Decidability and Undecidability in stand-alone Feature Logics

Patrick Blackburn
Department of Philosophy, University of Utrecht
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands
Email: patrick@phil.ruu.nl

Edith Spaan
Department of Computer Science, SUNY at Buffalo
226 Bell Hall, Buffalo, NY 14260, United States of America
Email: spaan@cs.buffalo.EDU

Abstract

This paper investigates the complexity of the satisfiability problem for feature logics strong enough to code entire grammars unaided. We show that feature logics capable of both enforcing re-entrancy and stating linguistic generalisations will have undecidable satisfiability problems even when most Boolean expressivity has been discarded. We exhibit a decidable fragment, but the restrictions imposed to ensure decidability render it unfit for stand-alone use. The import of these results is discussed, and we conclude that there is a need for feature logics that are less homogeneous in their treatment of linguistic structure.

1 Introduction

This paper investigates decidability and undecidability in stand-alone feature logics, that is, feature logics strong enough to express entire grammars without the assistance of a phrase-structure backbone. Our results are predominately negative and seem applicable to most existing stand-alone formalisms. We strengthen a result of [Blackburn and Spaan 1991, 1992] to show that the ability to express re-entrancy and the ability to express generalisations about feature structures interact in ways that lead to undecidability even if most Boolean expressivity has been dropped from the logic. Even our positive results have a negative flavour. We exhibit a decidable fragment, but the restrictions imposed to attain decidability render it incapable of encoding interesting grammars unaided.

But what is the import of such results? This is the question we turn to in the last section of the paper. Basically, we regard such results as a sign that existing feature logics treat linguistic structure too homogeneously. What is needed are feature logics which are more sensitive to the fine structure of linguistic theorising.

The paper is relatively self contained, nonetheless the reader may find it helpful to have [Kasper and Rounds 1986, 1990] and [Blackburn and Spaan 1991, 1992] to hand.

2 Preliminaries

Feature logics are abstractions from the unification based formalisms of computational linguistics. Originally feature logics embodied just one component of unification based formalisms. Early unification formalisms such as GPSG [Gazdar *et al.* 1985] and LFG [Kaplan and Bresnan 1982] have important phrase structure components in addition to their feature passing mechanisms, and the study of feature logic was originally intended to throw light only on the latter. These early unification formalisms are thus highly heterogeneous: they are architectures with roots in both formal language theory and logic.

In recent years this picture has changed. For example, in HPSG [Pollard and Sag 1987] the feature machinery has largely displaced the phrase structure component. Indeed in HPSG the residue of the phrase structure component is coded up as part of the feature system. Logic has swallowed formal language theory, and in effect the entire HPSG formalism is a powerful feature logic, a stand-alone formalism, capable of encoding complex grammars without the help of any other component.[1]

In this paper we are going to investigate the computational complexity of the satisfiability problem for such stand-alone feature logics. This is an important problem to investigate. Natural language grammars are expressed as logical theories in stand-alone formalisms, and sentences are represented as wffs. This means that the problem of deciding whether or not a sentence is grammatical reduces to the problem of building a model of the sentence's logical representation that conforms to all the constraints imposed by the logical encoding of the grammar. In short, the complexity of the satisfiability problem is essentially the worst case complexity of the recognition problem for grammars expressed in the stand-alone formalism.

We will tackle this issue by investigating the complexity of the satisfiability problem for one particular stand-alone formalism, namely $L^{KR\Rightarrow}$. This is the language of Kasper Rounds logic augmented with the strict implication operator. $L^{KR\Rightarrow}$ possesses two of the most fundamental properties of stand-alone formalisms: the ability to express re-entrancy, and the ability to express generalisations about feature structures. It is important to note that $L^{KR\Rightarrow}$ is actually a fairly *minimal* language with these properties; many feature logics can express a lot more besides (for example, set values), thus the negative results for $L^{KR\Rightarrow}$ we present are rather strong: they extend straightforwardly to richer formalisms.

So let's begin by defining $L^{KR\Rightarrow}$. By a signature $\langle \mathcal{L}, \mathcal{S}\rangle$ is meant a pair of non-empty sets $\mathcal{L}$ and $\mathcal{S}$, the set of arc labels and the set of sorts respectively. Syntactically, the language $L^{KR\Rightarrow}$ (of signature $\langle \mathcal{L}, \mathcal{S}\rangle$) contains the following items: an $\mathcal{S}$ indexed collection of propositional symbols (or *sort symbols*); all the standard Boolean operators;[2] an $\mathcal{L}$ indexed collection of distinct unary modalities (that is, *features*); a binary modality $\Rightarrow$; and two special symbols 0 and $\approx$. We use $\approx$ to make *path equations*: given any non-empty sequences A and B consisting of only unary modalities and 0, then $A \approx B$ is a path equation. Intuitively $A \approx B$ says that making the sequence of feature transitions encoded by A leads to the same node as making the transition sequence coded by B. The symbol 0 is a name for the null transition. The *strict implication* operator $\Rightarrow$ will enable us to express generalisations about feature structures.

We make the wffs of $L^{KR\Rightarrow}$ as follows. First, all propositional symbols, all path equations and *True* and *False* are wffs. Second, if ϕ and ψ are wffs then so are all Boolean combinations of ϕ and ψ, so is $\langle l\rangle\phi$ (for all $l \in \mathcal{L}$) and so is $\phi \Rightarrow \psi$. Third, nothing else is a wff. If a wff of $L^{KR\Rightarrow}$ does not contain any occurrences of $\Rightarrow$ then we say it is an L^{KR} wff. Apart from trivial notational changes, the negation free fragment of L^{KR} is the language defined and studied by Kasper and Rounds.[3] That is, the L^{KR} wffs are essentially a way of writing the familiar Attribute Value Matrices (AVMs) in linear format. For example, the following L^{KR} wff:

$$\langle \text{NUMBER}\rangle \textit{plural} \wedge \langle \text{CASE}\rangle(\textit{nom} \vee \textit{gen} \vee \textit{acc})$$

is essentially the following AVM:

$$\begin{bmatrix} \text{NUMBER} & \textit{plural} \\ \text{CASE} & \textit{nom} \text{ or } \textit{gen} \text{ or } \textit{acc} \end{bmatrix}$$

To interpret $L^{KR\Rightarrow}$ we use *feature structures* **M** of signature $\langle \mathcal{L}, \mathcal{S}\rangle$. A feature structure is a triple $\langle W, \{R_l\}_{l\in\mathcal{L}}, V\rangle$, where W is a non-empty set (the set of nodes); each R_l is a binary relation on W that is also a partial function; and V (the valuation) is a function which assigns each propositional symbol $p \in \mathcal{S}$ a subset of W. Note that as we have defined them features structures are merely multi-modal Kripke models,[4] and we often refer to feature structures as models in what follows.

Now for the satisfaction definition. As the symbol 0 is to act as a name for the null transition, in what follows we shall assume without loss of generality that $0 \notin \mathcal{L}$, and we will denote the identity relation on any set of nodes W by R_0. This convention somewhat simplifies the statement of the satisfaction

[1]See [Johnson 1992] for further discussion of the distinction between stand-alone formalisms and formalisms with a phrase structure backbone.

[2]That is, we have the symbols *True* (constant true), *False* (constant false), $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), $\rightarrow$ (material implication) and $\leftrightarrow$ (material equivalence). For the purposes of the present paper it is sensible to assume that all these operators are primitives, as in general we will be working with various subsets of the full language and it would be tedious to have to pay attention to trivial issues involving the interdefinability of the Boolean operators in these weaker fragments.

[3]Computer scientists may have met L^{KR} in another guise. The language of Kasper Rounds logic is a fragment of (deterministic) Propositional Dynamic Logic (PDL) with intersection (see [Harel 1984]). An L^{KR} path equation $A \approx B$ is written as $\langle A \cap B\rangle$ *True* in PDL with intersection.

[4]For further discussion of the modal perspective on feature logic, see [Blackburn and Spaan 1991, 1992].

definition:

$$\begin{array}{llll}
\mathbf{M} \models & p_\alpha[w] & \text{iff} & w \in V(p_\alpha) \\
\mathbf{M} \models & \langle l_1 \rangle \cdots \langle l_k \rangle & \text{iff} & \exists w'(wR_{l_1} \ldots R_{l_k} w' \\
 & \approx \langle l'_1 \rangle \cdots \langle l'_m \rangle [w] & & \&\ wR_{l'_1} \ldots R_{l'_m} w') \\
\mathbf{M} \models & \neg\phi[w] & \text{iff} & \mathbf{M} \not\models \phi[w] \\
\mathbf{M} \models & \phi \vee \psi[w] & \text{iff} & \mathbf{M} \models \phi[w] \text{ or} \\
 & & & \mathbf{M} \models \psi[w] \\
\mathbf{M} \models & \langle l \rangle \phi[w] & \text{iff} & \exists w'(wR_l w' \text{ and} \\
 & & & \mathbf{M} \models \phi[w']) \\
\mathbf{M} \models & \phi \Rightarrow \psi[w] & \text{iff} & \forall w'(\mathbf{M} \models \phi[w'] \\
 & & & \text{implies } \mathbf{M} \models \psi[w'])
\end{array}$$

The satisfaction clauses for *True*, *False*, $\wedge$, $\rightarrow$ and $\leftrightarrow$ have been omitted; these symbols receive their standard Boolean interpretations. If $\mathbf{M} \models \phi[w]$ then we say that $\mathbf{M}$ *satisfies* ϕ at w, or ϕ is *true* in $\mathbf{M}$ at w (where $w \in W$).

The key things to note about this language is that it has both the ability to express re-entrancy (the Kasper Rounds path equality $\approx$ achieves this) and the ability to express generalisations about feature structures (note that $\phi \Rightarrow \psi$ means that at every node where ϕ is true, ψ must also be true). Thus $L^{KR\Rightarrow}$ can certainly express many of the conditions we might want to impose on feature structures. For instance, we might want to impose a sort hierarchy. As a simple example, given sorts *list* and *nelist* (non-empty list) we might wish to insist that every node of sort *nelist* is also of sort *list*. The wff

$$\textit{nelist} \Rightarrow \textit{list}$$

forces this. As a second example, we might want to insist that any node from which it is possible to make a CONSTITUENT-STRUCTURE transition must be of sort *phrasal*. That is, if a node has constituent structure, it is a phrasal node. The wff

$$\langle \text{CONSTITUENT-STRUCTURE} \rangle \textit{True} \Rightarrow \textit{phrasal}$$

forces this. Indeed quite complex demands can be imposed using L^{KR}. For example the following wff embodies the essence of the constraint known as the head feature convention in HPSG:

$$\textit{phrasal} \Rightarrow \langle \text{HEAD} \rangle \approx \langle \text{HEAD-DTR} \rangle \langle \text{HEAD} \rangle.$$

This wff says that at any node of sort *phrasal* in a feature structure, it is possible to make a HEAD transition and it is also possible to make a HEAD-DTR transition followed by a HEAD transition, and furthermore both transition sequences lead to the same node. In view of such examples it doesn't seem wholly unrealistic to claim that L^{KR} has the kind of expressive power a stand-alone feature logic needs.

However $L^{KR\Rightarrow}$ has crossed a significant complexity boundary: it has an undecidable satisfiability problem. This was proved in [Blackburn and Spaan 1991, 1992] using a tiling argument.[5] Now, the result for the full $L^{KR\Rightarrow}$ language is not particularly surprising (undecidability results for related feature logics, can be found in the literature; see [Carpenter 1992] for discussion) but it does lead to an important question: what can be salvaged? To put it another way, are there decidable fragments of L^{KR} that are capable of functioning as stand-alone feature logics? The pages that follow explore this question and yield a largely negative response.

3 Decidability

To begin our search for decidable fragments we will take our cue from Kasper and Rounds' original work. Kasper and Rounds' system was negation free, so the first question to ask is: what happens if we simply remove negation from $L^{KR\Rightarrow}$? Of course, if this is all we do we trivialise the satisfiability problem: it is immediate by induction on the structure of negation free wffs ϕ, that *every* negation free $L^{KR\Rightarrow}$ wff is satisfied in the following model: $M = \langle \{w\}, \{R_l\}_{l \in \mathcal{L}}, V \rangle$ where $R_l = \{\langle w, w \rangle\}$ for all $l \in \mathcal{L}$, and $V(p) = \{w\}$ for all propositional variables p. So we have regained decidability, but in a very uninteresting way.

Now, what made the results of Kasper and Rounds interesting was that not only did they consider the negation free fragment (of L^{KR}), they also imposed certain semantic restrictions. Only extensional models without constant-constant or constant-compound clashes were considered.[6] Will imposing any (or all) of these restrictions make it easier to find decidable fragments of $L^{KR\Rightarrow}$? In fact demanding *extensionality* (that is, working only with models in which each atomic symbol is true at at most one node), does make it easy to find a decidable fragment.

The fragment is the following. We consider wffs of the following form:

$$\phi \wedge (\alpha_1 \Rightarrow \kappa_1) \wedge \cdots \wedge (\alpha_n \Rightarrow \kappa_n).$$

Here ϕ is a metavariable over L^{KR} wffs (that is, ϕ contains no occurrences of $\Rightarrow$); the α_i $(1 \leq i \leq n)$

[5]These papers take the universal modality $\Box$ as primitive rather that $\Rightarrow$, as it is somewhat easier to work with unary modalities. In the presence of full Boolean expressivity $\Box$ and $\Rightarrow$ are interdefinable: $\Box\phi$ is *True* $\Rightarrow \phi$, and $\Rightarrow$ is $\Box(\phi \rightarrow \psi)$. However in what follows we will work with fragments without enough Boolean expressivity to interdefine these operators. As $\Rightarrow$ is the operator we are really interested in we have chosen it as our primitive here.

[6]As Kasper and Rounds showed, introducing this limited form of negation failure results in an NP complete satisfiability problem.

are metavariables over combinations of sort symbols containing only $\vee$ and $\wedge$ as logical operators; and the κ_i $(1 \le i \le n)$ are metavariable over L^{KR} wffs.

Note the general form of the wffs of this fragment. We have an L^{KR} wff ϕ conjoined with n general constraints $\alpha_i \Rightarrow \kappa_i$.[7] The ϕ can be thought of as the AVM associated with some particular natural language sentence, while the wffs of the form $\alpha_i \Rightarrow \kappa_i$ can be thought of as encoding the generalisations embodied in our grammatical theory. Looking for a satisfying model for a wff from this fragment is thus like asking whether the analysis of some particular string of symbols is compatible with a grammar.

The proof that this fragment has a decidable satisfiability problem is straightforward. We're going to show that given any wff Φ belonging to this fragment, there is an upper bound on the size of the models that need to be inspected to determine whether or not Φ is satisfiable. The fact that such an upper bound exists is a direct consequence of three lemmas which we will now prove.

The first lemma we need is extremely obvious, but will play a vital role.

Lemma 3.1 *Let α be any wff containing no logical connectives apart from $\vee$ and $\wedge$. Then in any extensional model, α is satisfied at at most m nodes, where m is the number of distinct sort symbols in α.*

Proof: By induction on the construction of α. □

The importance of this lemma is that it gives us an upper bound on the number of nodes at which the antecedents α_i of the constraints permitted in our fragment can be satisfied.

Next we need a similar result for the L^{KR} wffs of the fragment; that is, for the ϕ and the consequents κ_i of the constraints. As the next two lemmas establish, given any L^{KR} wff ψ which is satisfiable at a node w in some model $\mathbf{M}$, we can always manufacture a very small model $\mathbf{M}|nodes(\psi, w)$ which also satisfies ψ. How we go about defining $\mathbf{M}|nodes(\psi, w)$ is suggested by the following observation: when evaluating a formula in some model, only certain of the model's nodes are relevant to the truth or falsity of the wff; all the irrelevant nodes can be thrown away. What the following two lemmas essentially tell us is that we can manufacture the small models we need by discarding nodes.

The nodes that are relevant when evaluating an L^{KR} wff ψ at a node w in a model $\mathbf{M}$ are the nodes selected by the function *nodes* : WFF $\times$ $W \longrightarrow Pow(W)$ that satisfies the following conditions:

[7] In what follows we refer to the α_i as the antecedents of the constraints, and the κ_i as the consequents.

$$\begin{array}{lcl} nodes(p, w) & = & \{w\} \\ nodes(\neg\psi, w) & = & nodes(\psi, w) \\ nodes(\psi \vee \theta, w) & = & nodes(\psi, w) \cup nodes(\theta, w) \\ nodes(\langle l \rangle\psi, w) & = & \{w\} \cup \bigcup_{w' : wR_l w'} nodes(\psi, w') \end{array}$$

One aspect of the definition of *nodes* may be bothering the reader: there is no clause for the path equations. In fact to give such a clause is rather messy, and it seems better to proceed as follows. Given a wff ψ of L^{KR} we define ψ^* to be the result of replacing every subformula of the form

$$\langle l_1 \rangle \cdots \langle l_k \rangle \approx \langle l'_1 \rangle \cdots \langle l'_m \rangle$$

in ψ by

$$\begin{array}{ll} & \langle l_1 \rangle \cdots \langle l_k \rangle \approx \langle l'_1 \rangle \cdots \langle l'_m \rangle \\ \wedge & \langle l_1 \rangle \cdots \langle l_k \rangle True \\ \wedge & \langle l'_1 \rangle \cdots \langle l'_m \rangle True. \end{array}$$

Clearly ψ is satisfiable at any node in any model iff ψ^* is (all we've done is make the node existence demands encoded in the path equalities explicit). The usefulness of this transformation is simply that the two new conjuncts make available to the simple version of *nodes* defined above all the information hidden in the path equations. From now on we'll assume that all the L^{KR} wffs we work with have been transformed in this fashion.

With these preliminaries out of the way we are ready to proceed. Given a model $\mathbf{M}$, an L^{KR} wff ψ and a node w we form $\mathbf{M}|nodes(\psi, w)$ in the obvious way: the nodes of the model are $nodes(\psi, w)$, and the relations and valuation are the restriction of those of $\mathbf{M}$ to this subset. As the following simple lemma shows, *nodes* indeed picks out the correct nodes:

Lemma 3.2 (Selection Lemma) *For all models* $\mathbf{M}$, *all nodes w of* $\mathbf{M}$ *and all $L^{KR\Rightarrow}$ wffs ψ.*

$$\mathbf{M} \models \psi[w] \quad \textit{iff} \quad \mathbf{M}|nodes(\psi, w) \models \psi[w].$$

Proof: By induction on the structure of ψ. (Note that it follows from the definition of *nodes* that $w \in nodes(\psi, w)$. Once this is observed the induction is straightforward.) □

The selection lemma is a completely general fact about modal languages. It doesn't depend on any special assumptions made in this paper, and in particular it doesn't make any use of the fact that we are only working with models in which each of the R_l is a partial function. Once this additional fact is taken into account, however, we see that $\mathbf{M}|nodes(\psi, w)$ is pleasingly small: there can only be one more node in $\mathbf{M}|nodes(\psi, w)$ than there are occurrences of modalities in ψ. That is, we have:

Lemma 3.3 (Size Lemma) *Let ψ be an L^{KR} wff, and let mod(ψ) be the number of occurrences of modalities in ψ. Then for all models* $\mathbf{M}$ *and all nodes w in* $\mathbf{M}$ *we have that $|nodes(\psi, w) \backslash \{w\}| \leq mod(\psi)$.*

Proof: By induction on the structure of ψ. □

We now have all the pieces we need to establish the decidability result. Using these lemmas we can show that given any wff Φ of our fragment it is possible to place an upper bound on the size of models that need to be checked to determine whether or not Φ is satisfiable. So, suppose Φ is a wff of the form

$$\phi \wedge (\alpha_1 \Rightarrow \kappa_1) \wedge \cdots \wedge (\alpha_n \Rightarrow \kappa_n)$$

that is satisfiable. That is, there is a model $\mathbf{M}$ and a node w in $\mathbf{M}$ such that $\mathbf{M} \models \Phi[w]$. Now, simply forming $\mathbf{M}|nodes(\phi, w)$ is *not* a process guaranteed to make a smaller model satisfying Φ. The problem is that while this model certainly satisfies ϕ, in the course of selecting all the needed nodes we may be forced to select a node that verifies an antecedent α_i of one of the general constraints, but we have no guarantee that we have selected all the nodes needed to make the matching consequent κ_i true.

But this is easy to fix. We must not only form $\mathbf{M}|nodes(\phi, w)$, but in addition, for all i ($1 \leq i \leq n$) we must form $\mathbf{M}|nodes(\alpha_i \wedge \kappa_i, s)$, where s ranges over all the nodes in $\mathbf{M}$ that satisfy α_i. More precisely, we define a new model $\mathbf{M}'$ by taking as nodes all the nodes in all these models (that is, we take the union of all the nodes in all these models) and we define the $\mathbf{M}'$ relations and valuation to be the restriction of the relations and valuation in $\mathbf{M}$ to this subset.

The new model $\mathbf{M}'$ has two nice properties. Firstly, it is clear that it makes ϕ true at w and moreover, whenever it makes one of the α_i true it makes the corresponding κ_i true also. (This follows because of our choice of the nodes of $\mathbf{M}'$; essentially we're making multiple use of the selection lemma here.) Secondly, it is clear that $\mathbf{M}'$ is finite, for its nodes were obtained as a finite union of finite sets. Indeed by making use of lemma 3.1 and the size lemma we can give an upper bound on the size of $\mathbf{M}'$ in terms of the number of symbols in Φ. (This is just a matter of counting the number of general constraints in Φ, the number of distinct propositional variables in the α_i, and the number of modal operators in the ϕ and κ_i; we leave the details to the reader.) Thus the decidability result follows: given a wff Φ of our fragment, bounded search through finite models suffices to determine whether or not Φ is satisfiable.

Alas, this is not a very powerful result. The fragment simply is not expressive enough to function as a stand-alone formalism. Its Achilles heel lies in the strong condition imposed on the α_i. There are two problems. First, because the α_i cannot contain occurrences of features or path equations, many important constraints that stand-alone feature might have to impose cannot be expressed. Second, it is far from clear that the restriction to extensional models is realistic for stand alone formalisms. Certainly if we were trying to capture the leading ideas of HPSG it would not be; the freedom to decorate different nodes with the same sortal information plays an important role in HPSG.

Can some of the restrictions on the α_i be dropped? As the proof of the result shows, there is no obvious way to achieve this: as soon as we allow features or path equations in the α_i, the assumption of extensionality no longer helps us find an upper bound on the number of satisfying nodes, and the proof no longer goes through. Essentially what is needed is a way of strengthening lemma 3.1, but it is hard to find a useful way of doing this. Even imposing an acyclicity assumption on our models doesn't seem to help. As the results of the next section show, this is no accident. The combination of $\approx$ and $\Rightarrow$ is intrinsically dangerous.

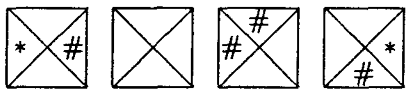
4 Undecidability

The starting point for this section is the undecidability result for the full $L^{KR\Rightarrow}$ language (see [Blackburn and Spaan 1991, 1992]) which was proved using reduction from a certain undecidable *tiling problem*. We're going to strengthen this undecidability result, and we're going to do so by using further tiling arguments. As the use of tiling arguments seem to be something of a novelty in the computational linguistics literature, we include a little background discussion of the method.

Tiling arguments are a well known proof technique in computer science for establishing computability and complexity results. (In fact, tiling arguments are used to introduce the basic concepts of complexity, decidability and undecidability in [Lewis and Papadimitriou 1981], one of the standard introductions to theoretical computer science.) They are also a popular method for analysing the complexity of logics; both [Harel 1983] and [Harel 1986] are excellent guides to the versatility of the method for this application.

One of the most attractive aspects of tiling problems is that they are extremely simple to visualise. A tile T is just a 1×1 square, fixed in orientation, that has coloured edges *right*(T), *left*(T), *up*(T), and *down*(T) taken from some denumerable set. A tiling problem takes the following form: given a finite set T

of tile types, can we cover a certain part of $\mathbf{Z} \times \mathbf{Z}$ ($\mathbf{Z}$ denotes the integers) using only tiles of this type, in such a way that adjacent tiles have the same colour on the common edge, and such that the tiling obeys certain constraints? For example, consider the following problem. Suppose $\mathcal{T}$ consists of the following four types of tile:



Can an 8 by 4 rectangle be tiled with the fourth type of tile placed in the left hand corner? The answer is 'yes' — but we'll leave it to the reader to work out how to do it.

There exist complete tiling problems for many complexity classes. In the proof that follows we make use of a certain Π^0_1 complete tiling problem, namely the problem of tiling the entire positive quadrant of the plane, that is, the problem of tiling $\mathbf{N} \times \mathbf{N}$ where $\mathbf{N}$ is the set of natural numbers.

We begin with the following remark: by inspection of the undecidability proof for $L^{KR\Rightarrow}$ in [Blackburn and Spaan 1991, 1992], it is immediate that we still have undecidability if we restrict the language to formulas that consist of a conjunction of formulas of the form $\phi_1 \Rightarrow \phi_2$, where ϕ_1 and ϕ_2 are L^{KR} formulas with negations applied to atoms only, and ϕ_2 is satisfiable. (The stipulation that ϕ_2 must be satisfiable prevents it from playing the role of *False* and thus smuggling in illicit negations.) Call this language L^-. Let's see if we can strengthen this result further.

So, suppose we look at L^- formulas with $\vee$ as the only binary boolean connective in ϕ_1 and ϕ_2. In this case, we show that the corresponding satisfiability problem is still undecidable by constructing another reduction from $\mathbf{N} \times \mathbf{N}$ tiling.

Let $\mathcal{T} = \{T_1, \ldots, T_k\}$ be a set of tiles. We construct a formula ϕ such that:

$$\mathcal{T} \text{ tiles } \mathbf{N} \times \mathbf{N} \text{ iff } \phi \text{ is satisfiable.}$$

First of all we will ensure that, if ϕ is satisfiable in a model $\mathbf{M}$, then $\mathbf{M}$ contains a gridlike structure. The nodes of $\mathbf{M}$ (henceforth W), play the role of points in a grid, R_r is the right successor relation, and R_u is the upward successor relation. Define:

$$\phi_{grid} = (\mathit{True} \Rightarrow \langle r\rangle\langle u\rangle \approx \langle u\rangle\langle r\rangle).$$

Clearly ϕ_{grid} forces gridlike models.

Next we must tile the model. To do this we use propositional variables $t_1, \ldots, t_k$, such that t_i is true at some node w, iff tile T_i is placed at w. To force a proper tiling, we need to satisfy the following three requirements:

1. There is exactly one tile placed at each node.

$$\phi_1 = (\mathit{True} \Rightarrow \bigvee_{i=1}^{k} t_i) \wedge \bigwedge_{1\leq i<j\leq k} (t_i \Rightarrow \neg t_j)$$

2. If T_i is the tile at w, and T_j is a tile such that $\mathit{right}(T_i) \neq \mathit{left}(T_j)$, then t_j should not be true at any R_r successor of w:

$$\phi_2 = \bigwedge_{\mathit{right}(T_i)\neq\,\mathit{left}(T_j)} (t_i \Rightarrow \langle r\rangle \neg t_j)$$

3. Similarly for up successors:

$$\phi_3 = \bigwedge_{\mathit{up}(T_i)\neq\,\mathit{down}(T_j)} (t_i \Rightarrow \langle u\rangle \neg t_j)$$

Let ϕ be $\phi_{grid} \wedge \phi_1 \wedge \phi_2 \wedge \phi_3$. It is not too difficult to prove that ϕ is satisfiable iff $\mathcal{T}$ tiles $\mathbf{N} \times \mathbf{N}$, which implies that the satisfiability problem for our fragment of L^- is undecidable.

Are there weaker undecidable fragments? Yes: we can remove *True* from ϕ. We do this by using a new propositional variable p_T which plays the role of *True*. Insisting that

$$p_T \wedge (p_T \Rightarrow [r]p_T) \wedge (p_T \Rightarrow [u]p_T)$$

ensures that p_T behaves like *True*.

Are even weaker fragments undecidable? Yes: we can ensure that $\vee$ occurs at most once in each clause. In fact we only have to rewrite part of ϕ_1 (namely, $\mathit{True} \Rightarrow \bigvee_{i=1}^{k} t_i$), for this is the only place in ϕ where $\vee$ occurs. We use new variables $b_2, \ldots, b_{k-1}$ for this purpose and we ensure that b_i is true iff t_j is true for some $j \leq i$. We do this as follows:

$$\begin{array}{rl}
(b_2 \Rightarrow t_1 \vee t_2) & \wedge \\
(b_3 \Rightarrow b_2 \vee t_3) & \wedge \\
 & \vdots \\
(b_{k-1} \Rightarrow b_{k-2} \vee t_{k-2}) & \wedge \\
(\mathit{True} \Rightarrow b_{k-1} \vee t_k) &
\end{array}$$

Clearly this has the desired effect.

5 Discussion

The results of this investigation are easy to summarise: the ability to express both re-entrancy and generalisations about feature structures lead to algorithmically unsolvable satisfiability problems even if most Boolean expressivity is dropped. What are the implications of these results?

Stand-alone feature formalisms offer (elegant) expressive power in a way that is compatible with the lexically driven nature of much current linguistic theorising. One of their disadvantages (at least in their current incarnations) is that they tend to hide computationally useful information. For example, as [Johnson 1992] points out, it is difficult even to formulate such demands as offline parsability for existing stand-alone formalisms; the configurational information required is difficult to isolate. The problem is that stand-alone formalisms tend to be too homogeneous. It is certainly elegant to treat information concerning complex categories and configurational information simply as 'features'; but unless this is done sensitively it runs the risk of 'reducing' a computationally easy problem to an uncomputable one.

Now, much current work on feature logic can be seen as attempts to overcome the computational bluntness of stand-alone formalisms by making visible computationally useful structure. For example, recent work on typed feature structures (see [Carpenter 1992]) explicitly introduces the type inheritance structure into the semantics; whereas in [Blackburn *et al.* 1993] composite entities consisting of trees fibered across feature structures are constrained using two distinct 'layers' of modal language. What is common to both these examples is the recognition that linguistic theories typically have subtle internal architectures. Only when feature logics become far more sensitive to the fine grain of linguistic architectures will it become realistic to hope for general decidability results.

Acknowledgements We would like to thank the anonymous referees for their comments on the abstract. Patrick Blackburn would like to acknowledge the financial support of the Netherlands Organization for the Advancement of Research (project NF 102/62-356 'Structural and Semantic Parallels in Natural Languages and Programming Languages').

References

[Blackburn and Spaan 1991] Blackburn, P. and Spaan, E.: 1991, On the Complexity of Attribute Value Logics. *Proceedings of the Eighth Amsterdam Colloquium*, edited by P. Dekker and M. Stokhof, Philosophy Department, Amsterdam University, The Netherlands.

[Blackburn and Spaan 1992] Blackburn, P. and Spaan, E.: 1992, A Modal Perspective on the Computational Complexity of Attribute Value Grammar. To appear in *Journal of Logic, Language and Information.*

[Blackburn *et al.* 1993] Blackburn, P., Gardent, C., and Meyer-Viol, W.: 1993, Talking about Trees. This volume.

[Carpenter 1992] Carpenter, B.: 1992, *The Logic of Typed Feature Structures*, Cambridge University Press.

[Gazdar *et al.* 1985] Gazdar, G.: Klein, E., Pullum, G., and Sag, S.: 1985, *Generalised Phrase Structure Grammar.* Basil Blackwell.

[Harel 1983] Harel, D.: 1983, Recurring dominoes: making the highly undecidable highly understandable, in *Proc. of the Conference on Foundations of Computing Theory*, Springer Lecture Notes in Computer Science **158**, 177-194.

[Harel 1984] Harel, D.: 1984, Propositional Dynamic Logic, in *Handbook of Philosophical Logic*, **2**, edited by D. Gabbay and F. Guenthner, Reidel.

[Harel 1986] Harel, D.: 1986, Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness, *Journal of the ACM*, **33(1)**, 224–248.

[Johnson 1992] Johnson, M.: 1992, Computing with features as formulas, manuscript, Cognitive and Linguistic Sciences Department, Brown University.

[Kasper and Rounds 1986] Kasper, R. and Rounds, W.: 1986, A logical semantics for feature structures, in *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, Columbia University, New York, 257–266.

[Kasper and Rounds 1990] Kasper, R. and Rounds, W.: 1990, The Logic of Unification in Grammar, *Linguistics and Philosophy* **13**, 33–58.

[Kaplan and Bresnan 1982] Kaplan, K. and Bresnan, J.: 1982, Lexical functional grammar: A formal system for grammatical representation, in *The Mental Representation of Grammatical Relations*, edited by Joan Bresnan, MIT Press, 173–281.

[Lewis and Papadimitriou 1981] Lewis, H. and Papadimitriou, C.: 1981, *Elements of the Theory of Computation*, Prentice-Hall.

[Pollard and Sag 1987] Pollard, C. and Sag, I.: *Information-Based Syntax and Semantics: Volume 1 - Fundamentals.* CSLI Lecture Notes, **13**, Stanford.