16

# Inessential Features, Ineliminable Features, and Modal Logics for Model Theoretic Syntax

Hans-Jörg Tiede †

**Abstract**

While monadic second-order logic (MSO) has played a prominent role in model theoretic syntax, modal logics have been used in this context since its inception. When comparing propositional dynamic logic (PDL) to MSO over trees, Kracht (1997) noted that there are tree languages that can be defined in MSO that can only be defined in PDL by adding new features whose distribution is predictable. He named such features "inessential features". We show that Kracht's observation can be extended to other modal logics of trees in two ways. First, we demonstrate that for each stronger modal logic, there exists a tree language that can only be defined in a weaker modal logic with inessential features. Second, we show that any tree language that can be defined in a stronger modal logic, but not in some weaker modal logic, can be defined with inessential features. Additionally, we consider Kracht's definition of inessential features more closely. It turns out that there are features whose distribution can be predicted, but who fail to be inessential in Kracht's sense. We will look at ways to modify his definition.

**Keywords** Model Theoretic Syntax, Modal Logic, Tree Automata

## 16.1 Introduction

Model theoretic syntax is a research paradigm in mathematical linguistics that uses tools from descriptive complexity theory to formalize

†This research was supported by an Illinois Wesleyan University grant and a junior leave.

*FG-MoL 2005.*
James Rogers (ed.).

grammatical theories using logic. It is the aim of model theoretic syntax to find for a given grammatical theory the weakest logic that suffices to formalize it (Cornell and Rogers, 2000). At present there exist two main approaches to model theoretic syntax, those based on modal logics and those based on monadic second order logic. Since there are few, if any, linguistic examples that cannot be defined in the weakest modal logic that has been considered, it is not always clear what motivates the use of the more expressive logics. We want to argue here that there are reasons for using weaker frameworks. The present study does not aim at finding the "true" logic for model theoretic syntax. Instead, it is guided by the methodological principle that we should always use the weakest formalism that suffices to capture the phenomena under consideration, and that the use of stronger formalisms should be justified.

In Afanasiev et al. (2005), three different modal logics for the description of trees are discussed: a basic modal logic of trees, $\mathcal{L}_{core}$, Palm's tense logic of trees, $\mathcal{L}_{cp}$, and Kracht's dynamic logic of trees, $\mathrm{PDL}_{tree}$. While the relationship between these logics and to others used in model theoretic syntax is well-understood, in that each stronger logic includes the weaker ones properly, and that all are properly included in Roger's Monadic Second Order Logic of Trees (MSO) (Rogers, 1998), the relationship of these logics to tree languages is not as well understood. We will make use of Kracht's (Kracht, 1997) concept of an inessential feature to get a better understanding of the tree languages that are definable or undefinable in these logics.

Kracht introduced the concept of an inessential feature to formalize the concept of a feature whose distribution is predictable from the other features. Two well-known linguistic examples of inessential features are the Slash feature of GPSG and Bar feature of X-bar theory. In addition to inessential features, Kracht also considered whether a feature is eliminable in some logic, which he identified with being globally explicitly definable. We will consider a slightly weaker notion of eliminability, but since we are mainly interested in *ineliminability*, this notion implies Kracht's by contraposition. It was shown by Kracht that there exists a set of feature trees that is definable in $\mathrm{PDL}_{tree}$ and an inessential feature that is ineliminable in $\mathrm{PDL}_{tree}$, but eliminable in MSO. The main purpose of this observation was to show that $\mathrm{PDL}_{tree}$ is strictly weaker than MSO over trees.

We show that Kracht's theorem can be generalized to all three modal logics of trees. The proof of this result involves Thatcher's theorem, which states that every regular tree language is the projection of a local tree language, and relates it to Kracht's inessential features. When applied to deterministic bottom-up finite tree automata, Thatcher's

construction of a local tree language introduces inessential features.

We also consider the definition of inessential features more closely. We want to argue that Kracht's formalization is too strong, as there are tree languages with features whose distribution can be predicted from the other features, but which fail to be inessential in Kracht's sense. Such features can be constructed by using Thatcher's construction on non-deterministic tree automata. It can be shown that such features can be turned into inessential features, which can then be eliminated in MSO. Thus, logics that can eliminate any inessential feature may be too strong. This can be seen as support for the use of weaker logics for model theoretic syntax.

## 16.2 Features and Ranked Alphabets

Kracht's definition of inessential features is given in the context of feature trees, in which each node is labeled with a set of boolean features. We are here considering trees to be terms over a ranked alphabet which differ from feature trees in that each node is labeled with a single symbol which has a fixed arity. We will use the following representation of boolean features as ranked symbols to translate feature trees into terms.

**Definition 41** Given a finite set of boolean features $F = \{f_1, \ldots, f_n\}$, the *(binary) ranked alphabet based on* $F$, $\Sigma^F$, is defined as

$$\Sigma^F = \{f_1, \neg f_1\} \times \cdots \times \{f_n, \neg f_n\} \times \{0, 2\}$$

where each $f_i, \neg f_i$ represents whether or not a feature holds at a given node and 0 or 2 represent the aritiy of the symbol. Thus, $(f_1, \neg f_2, 0)$ would be a leaf symbol, and $(f_1, \neg f_2, 2)$ would be an internal node symbol. □

The previous definition can be easily generalized to trees of any arity.

**Definition 42** A *tree* is a term over a finite ranked alphabet $\Sigma$. The set of $n$-ary function symbols in $\Sigma$ will be denoted by $\Sigma_n$. The set of all trees over $\Sigma$ is denoted by $T_\Sigma$; a subset of $T_\Sigma$ is called a *tree language*. The *yield* of a tree $t$, denoted by $yield(t)$, is defined by

$$\begin{aligned} yield\,(c) &= c \\ yield\,(f\,(t_1, \ldots, t_n)) &= yield\,(t_1) \cdots yield\,(t_n) \end{aligned}$$

with $c \in \Sigma_0$ and $f \in \Sigma_n, n > 0$. □

We next define projections which we will use to study eliminability of features in the context of terms.

**Definition 43** Given a finite set of feature $F = \{f_1, \ldots, f_n\}$ and a feature $f_i \in F$, we define the *projection*, $\pi$, that eliminates $f_i$ in the natural way:

$$\pi : \Sigma^F \to \Sigma^{F-\{f_i\}}$$

This definition can be extended to arbitrary subsets $G \subseteq F$, where

$$\pi : \Sigma^F \to \Sigma^{F-G}$$

Given a projection $\pi$, we extend $\pi$ to a *tree homomorphism* $\hat{\pi}$ as follows:

$$\begin{aligned} \hat{\pi}(c) &= \pi(c) \\ \hat{\pi}(f(t_1, \ldots, t_n)) &= \pi(f)(\hat{\pi}(t_1), \ldots, \hat{\pi}(t_n)) \end{aligned}$$

with $c \in \Sigma_0$ and $f \in \Sigma_n, n > 0$. For a tree language $L$, we define $\hat{\pi}(L) = \{\hat{\pi}(t) \mid t \in L\}$. □

## 16.3 Regular Tree Languages, Local Tree Languages, and Thatcher's Theorem

The regular tree languages play a central role in model theoretic syntax because they correspond to the MSO-definable languages. There are different, equivalent ways of defining the regular tree languages. We will use bottom-up (frontier-to-root) finite tree automata, because they can be determinized.

**Definition 44** A (bottom-up, non-deterministic) *finite tree automaton* (FTA) $M$ is a structure $(\Sigma, Q, F, \Delta)$ where $\Sigma$ is a ranked alphabet, $Q$ is a finite set of states, $F \subseteq Q$ is the set of final states, and $\Delta$ is a finite set of transition rules of the form $f(q_1, \ldots, q_n) \to q$ with $f \in \Sigma_n$ and $q, q_1, \ldots, q_n \in Q$. An FTA is *deterministic* if there are no two transition rules with the same left-hand-side. □

**Definition 45** A *context* $s$ is a term over $\Sigma \cup \{x\}$ containing the zeroary term $x$ exactly once. We write $s[x \mapsto t]$ for the term that results from substituting $x$ in $s$ with $t$. □

**Definition 46** Given a finite tree automaton $M = (\Sigma, Q, F, \Delta)$ the derivation relation $\Rightarrow_M \subseteq T_{Q\cup\Sigma} \times T_{Q\cup\Sigma}$ is defined by $t \Rightarrow_M t'$ if for some context $s \in T_{\Sigma\cup Q\cup\{x\}}$ there is a rule $f(q_1, \ldots, q_n) \to q$ in $\Delta$, and

$$\begin{aligned} t &= s[x \mapsto f(q_1, \ldots, q_n)] \\ t' &= s[x \mapsto q] \end{aligned}$$

We use $\Rightarrow_M^*$ to denote the reflexive, transitive closure of $\Rightarrow_M$. A finite automaton $M$ *accepts* a term $t \in T_\Sigma$ if $t \Rightarrow_M^* q$ for some $q \in F$. The *tree language accepted* by a finite tree automaton $M$, $L(M)$, is

$$L(M) = \{t \in T_\Sigma \mid t \Rightarrow_M^* q, \text{ for some } q \in F\}.$$

A tree language, $L$, is *regular* if $L = L(M)$ for some FTA $M$. □

We will now consider the relationship between regular tree languages and context-free string languages. We assume that the reader is familiar with context-free grammars (CFGs) and their languages (CFLs).

**Theorem 43** (Thatcher, 1967) If $L \subseteq T_\Sigma$ is regular, then

$$\{yield(t) \mid t \in L\}$$

is a CFL. □

While the yields of regular tree languages are CFLs, regular tree languages are more complex than the derivation trees of CFG. In order to compare the regular tree languages to the derivation trees of CFGs, we formalize the latter using the local tree languages.

**Definition 47** The *fork* of a tree $t$, $fork(t)$, is defined by

$$\begin{aligned} fork(c) &= \varnothing \\ fork(f(t_1, \cdots, t_n)) &= \{(f, root(t_1), \ldots, root(t_n))\} \cup \bigcup_{i=1}^{n} fork(t_i) \end{aligned}$$

with $c \in \Sigma_0$, $f \in \Sigma_n, n > 0$, and *root* being the function that returns the symbol at the root of its argument. For a tree language $L$, we define

$$fork(L) = \bigcup_{t \in L} fork(t)$$

□

The intuition behind the definition of $fork$ is that an element of $fork(T_\Sigma)$ corresponds to a rewrite rule of a CFG. Note that $fork(T_\Sigma)$ is always finite, since $\Sigma$ is finite.

**Definition 48** A tree language $L \subseteq T_\Sigma$ is *local* if there are sets $R \subseteq \Sigma$ and $E \subseteq fork(T_\Sigma)$, such that, for all $t \in T_\Sigma, t \in L$ iff $root(t) \in R$ and $fork(t) \subseteq E$. □

We quote without proof the following two theorems by Thatcher (1967).

**Theorem 44** (Thatcher, 1967) A tree language is a set of derivation trees of some CFG iff it is local. □

**Theorem 45** (Thatcher, 1967) Every local tree language is regular. □

While there are regular tree languages that are not local, the following theorem, also due to Thatcher (1967), demonstrates that we can obtain the regular tree languages from the local tree languages via projections.

We will review the main points of the proof, because we will use some of its details later on.

**Theorem 46** (Thatcher, 1967) For every regular tree language $L$, there is a local tree language $L'$ and a projection $\pi$, such that $L = \hat{\pi}(L')$.

**Proof** Let $L$ be a regular tree language accepted by $M = (\Sigma, Q, F, \Delta)$. We define $L'$ terms of $R$ and $E$ as follows: $R = \Sigma \times F$ and

$$E = \{((f,q),(f_1,q_1),\ldots,(f_n,q_n)) \mid f(q_1,\ldots,q_n) \to q \in \Delta, \\ f_1,\ldots,f_n \in \Sigma\}$$

We then define $L' = \{t \in T_{\Sigma\times Q} \mid root(t) \in R, fork(t) \subseteq E\}$. Notice that the trees in $L'$ encode runs of $M$. That the tree homomorphisms $\hat{\pi}$ based on the projection $\pi : \Sigma \times Q \to \Sigma$ maps $L'$ to $L$ can be easily verified.

It should be noted that, if $M$ is deterministic, there exists exactly one accepting run for each tree in $L(M)$ and thus the homomorphism $\hat{\pi} : L' \to L$ is one-to-one. □

## 16.4 Modal Logics for Model Theoretic Syntax

Model theoretic syntax is concerned with the definability of grammatical theories in certain logics. While MSO has been a particularly successful logic for this purpose, modal logics have been used for model theoretic syntax from its inception. We now define three modal logics that were considered by Afanasiev et al. (2005).

**Definition 49** The syntax of formulas for all three modal logics is defined as follows:

$$\varphi := \ p_i \mid \neg\varphi \mid \varphi \wedge \psi \mid [\pi]\varphi$$

The syntax of programs is defined for each of the three logics:

$$\pi := \to \mid \leftarrow \mid \uparrow \mid \downarrow \mid \ \pi^* \qquad (\mathcal{L}_{core})$$
$$\pi := \to \mid \leftarrow \mid \uparrow \mid \downarrow \mid \ \pi;\varphi? \ \mid \ \pi^* \qquad (\mathcal{L}_{cp})$$
$$\pi := \to \mid \leftarrow \mid \uparrow \mid \downarrow \mid \ \varphi? \ \mid \ \pi;\sigma \ \mid \ \pi \cup \sigma \ \mid \ \pi^* \qquad (\mathrm{PDL}_{tree})$$

Given a logic $\mathcal{L}$, we will denote the set of formulas of $\mathcal{L}$ over a finite set of atomic formulas $F$ by $\mathcal{L}^F$. □

The following definition is adapted from Afanasiev et al. (2005). We consider only binary trees here.

**Definition 50** Let $\{0,1\}^*$ denote the set of finite sequences over $\{0,1\}$. A *(binary) tree structure* is a tuple $(T, R_\downarrow, R_\rightarrow)$ where $T$ is a binary tree domain, i.e. $T \subseteq \{0,1\}^*$, such that if $uv \in T$, then $u \in T$, and if $u1 \in T$, then $u0 \in T$, $R_\downarrow$ is the daughter-of relation, i.e. $(n,m) \in R_\downarrow$

iff $m = n0$ or $m = n1$, $R_{\rightarrow}$ is the left-sister-of relation, i.e. $(m, n) \in R_{\rightarrow}$ iff $m = s0$ and $n = s1$ for some $s$. A model is a pair $\mathcal{M} = (\mathcal{T}, V)$, such that $\mathcal{T}$ is a tree structure and $V : F \rightarrow \wp(T)$ is a valuation. We define $\mathcal{M}, v \models \varphi$ in the usual way, the only interesting case being:

$$\mathcal{M}, v \models [\pi]\varphi \text{ iff for all } u, \text{ such that } (v, u) \in R_\pi, \mathcal{M}, u \models \varphi$$

and

$$\begin{aligned} R_{\uparrow} &= R_{\downarrow}^{-1} & R_{\pi\cup\sigma} &= R_\pi \cup R_\sigma \\ R_{\rightarrow} &= R_{\leftarrow}^{-1} & R_{\pi;\sigma} &= R_\pi \circ R_\sigma \\ R_{\pi^*} &= R_\pi^* & R_{\varphi?} &= \{(v, v) \mid \mathcal{M}, v \models \varphi\} \end{aligned}$$

where $R^*$ denotes the transitive closure of $R$ and $\circ$ denotes relation composition. □

We can associate terms with tree models by identifying the atomic formulas with features.

**Definition 51** Let $F$ be a finite set of features and $\mathcal{L}$ be a logic. We say that $L \subseteq T_{\Sigma^F}$ is *definable* in $\mathcal{L}$ if there is a formula $\varphi$ in $\mathcal{L}^F$ such that

$$L = \{t \mid t, \varepsilon \models \varphi\}$$

where $\varepsilon$ is the root of the tree. We write $\mathcal{L}_1 \leq \mathcal{L}_2$ if any tree language definable in $\mathcal{L}_1$ is definable in $\mathcal{L}_2$. □

The following two proposition relate tree languages to definability. The first is due to Blackburn and Meyer-Viol (1994) who proved it for a related logic.

**Proposition 47** (Blackburn and Meyer-Viol, 1994) Every local tree language is definable in $\mathcal{L}_{core}$. □

**Proposition 48** (Thatcher and Wright, 1968) A tree language is regular iff it is MSO-definable. □

The following, well-known, inclusions follow primarily from the definition of the three modal logics. Next, we will consider strictness of these inclusions.

**Theorem 49** $\mathcal{L}_{core} \leq \mathcal{L}_{cp} \leq \text{PDL}_{tree} \leq \text{MSO}$

**Proof** The first two inclusions follow from Definition 49. The third inclusion follows from the fact that transitive closure is MSO-definable. □

**Proposition 50** (Schlingloff, 1992) Let $F = \{a, b\}$. The tree language $L_1 \subseteq T_{\Sigma^F}$ such that each tree in $L_1$ contains a path from the root to

a leaf at which exactly one $a$ holds is not $\mathcal{L}_{core}$-definable, but is $\mathcal{L}_{cp}$-definable. □

**Proposition 51** Let $\Sigma = \{\wedge, \vee, 0, 1\}$. The tree language $L_2 \subseteq T_\Sigma$ such that each tree in $L_2$ evaluates to true is not $\mathcal{L}_{cp}$-definable, but is $\mathrm{PDL}_{tree}$-definable.

**Proof** Potthoff (1994) showed that $L_2$ is not definable in an extension of first-order logic with modular counting quantifiers, and since $\mathcal{L}_{cp}$ is equivalent to first-order logic on trees (Afanasiev et al., 2005), the undefinability follows. That $L_2$ is definable in $\mathrm{PDL}_{tree}$ is shown in Afanasiev et al. (2005). □

**Proposition 52** (Kracht, 1999, 2001) Let $F = \{p, q\}$. Let $L_3 \subseteq T_{\Sigma^F}$ where each tree in $L$ is a ternary branching tree such that $p$ is true along a binary branching subtree and $q$ is true at all leaves at which $p$ is true. The language $L_4 \subseteq T_{\Sigma^{\{q\}}}$ obtained from the projection that eliminates $p$ is not $\mathrm{PDL}_{tree}$-definable, but is MSO-definable. □

Next, we will consider how languages that are undefinable in one of these logics can be defined with additional features.

## 16.5 Inessential and Ineliminable Features

The following definition of inessential features is adapted from Kracht (1997). Its purpose is to formalize the concept of a feature whose distribution in a language can be predicted from the other features.

**Definition 52** Let $F$ be a finite set of features, $G \subseteq F$, $L \subseteq T_{\Sigma^F}$, and $\pi : \Sigma^F \to \Sigma^{F-G}$ be a projection. We call the features in $G$ *inessential for* $L$ if the homomorphism $\hat{\pi} : L \to T_{\Sigma^{F-G}}$ based on $\pi$ is one-to-one. □

The intuition for this definition of inessential features is that no two trees in $L$ can be distinguished using features in $G$. Thus, given a tree $t$ in $\hat{\pi}(L)$, we can recover the features from $G$ in $t$ using $\hat{\pi}^{-1}$, since $\hat{\pi}$ is one-to-one. While being an inessential feature is defined with respect to a language, being eliminable is defined with respect to a logic and a language.

**Definition 53** Let $F$ be a finite set of features, $G \subseteq F$, $L \subseteq T_{\Sigma^F}$, $\pi : \Sigma^F \to \Sigma^{F-G}$ be a projection, and $\mathcal{L}$ be a logic. Suppose that $L$ is definable in $\mathcal{L}^F$. We say that $G$ is *eliminable in* $\mathcal{L}$ *for* $L$ if $\hat{\pi}(L)$ is definable in $\mathcal{L}^{F-G}$. □

It should be noted that this definition of eliminability does not coincide with Kracht's (Kracht, 1997), who defines eliminable as being globally explicitly definable. Kracht's definition implies the definition used here,

and thus is stronger. However, since we are interested in *ineliminability*, by contraposition, the definition employed here implies Kracht's definition of ineliminability. Kracht's proof of Proposition 52 depends on the following proposition.

**Proposition 53** (Kracht, 2001) The feature $p$ in Proposition 52 is inessential for $L_3$, but ineliminable in $\mathrm{PDL}_{tree}$. □

We now show how to generalize Kracht's theorem to $\mathcal{L}_{core}$ and $\mathcal{L}_{cp}$:

**Theorem 54** There exists a set of features $F$, a tree language $L \subseteq T_{\Sigma^F}$, and a subset $G \subseteq F$, such that $G$ is ineliminable in $\mathcal{L}_{core}$ (resp. $\mathcal{L}_{cp}$) but eliminable in $\mathcal{L}_{cp}$ (resp. $\mathrm{PDL}_{tree}$).

**Proof** Both of these construction work the same way. Given two of our logics $\mathcal{L}_1, \mathcal{L}_2$, with $\mathcal{L}_1 \leq \mathcal{L}_2$, pick a tree language, $L$, that is not definable in $\mathcal{L}_1$ but is definable in $\mathcal{L}_2$, which exists by Propositions 50 and 51.

By Theorem 49, we know that $L$ is regular, and by Theorem 47, we know that any local tree language is definable in $\mathcal{L}_1$. Given a deterministic FTA $M = (\Sigma, Q, F, \Delta)$, with $L = L(M)$, we can use theorem 46 to construct a local tree language $L' \subseteq T_{\Sigma \times Q}$ such that $\hat{\pi}(L') = L$. Now, the features in $Q$ are inessential, since $M$ is deterministic, but ineliminable, since $L$ is undefinable in $\mathcal{L}_1$. However, since $L$ is definable in $\mathcal{L}_2$, the features in $Q$ are eliminable in $\mathcal{L}_2$. □

The previous theorem can be strengthened in that it can be used to *characterize* the tree languages that are undefinable in some logic $\mathcal{L}_1$ but definable in some other logic $\mathcal{L}_2$, with $\mathcal{L}_1 \leq \mathcal{L}_2$.

**Theorem 55** Any tree language that is not definable in $\mathcal{L}_{core}$ (resp. $\mathcal{L}_{cp}$) but is definable in $\mathcal{L}_{cp}$ (resp. $\mathrm{PDL}_{tree}$) can be defined with additional, inessential features in $\mathcal{L}_{core}$ (resp. $\mathcal{L}_{cp}$) that are not eliminable in $\mathcal{L}_{core}$ (resp. $\mathcal{L}_{cp}$). □

While it was pointed out by Volger (1999) that these and other logics that are used in model theoretic syntax are equivalent modulo a projection, the main contribution of these two theorems is that they connect Volger's observation to Kracht's inessential features. It thus demonstrates the central role that inessential features play in the comparison of logics for model theoretic syntax.

## 16.6 Inessential Features and Non-Deterministic Tree Automata

We now want to consider the definition of inessential features more closely. As was pointed out by Kracht (1997), the purpose of Definition

52 was to formalize the concept of a feature whose distribution is fixed by the other features. We now want to assess whether this formalization captures this concept correctly. For this assessment, the relationship between inessential features and Thatcher's theorem will again play a central role; but this time, we will consider the construction in Theorem 46 using non-deterministic FTAs.

Recall that the observation that Thatcher's theorem yields a language with inessential features depended on the use of a deterministic FTA, since each tree accepted by a deterministic FTA has exactly one accepting run. When we apply Thatcher's construction to non-deterministic tree automata, there can be two different accepting runs for a given tree, and so the added features fail to be inessential in Kracht's sense. However, it is clear that the distribution of the states that are used as extra features can be predicted from the other features, in the sense that we can label a tree that is accepted by a non-deterministic FTA with the states from an accepting run. It's just that there are potentially multiple such accepting runs.

Since bottom-up tree automata can be determinized, these features can be turned into inessential features using the power set construction, and since any inessential feature can be eliminated in MSO (Kracht, 1997), we can now eliminate an essential feature. This observation sheds light on the question whether the fact that certain logics cannot eliminate some inessential features is a strength or a weakness of that logic, i.e. whether or not we want logics for model theoretic syntax to be able to eliminate all inessential features. If we can turn essential features into inessential features and then eliminate them, a logic in which all inessential features can be eliminated may be too strong. This can be seen as support for the use of weaker logics for model theoretic syntax.

It should be noted that lifting the restriction that the homomorphism $\hat{\pi}$ based on a projection $\pi$ be one-to-one in order to extend Kracht's definition can easily make it vacuous, since any feature can be removed with a projection that is not one-to-one. What is needed is a mechanism that captures the essence of the example above. One approach might be to identify an inessential feature with a feature that is *determinizable* in the sense that a feature can be turned into an inessential feature using the power set construction. That this approach is not vacuous can be verified by applying Thatcher's construction to top-down (root-to-frontier) FTAs, which cannot be determinized. We leave the question how this definition of an inessential feature might relate to definability and eliminability as an open problem.

## 16.7 Conclusion

After significant progress in formalizing grammatical theories, one of the more pressing foundational questions in model theoretic syntax right now is how to assess in which logic to carry out this formalization. Since the logics considered here differ only with respect to which inessential features are eliminable, the central question for this assessment is whether the ineliminability of such features is a strength or a weakness of a given logic. It is argued here that, in some cases, ineliminability can be a strength. It would be interesting to consider inessential features from linguistic applications and assess their eliminability in the logics considered here.

## References

Afanasiev, L., P. Blackburn, I. Dimitriou, B. Gaiffe, E. Goris, M. Marx, and M. de Rijke. 2005. PDL for ordered trees. *Journal of Applied Non-Classical Logic* 15(2):115–135.

Blackburn, P. and W. Meyer-Viol. 1994. Linguistics, logic and finite trees. *Logic Journal of the IGPL* 2(1):3–29.

Cornell, Thomas and James Rogers. 2000. Model theoretic syntax. In L. L.-S. Cheng and R. Sybesma, eds., *The GLOT International State-of-the Article Book*. Berlin: de Gruyter.

Kracht, Marcus. 1997. Inessential features. In A. Lecomte, F. Lamarche, and G. Perrier, eds., *Logical aspects of computational linguistics*. Berlin: Springer.

Kracht, Marcus. 1999. *Tools and techniques in modal logic*. Amsterdam: North-Holland.

Kracht, Marcus. 2001. Logic and syntax—a personal perspective. In M. Zakharyaschev, K. Segerberg, M. de Rijke, and H. Wansing, eds., *Advances in modal logic, Vol. 2*. Stanford, CA: CSLI Publications.

Potthoff, Andreas. 1994. Modulo-counting quantifiers over finite trees. *Theoretical Computer Science* 126(1):97–112.

Rogers, James. 1998. *A descriptive approach to language-theoretic complexity*. Stanford, CA: CSLI Publications.

Schlingloff, Bernd-Holger. 1992. On the expressive power of modal logics on trees. In A. Nerode and M. A. Taitslin, eds., *Logical Foundations of Computer Science - Tver '92, Second International Symposium, Tver, Russia, July 20-24, 1992, Proceedings*. Berlin: Springer-Verlag.

Thatcher, J. W. 1967. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences* 1:317–322.

Thatcher, J. W. and J. B. Wright. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory* 2:57–81.

Volger, Hugo. 1999. Principle languages and principle based parsing. In H.-P. Kolb and U. Mönnich, eds., *The Mathematics of Syntactic Structure*. Berlin: de Gruyter.