a basic program, $\langle \pi \rangle \phi \in \mathcal{L}_{until}$. If P itself is of the form Q^* , then $\langle \pi \rangle \phi = \langle \langle Q^* \rangle^* \rangle \phi \equiv \langle Q^* \rangle \phi$ which then by IH is equivalent to a formula in \mathcal{L}_{until} . If P is of the form $? \psi; Q$, then if Q is atomic $\langle \pi \rangle \phi = \langle \langle ? \psi; Q \rangle^* \rangle \phi \equiv U_Q(\phi, \psi)$, whence in \mathcal{L}_{until} . If Q is of the form $? \theta; Q'$ it reduces as before. If $Q = \langle Q' \rangle^*$, then $\langle \pi \rangle \phi = \langle \langle ? \psi; \langle Q' \rangle^* \rangle^* \rangle \phi$ which is equivalent to $\phi \vee (\psi \wedge \langle \langle Q' \rangle^* \rangle \phi)$, which by IH is equivalent to a formula in \mathcal{L}_{until} . \square

Actually, one can be even more economic in defining this extension of \mathcal{L}_B . Let us redefine \mathcal{L}_{until} to be the modal language with the following four binary modal operators: for $\pi \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$, $\mathfrak{M}, t \models \text{Until}_\pi(\phi, \psi)$ iff there exists a t' such that $tR_{\pi^+}t'$ and $\mathfrak{M}, t' \models \phi$ and for all t'' such that $tR_{\pi^+}t''R_{\pi^+}t'$ it holds that $\mathfrak{M}, t'' \models \psi$. Then $\langle \pi \rangle \phi$ and $\langle \pi^* \rangle \phi$ can be defined to be $\text{Until}_\pi(\phi, \perp)$ and $\phi \vee \text{Until}_\pi(\phi, \top)$, respectively. The previous (non strict) until construct $U_\pi(\phi, \psi)$ is equivalent to $\phi \vee (\psi \wedge \text{Until}_\pi(\phi, \psi))$.

Let us briefly discuss the relationship between the modal languages discussed in this paper and the first order logic of ordered trees. Let \mathcal{L}_{FO} denote the first order language over the signature with binary predicates $\{R_\downarrow, R_\leftarrow, R_{\downarrow^*}, R_{\leftarrow^*}\}$ and countably many unary predicates. \mathcal{L}_{FO} is interpreted on ordered trees in the obvious way, with R_\downarrow being the daughter relation, and so on. Kracht's language \mathcal{L}_K can express properties beyond the power of \mathcal{L}_{FO} . For examples, it can express the property of having an odd number of daughters:

$$(1.2.3) \quad \langle \downarrow \rangle (\neg \langle \leftarrow \rangle \top \wedge \langle (\rightarrow; \rightarrow)^* \rangle \neg \langle \rightarrow \rangle \top).$$

On the other hand, Theorem 3.2.1 entails that every Palm formula is equivalent to a formula $\phi(x)$ in \mathcal{L}_{FO} ; we simply use the *standard translation* of until into \mathcal{L}_{FO} (see Blackburn *et al* (2001)). We conjecture that the converse is also true: \mathcal{L}_P is functionally complete with respect to \mathcal{L}_{FO} . For unordered trees such a result (generalizing Kamp's famous theorem to trees) can be found in ?.

Theorem 3.2.1 together with (3.2.3) entail that \mathcal{L}_P is strictly contained in \mathcal{L}_K . That \mathcal{L}_B is strictly contained in \mathcal{L}_P follows from the well known fact that until is not expressible on linear orders from the future and past modalities. For a concrete example of difference in expressive power, note that the property of having exactly $2p$ daughters is expressible in \mathcal{L}_P :

$$(1.2.4) \quad \langle \downarrow \rangle (p \wedge \neg \langle \leftarrow^+ \rangle p \wedge \langle \rightarrow \rangle \langle (\rightarrow; \rightarrow)^* \rangle (p \wedge \neg \langle \rightarrow^+ \rangle p))$$

However an easy *bisimulation* argument (see Blackburn *et al* (2001) for the definition of bisimulation) can be used to show that \mathcal{L}_B cannot express this property.

We conclude this section with a summary of the relative expressive power of the languages we have discussed:

- $\mathcal{L}_B \subsetneq \mathcal{L}_P = \mathcal{L}_{until} \subsetneq \mathcal{L}_K$.
- $\mathcal{L}_P \subseteq \mathcal{L}_{FO}$ and $\mathcal{L}_K \not\subseteq \mathcal{L}_{FO}$.
- **Conjecture:** $\mathcal{L}_{until} = \mathcal{L}_{FO}$.

1.3 Complexity

In model theoretic syntax we specify a certain class of trees by stating a theory Θ in a tree language (Θ is our grammatical theory). Thus a key question is: given a formula ϕ , does ϕ describe a structure that is grammatical with respect to this theory? More formally: does there exist a model \mathfrak{M} such that \mathfrak{M} is a model of Θ (i.e., every formula in Θ is true at every node in \mathfrak{M}) and \mathfrak{M} satisfies ϕ (i.e., ϕ is true at the root of \mathfrak{M})? This holds iff $\Theta \not\models \text{root} \rightarrow \neg\phi$. (Here and below we also use root to denote the formula $\neg(\uparrow)\top$, which indeed is satisfied at the root of a tree only.) This is the type of problem we will study. For L a language, the L consequence problem consists of all pairs (Γ, χ) with $\Gamma \cup \{\chi\}$ a finite set of L formulas such that $\Gamma \models \chi$. We now study the complexity of this problem for $\mathcal{L}_B, \mathcal{L}_P$ and \mathcal{L}_K .

Decidability of the \mathcal{L}_K consequence problem is shown in ?, Theorem 5, via a reduction to the \mathcal{L}_B consequence problem. Unfortunately the reduction is not correct (a counterexample is given in the Appendix to this paper). However \mathcal{L}_K decidability can be proved by interpreting it in $L_{K,P}^2$, the monadic second order logic of variably branching trees of ?. (The decidability of the satisfiability problem for $L_{K,P}^2$ follows, in turn, via an interpretation into $S\omega S$.) The translation of \mathcal{L}_K formulas into $L_{K,P}^2$ is straightforward. Note, in particular, that we can use second order quantification to define the transitive closure of a relation: for R any binary relation, xR^*y holds iff

$$x = y \vee \forall X (X(x) \wedge \forall z, z' (X(z) \wedge zRz' \rightarrow X(z')) \rightarrow X(y)).$$

Note that although this reduction yields \mathcal{L}_K decidability, it only gives us a non elementary decision procedure.

What of the complexity of these consequence problems? In ? the problem for \mathcal{L}_B was claimed to be in EXPTIME², but the proof contains a mistake. Here we show that the claim is indeed correct, and that the same result holds for the language \mathcal{L}_P . Before we go into the proof details we consider the problem in a bit more detail. We first look at the lower bound:

Theorem 1.3.1. *The consequence problem for the language with only \downarrow is EXPTIME-hard.*

Proof: This is an immediate corollary of ? analysis of the lower bound result for PDL. She notes that the following fragment of PDL is EXPTIME-hard: formulas

²EXPTIME is the class of all problems solvable in exponential time. A problem is solvable in exponential time if there is a deterministic exponentially time bounded Turing machine that solves it. A deterministic Turing machine is exponentially time bounded if there is a polynomial $p(n)$ such that the machine always halts after at most $2^{p(n)}$ steps, where n is the length of the input.

of the form $\psi \wedge [a^*]\theta$, (where ψ and θ contain only the atomic program a and no embedded modalities) that are satisfiable at the root of a finite binary tree. Identifying the program a with \downarrow , the result follows (because $[\downarrow^*]\theta \wedge \psi$ is satisfiable at the root of a finite tree iff $\theta \not\models_{\text{root}} \neg\psi$). \square For full PDL this

bound is optimal. There is even a stronger result: every satisfiable PDL formula ϕ can be satisfied on a model with size exponential in the length of ϕ . Unfortunately with tree-based models there is no hope for such a result:

For every natural number n , there exists a satisfiable formula of size $\mathcal{O}(n^2)$ in the language with only \downarrow and \downarrow^* which can only be satisfied on at least binary branching trees of depth at least 2^n .

A formula which forces the deep branch is given in ? : Proposition 6.51; one only has to add the conjunct $[\downarrow^*](\langle\downarrow\rangle p \wedge \langle\downarrow\rangle\neg p)$ for some new variable p to enforce binary branching. Note that the size of the model is double exponential in the size of the formula. This means that a decision algorithm which tries to construct a tree model must run at least in exponential space, as it will need to keep a whole branch in memory.

Fortunately we can do better, taking a cue from the completeness proof for a related language in ?. Instead of constructing a model we design an algorithm which searches for a “good” set of labelings of the nodes of a model. Label sets consist of subformulas of the formula ϕ whose satisfiability is to be decided. From a good set of labels we can construct a labeled tree model which satisfies ϕ . The gain in complexity comes from the fact that the number of labels is bound by an exponential in the number of subformulas of ϕ . As we shall show, the search for a good set of labels among the possible ones can be implemented in time polynomial in the number of possible labels using the technique of elimination of Hintikka sets developed by ?. Thus we will be able to prove:

Theorem 1.3.2. *The \mathcal{L}_p consequence problem is in EXPTIME.*

The proof of Theorem 3.3.2 consists of a reduction and a decision algorithm. The reduction combines ideas from ?, Theorem 5 and Rabin’s reduction of $S\omega S$ to $S2S$.

Let \mathcal{L}_2 be the modal language with only the two programs $\{\downarrow_1, \downarrow_2\}$ and the modal constant $root$. \mathcal{L}_2 is interpreted on finite ordered *binary* trees, with \downarrow_1 and \downarrow_2 interpreted by the first and second daughter relation, respectively, and $root$ holds exactly at the root. We present such trees by triples (T, \succ_1, \succ_2) .

Lemma 1.3.3. There is an effective reduction from the \mathcal{L}_p consequence problem to the \mathcal{L}_2 consequence problem.

The proof is provided in the appendix. The theorem now follows from the previous lemma together with the following one, which we shall prove in the next section:

Lemma 1.3.4. The \mathcal{L}_2 consequence problem is in EXPTIME.

1.4 Deciding \mathcal{L}_2

We will give an EXPTIME algorithm that on input \mathcal{L}_2 formulas γ, χ decides whether there exists a model \mathfrak{M} in which γ is true everywhere and χ is true at the root. To this the consequence problem reduces because $\gamma \not\models \phi$ iff there exists a model in which $\gamma \wedge (p \leftrightarrow \neg\phi \vee \langle \downarrow_1 \rangle p \vee \langle \downarrow_2 \rangle p)$ is true everywhere and p is true at the root. Here p is a new propositional variable whose intended meaning is $\langle (\downarrow_1 \cup \downarrow_2)^* \rangle \neg\phi$.

Preliminaries. Recall that a set of formulas Σ is said to be closed under subformulas iff for all $\phi \in \Sigma$, if ψ is a subformula of ϕ then $\psi \in \Sigma$. It is closed under single negations if whenever ϕ is in the set and ϕ is not of the form $\neg\psi$ then also $\neg\phi$ is in the set. For Σ a set of formulas, $Cl(\Sigma)$ (called the *closure* of Σ) is defined to be the smallest set of formulas containing Σ that is closed under subformulas and single negations and which contains the constant *root* and the formulas $\langle \downarrow_1 \rangle \top$ and $\langle \downarrow_2 \rangle \top$. From now on we fix two arbitrary formulas γ and χ .

Definition 1.4.1 (Hintikka Set). Let $A \subseteq Cl(\{\gamma, \chi\})$. We call A a *Hintikka Set* if A satisfies the following conditions:

1. $\gamma \in A$ and $\top \in A$.
2. If $\phi \in Cl(\{\gamma, \chi\})$ then $\phi \in A$ iff $\neg\phi \notin A$.
3. If $\phi \wedge \psi \in Cl(\{\gamma, \chi\})$ then $\phi \wedge \psi \in A$ iff $\phi \in A$ and $\psi \in A$.
4. $\langle \downarrow_1 \rangle \top \in A$ iff $\langle \downarrow_2 \rangle \top \in A$.

Let $HS(\gamma, \chi)$ denote the set of all Hintikka Sets which are a subset of $Cl(\gamma, \chi)$. Note that $|HS(\gamma, \chi)| \leq 2^{|Cl(\gamma, \chi)|}$.

For H a set of Hintikka sets, let $l : H \rightarrow \{0, 1, \dots, |H|\}$ be a function assigning to each $A \in H$ a level. We call a structure (H, l) an ordered set of Hintikka sets.

Definition 1.4.2 (Saturation). Let (H, l) be an ordered set of Hintikka sets and let k be either 1 or 2. We call (H, l) saturated if for all $A \in H$, $\langle \downarrow_k \rangle \phi \in A$ only if there exists a $B \in H$ such that $l(A) > l(B)$ and for all $\langle \downarrow_k \rangle \psi \in Cl(\gamma, \chi)$, $\langle \downarrow_k \rangle \psi \in A$ iff $\psi \in B$.

The connection. We are ready to formulate our most important lemma.

Lemma 1.4.1. The following are equivalent:

1. There exists a model over a finite binary branching tree in which γ is true everywhere and χ is true at the root;
2. There exists a saturated ordered set of Hintikka Sets (H, l) , with $H \subseteq HS(\gamma, \chi)$ and there is an $A \in H$ with $\{root, \chi\} \subseteq A$.

Proof: First assume \mathfrak{M} is a model over a finite binary branching tree in which γ is true everywhere and χ is true at the root. For each node n define $A_n = \{\psi \in Cl(\gamma, \chi) \mid \mathfrak{M}, n \models \psi\}$. Obviously each A_n is a Hintikka set and there is an A with $\{root, \chi\} \subseteq A$. Let H be the set of all such A_n . Let \hat{A}_n abbreviate the conjunction of all formulas in A_n . Inductively define the level function on H . First define which Hintikka Sets are of level 0:

$$l(A) = 0 \text{ if } t \in A.$$

Next, suppose the i -th level is defined. First define: $S_i = \{A \in H \mid l(A) \leq i\}$. Next, if $H \setminus S_i$ is non-empty then the $i + 1$ -th level is defined as follows: $l(A) = i + 1$ if $A \notin S_i$ and

$$\mathfrak{M}, root \models \langle (\downarrow_1 \cup \downarrow_2)^* \rangle (\hat{A} \wedge [\downarrow_1 \cup \downarrow_2][(\downarrow_1 \cup \downarrow_2)^*] \bigvee_{B \in S_i} \hat{B}).$$

On the other hand, if $H \setminus S_i$ is empty then there is no $i + 1$ -th level. It is not hard to show that (H, l) is saturated.

Now assume (H, l) is saturated and there is an $A_0 \in H$ with $\{root, \chi\} \subseteq A_0$.

We inductively construct a finite binary tree and a function h from the nodes to H in such a way that we can turn the tree into a model \mathfrak{M} for which we can prove the truth lemma,

$$\text{for all } \psi \in Cl(\gamma, \chi), \mathfrak{M}, n \models \psi \text{ if and only if } \psi \in h(n).$$

By the first condition on Hintikka sets and the existence of $A_0 \in H$, this yields a model in which γ is true everywhere and χ at the root. Let \mathcal{T} be some denumerably infinite set; we shall use (finitely many) of its elements as the tree nodes.

Stage 0. Define T_0 to be $\{t_0\}$; \succ_1^0 to be \emptyset ; \succ_2^0 to be \emptyset ; and h_0 to be $\{\langle w_0, A_0 \rangle\}$.

Stage $n + 1$. Suppose n stages of the inductive construction have been performed. We call a pair $\langle t, k \rangle$ (where $t \in T_n$ and $k \in \{1, 2\}$) an *unsatisfied demand* iff $\langle \downarrow_k \rangle \phi \in h(t)$ but there is no $t' \in T_n$ such that $t \succ_k t'$. If there are no unsatisfied

demands the construction is complete. Otherwise, choose an unsatisfied demand $\langle t, k \rangle$. As (H, l) is saturated there exists a $B \in H$ such that $l(h_n(t)) > l(B)$ and for all $\langle \downarrow_k \rangle \psi \in Cl(\Sigma)$, $\langle \downarrow_k \rangle \psi \in h_n(t)$ iff $\psi \in B$. Let $t' \in \mathcal{T} \setminus T_n$. Define:

$$\begin{aligned} T_{n+1} &= T_n \cup \{t'\} \\ \succ_k^{n+1} &= \succ_k^n \cup \{\langle t, t' \rangle\} \\ \succ_j^{n+1} &= \succ_j^n \\ h_{n+1} &= h_n \cup \{\langle t', B \rangle\}. \end{aligned}$$

While adjoining a new node t' to t as described in the inductive step may result in new unsatisfied demands $\langle t', k \rangle$, where $k \in \{1, 2\}$, we were careful to choose $h(t')$ from a strictly lower level than $h(t)$. This means that in the course of the construction we will be forced to map the newly adjoined node t' to a Hintikka set of level zero; but doing so cannot give rise to an unsatisfied demand. Thus the construction process terminates.

Let $\langle T, \succ_1, \succ_2 \rangle$ be the result of the final stage. Note that by the last condition on Hintikka Sets and the fact that there are no unsatisfied demands every non leaf node has two daughters. Turn $\langle W, \succ_1, \succ_2 \rangle$ into a model $\mathfrak{M} = \langle W, \succ_1, \succ_2, V \rangle$ by setting $n \in V(p)$ iff $p \in h(n)$.

Prove the truth lemma by an induction on the complexity of the formulas. The base case is by definition of V . The boolean cases are by conditions 2 and 3 on Hintikka sets. The cases for $\langle \downarrow_k \rangle$ follow from the fact that all demands are satisfied. \square

The algorithm. The decision algorithm for \mathcal{L}_2 satisfiability is presented in Figure 3.1. Its most important properties are presented in the next lemma.

- Lemma 1.4.2.**
1. *Elimination of HS*(γ, χ) terminates after at most $|HS(\gamma, \chi)|$ rounds of the do loop.
 2. The statement “ $\langle S, l \rangle$ is a saturated ordered set of Hintikka sets” holds after the do loop of *Elimination of HS*(γ, χ).

Proof: (1) The bound function of the do loop is the size of $P_{\circ\circ 1}$ which is being reduced in every round, or the loop terminates because $L = \emptyset$. The initial size of $P_{\circ\circ 1}$ is bounded by $|HS(\gamma, \chi)| = 2^{|Cl(\gamma, \chi)|}$.

(2) Because the statement “ $HS(\gamma, \chi) = P_{\circ\circ 1} \uplus S$ and $\langle S, l \rangle$ is a saturated ordered set of Hintikka sets” holds before the do loop and is an invariant of the do loop. \square

This lemma immediately yields our desired result:

PROOF OF LEMMA 3.3.4. In order to decide whether there exists a model in which γ is true everywhere and χ is true at the root we run *Elimination of HS*(γ, χ). The algorithm is correct by Lemma 3.4.1 and part 2 of Lemma 3.4.2. By part 1 the algorithm terminates after at most $|HS(\gamma, \chi)| \leq 2^{|Cl(\gamma, \chi)|}$ rounds of the do loop. As in **?**, the tests inside the do loop take time bounded by $p(|HS(\gamma, \chi)|)$ for some polynomial p . Since $|Cl(\gamma, \chi)|$ is linear in the number of subformulas of γ, χ , the algorithm is in EXPTIME. QED

```

begin
  L := {A ∈ HS(γ, χ) | ¬⟨↓1⟩T ∈ A};
  Pool := HS(γ, χ) \ L;
  S := L;
  i := 0;
  l := {(A, i) | A ∈ L};
  do L ≠ ∅ →
    L := {A ∈ Pool | (S ∪ {A}, l ∪ (A, i + 1))
              is a saturated ordered set of
              Hintikka Sets };
    Pool := Pool \ L;
    S := S ∪ L;
    i := i + 1;
    l := l ∪ {(A, i) | A ∈ L}
  od;
  if ∃A ∈ S: {χ, root} ⊆ A
  then true
  else fail
  fi
end

```

Figure 1.1: The algorithm *elimination of HS*(γ, χ).

1.5 Conclusions

We discussed the relative expressivity of three modal languages proposed for specifying grammatical constraints on finite ordered trees. We added a fourth language, $\mathcal{L}_{\text{until}}$, and conjectured it to be precisely as expressive as the first order language

of ordered trees. We showed that the consequence problems for \mathcal{L}_B , \mathcal{L}_P and $\mathcal{L}_{\text{until}}$ are EXPTIME-complete. We conjecture that the same bound holds for \mathcal{L}_K as well (note that if Kracht’s polynomial reduction of \mathcal{L}_K satisfiability to \mathcal{L}_B satisfiability can be repaired, this follows immediately from the results in this paper).

Palm argued that writing grammatical constraints in the language \mathcal{L}_P is straightforward and yields formulas which are simpler and easier to understand than first order formulas. We think this is due to the lack of variables and the direct use of the “tree-axis” in \mathcal{L}_P formulas. It is interesting to note that the language XPath contains exactly these two features. XPath was designed to extract elements from XML documents, and the natural models of XML documents are finite ordered trees.

Acknowledgements This work was carried out as part of the INRIA funded research partnership between LIT (Language and Inference Technology Group, University of Amsterdam) and LED (Langue et Dialogue, LORIA, Nancy). Marx is supported by NWO grant 612.000.106.

Bibliography

- Blackburn, P., M. de Rijke, and Y. Venema (2001). *Modal Logic*. Cambridge University Press.
- Blackburn, P. and W. Meyer-Viol (1994). Linguistics, logic, and finite trees. *Logic Journal of the IGPL*, 2:3–29.
- Blackburn, P., W. Meyer-Viol, and M. de Rijke (1996). A proof system for finite trees. In H. K. Büning, ed., *Computer Science Logic*, volume 1092 of *LNCS*, pp. 86–105. Springer.
- Harel, D., D. Kozen, and J. Tiuryn (2000). *Dynamic Logic*. MIT Press.
- Kracht, M. (1995). Syntactic codes and grammar refinement. *Journal of Logic, Language and Information*, 4:41–60.
- Kracht, M. (1997). Inessential features. In C. Retore, ed., *Logical Aspects of Computational Linguistics*, number 1328 in *LNAI*, pp. 43–62. Springer.
- Palm, A. (1999). Propositional tense logic for trees. In *Sixth Meeting on Mathematics of Language*. University of Central Florida, Orlando, Florida.
- Pratt, V. (1979). Models of program logics. In *Proceedings of the 20th IEEE symposium on Foundations of Computer Science*, pp. 115–122.

- Rogers, J. (1998). *A descriptive approach to language theoretic complexity*. CSLI Press.
- Schlingloff, B.-H. (1992). Expressive completeness of temporal logic of trees. *Journal of Applied Non-Classical Logics*, 2(2):157–180.
- Spaan, E. (1993). *Complexity of modal logics*. Ph.D. thesis, University of Amsterdam, Institute for Logic, Language and Computation.
- Weyer, M. (2002). Decidability of S1S and S2S. In E. G. et al., ed., *Automata, Logics, and Infinite Games*, volume 2500 of LNCS, pp. 207–230. Springer.

Appendix

Counterexample to Kracht’s reduction. We present a counterexample to the reduction from \mathcal{L}_K to \mathcal{L}_B given in the proof of Theorem 5 in ?. Take the following non satisfiable formula $\langle\langle\downarrow^*\rangle^*\rangle\perp$. Then $\nabla(\psi)$ is

$$\begin{aligned} q_\perp &\leftrightarrow \perp \\ q_{\langle\langle\downarrow^*\rangle^*\rangle\perp} &\leftrightarrow q_\perp \vee q_{\langle\downarrow^*\rangle\langle\langle\downarrow^*\rangle^*\rangle\perp} \\ q_{\langle\downarrow^*\rangle\langle\langle\downarrow^*\rangle^*\rangle\perp} &\leftrightarrow q_{\langle\langle\downarrow^*\rangle^*\rangle\perp} \vee q_{\langle\downarrow^*\rangle\langle\downarrow^*\rangle\langle\langle\downarrow^*\rangle^*\rangle\perp} \\ q_{\langle\downarrow^*\rangle\langle\downarrow^*\rangle\langle\langle\downarrow^*\rangle^*\rangle\perp} &\leftrightarrow \langle\downarrow^*\rangle q_{\langle\downarrow^*\rangle\langle\langle\downarrow^*\rangle^*\rangle\perp}. \end{aligned}$$

$q_{\langle\langle\downarrow^*\rangle^*\rangle\perp}$ can be made true in the tree with domain $\{0,00\}$, with 0 the root and 00 her daughter and the following valuation:

$$\begin{aligned} V(0) &= \{q_{\langle\downarrow^*\rangle\langle\langle\downarrow^*\rangle^*\rangle\perp}, q_{\langle\downarrow^*\rangle\langle\langle\downarrow^*\rangle^*\rangle\perp}, q_{\langle\langle\downarrow^*\rangle^*\rangle\perp}\} \\ V(00) &= \{q_{\langle\downarrow^*\rangle\langle\langle\downarrow^*\rangle^*\rangle\perp}, q_{\langle\langle\downarrow^*\rangle^*\rangle\perp}\} \end{aligned}$$

This model makes $\nabla(\psi)$ true. But clearly we do not have $0 \models \langle\langle\uparrow;\downarrow^*\rangle^*\rangle\perp$, contrary to Kracht’s claim that for every node n , and for every formula χ in the Fisher Ladner closure it holds that $n \models \chi \leftrightarrow q_\chi$.

Proof of Lemma 3.3.3 Although Kracht’s reduction of \mathcal{L}_K to \mathcal{L}_B is flawed, his approach can be used to give a reduction of \mathcal{L}_P to \mathcal{L}_2 , and we shall do so here. Note that $\gamma_1, \dots, \gamma_n \models \chi$ iff $\models [\downarrow^*](\gamma_1 \wedge \dots \wedge \gamma_n) \rightarrow \chi$. Thus we need only reduce the consequence problem for empty Γ . The proof of Theorem 3.2.1 gives an effective reduction from \mathcal{L}_P to $\mathcal{L}_{\text{until}}$ formulas.

Let $\chi \in \mathcal{L}_{\text{until}}$. Let $Cl(\chi)$ be the smallest set of formulas containing all subformulas of χ and which is closed under taking single negations and under the rule: $Until_\pi(\phi, \psi) \in Cl(\chi) \Rightarrow \psi \wedge Until_\pi(\phi, \psi) \in Cl(\chi)$.

We associate a formula $\nabla(\chi)$ with χ as follows. We create for each $\phi \in Cl(\chi)$, a new propositional variable q_ϕ . Now $\nabla(\chi)$ “axiomatizes” these new variables as follows:

$$\begin{aligned} q_p &\leftrightarrow p \\ q_{\neg\phi} &\leftrightarrow \neg q_\phi \\ q_{\phi \wedge \psi} &\leftrightarrow q_\phi \wedge q_\psi \\ q_{Until_\pi(\phi, \psi)} &\leftrightarrow \langle \pi \rangle q_\phi \vee \langle \pi \rangle q_{(\psi \wedge Until_\pi(\phi, \psi))}. \end{aligned}$$

We claim that for every model \mathfrak{M} which validates $\nabla(\chi)$, for every node n and for every subformula $\phi \in Cl(\chi)$, $\mathfrak{M}, n \models q_\phi$ iff $\mathfrak{M}, n \models \phi$.

The proof is by induction on the structure of the formula, and for the left to right direction of the until case by induction on the depth of direction of π . We do that case for $Until_\downarrow$. Let n be a leaf. By the axiom in $\nabla(\chi)$, $\mathfrak{M}, n \not\models q_{Until_\downarrow(\phi, \psi)}$. But also $\mathfrak{M}, n \not\models Until_\downarrow(\phi, \psi)$. Now let n be a node with $k+1$ descendants, and let the claim hold for nodes with k descendants. Let $\mathfrak{M}, n \models q_{Until_\downarrow(\phi, \psi)}$. Then by the axiom $\mathfrak{M}, n \models \langle \downarrow \rangle q_\phi$ or $\mathfrak{M}, n \models \langle \downarrow \rangle q_{(\psi \wedge Until_\downarrow(\phi, \psi))}$. In the first case, there exists a daughter m of n and $\mathfrak{M}, m \models q_\phi$. By inductive hypothesis, $\mathfrak{M}, m \models \phi$, whence $\mathfrak{M}, n \models Until_\downarrow(\phi, \psi)$. In the second case, there exists a daughter m of n and $\mathfrak{M}, m \models q_\psi$ and $\mathfrak{M}, m \models q_{Until_\downarrow(\phi, \psi)}$. Whence, by first inductive hypothesis, $\mathfrak{M}, m \models \psi$ and the second inductive hypothesis $\mathfrak{M}, m \models Until_\downarrow(\phi, \psi)$. But then also $\mathfrak{M}, n \models Until_\downarrow(\phi, \psi)$. Hence the following holds for each $\chi \in \mathcal{L}_{until}$,

$$\models \chi \Leftrightarrow \nabla(\chi) \models q_\chi.$$

Note that the only modalities occurring in $\nabla(\chi)$ are $\langle \pi \rangle$ for π one of the four compass arrows. We can further reduce the number of arrows to only \downarrow, \rightarrow when we add two modal constants *root* and *first* for the root and first elements, respectively.

Let χ be a formula in this fragment. As before create a new variable q_ϕ for each (single negation of a) subformula ϕ of χ . Create $\nabla(\chi)$ as follows:

$$\begin{aligned} q_p &\leftrightarrow p \\ q_{\neg\phi} &\leftrightarrow \neg q_\phi \\ q_{\phi \wedge \psi} &\leftrightarrow q_\phi \wedge q_\psi \\ q_{\langle \pi \rangle \phi} &\leftrightarrow \langle \pi \rangle q_\phi \text{ for } \pi \in \{\downarrow, \rightarrow\}. \end{aligned}$$

And for each subformula $\langle \uparrow \rangle \phi$ and $\langle \leftarrow \rangle \phi$ we add to $\nabla\chi$ the axioms

$$\begin{aligned} q_\phi &\rightarrow [\downarrow]q_{\langle \uparrow \rangle \phi}, & \langle \downarrow \rangle q_{\langle \uparrow \rangle \phi} &\rightarrow q_\phi, & q_{\langle \uparrow \rangle \phi} &\rightarrow \neg root, \\ q_\phi &\rightarrow [\rightarrow]q_{\langle \leftarrow \rangle \phi}, & \langle \rightarrow \rangle q_{\langle \leftarrow \rangle \phi} &\rightarrow q_\phi, & q_{\langle \leftarrow \rangle \phi} &\rightarrow \neg first. \end{aligned}$$

We claim that for every model \mathfrak{M} which validates $\nabla(\chi)$, for every node n and for every subformula $\phi \in Cl(\chi)$, $\mathfrak{M}, n \models q_\phi$ iff $\mathfrak{M}, n \models \phi$. An easy induction shows this. We do the case for $\langle \uparrow \rangle \phi$. If $n \models \langle \uparrow \rangle \phi$, then the parent of n models ϕ , whence by inductive hypothesis, it models q_ϕ , so by the axiom $q_\phi \rightarrow [\downarrow]q_{\langle \uparrow \rangle \phi}$, $n \models q_{\langle \uparrow \rangle \phi}$. Conversely, if $n \models q_{\langle \uparrow \rangle \phi}$, then by axiom $q_{\langle \uparrow \rangle \phi} \rightarrow \neg root$, n is not the root. So the parent of n exists and it models $\langle \downarrow \rangle q_{\langle \uparrow \rangle \phi}$. Then it models q_ϕ by axiom $\langle \downarrow \rangle q_{\langle \uparrow \rangle \phi} \rightarrow q_\phi$, and by inductive hypothesis it models ϕ . Thus $n \models \langle \uparrow \rangle \phi$. Hence, the following holds

$$\gamma \models \chi \Leftrightarrow \nabla(\gamma \wedge \chi), q_\gamma \models q_\chi.$$

Note that the formulas on the right hand side only contain the modalities $\langle \downarrow \rangle$ and $\langle \rightarrow \rangle$. Finally we reduce the consequence problem to that of binary branching trees.

Let χ be a formula, let d and q_{first} be new variables. Let $(\cdot)'$ be the following translation:

$$\begin{aligned} p' &= p \\ (\neg \phi)' &= \neg \phi' \\ (\phi \wedge \psi)' &= \phi' \wedge \psi' \\ (\langle \downarrow \rangle \phi)' &= \langle \downarrow_1 \rangle \langle (?d; \downarrow_2)^* \rangle (d \wedge \phi') \\ (\langle \rightarrow \rangle \phi)' &= \langle \downarrow_2 \rangle (d \wedge \phi') \\ root' &= root \\ first' &= q_{first}. \end{aligned}$$

Note that this translation goes to the Palm language generated from the programs \downarrow_1 and \downarrow_2 . Then χ is satisfiable on a tree in which γ is true in every node iff $d \wedge \chi'$ is satisfiable on a binary branching tree in which $d \rightarrow \gamma'$ and $[\downarrow_1]q_{first} \wedge [\downarrow_2]\neg q_{first} \wedge (root \rightarrow q_{first})$ is true everywhere. This is shown using the main idea from the reduction from $S\omega S$ to $S2S$ explained in ?. Whence we have that

$$\gamma \models \chi \Leftrightarrow d \rightarrow \gamma' \wedge [\downarrow_1]q_{first} \wedge [\downarrow_2]\neg q_{first} \wedge (root \rightarrow q_{first}) \models d \rightarrow \chi'.$$

Now we can use the first reduction again to reduce this problem to the consequence problem of the language with just the modalities $\langle \downarrow_1 \rangle$ and $\langle \downarrow_2 \rangle$, interpreted on binary trees.

Chaining these reductions together, we obtain the reduction stated in the lemma.