THE MONOTONOUS ELIMINATION OF PREDICATE VARIABLES

Harold Simmons
Department of Computer Science
The University
Manchester M13 9PL

January 20, 1994

Abstract

I describe an algorithm which when given a sentence Γ involving universally quantified predicate variables, first reduces this sentence to a normal form Γ^* and then, if Γ^* satisfies certain extra restrictions, eliminates the predicate variables of Γ^* to produce a predicate-free equivalent of Γ. This process may involve the introduction of skolem functions but in many cases does not (and so produces a more elementary version of Γ).

The algorithm eliminates the predicate variables by witnessing them with certain definable predicates. The choice of these predicates is guided by the shape of the normal form Γ^*.

The algorithm was initially designed to deal with modal correspondence problems. In this respect it is an extension of Sahlqvist's algorithm which produces first order equivalents of a wide class of modal formulas. The algorithm is illustrated by several examples from modal logic.

1 Introduction

Many interesting properties of structures can be defined by a sentence of the shape

$$(\forall P_1, \ldots, P_m)[M] \tag{1}$$

where $P_1, \ldots, P_m$ are predicate variables constrained to range over the subsets of certain domains, and the matrix M is built up from $P_1, \ldots, P_m$ and other symbols. There is a tacit assumption here that, in some way, M is essentially simpler in kind than the full sentence (1). For instance, the Π^1_1 sentences are those sentences (1) where M is first order.

It is known that many of these sentences (1) are equivalent to predicate free sentences, i.e. sentences built up without the use of predicate variables (but perhaps using other linguistic gadgets). There are also many sentences (1) which do not have such equivalents. Given a sentence (1), how can we determine whether or not it does have a predicate free equivalent, and how can we find an equivalent when it exists?

For instance, in modal logic the standard semantics of a modal formula can be phrased as a Π^1_1 sentence. Correspondence theory [1] is concerned with finding first order equivalents of such sentences. Of course, not every modal formula has such a first order equivalent (otherwise modal logic would be of little use). A standard result due to Sahlqvist [2] describes a wide class of formulas with first order equivalents, and an analysis of this result (as given in [3] or [4]) provides an algorithm for producing such equivalents.

An important observation is that many sentences (1) are equivalent to some of its particular instances obtained by witnessing the predicate variables by particular predicates. In correspondence theory this is the essence of the 'substitution method'.

I will describe an algorithm which, given an initial sentence (1) will first reduce the sentence to a certain normal form and then, if appropriate extra conditions are satisfied, will produce predicate free witnesses which give a predicate free equivalent.

There are several aspects of this algorithm worth noting.

The general outline of the algorithm is derived from an analysis of Sahlqvist's result however the applicability of the algorithm is not restricted to modal logic. Furthermore, the algorithm is not concerned only with finding first order equivalents, in many cases there are acceptable higher order equivalents where no first order one exists, and the algorithm will search for these.

The reduction phase of the algorithm is a refined version of the usual process of reduction to skolem normal form (which involves the introduction of skolem variables). However, various parts of the sentence play different roles, and some of these need not be skolemized. The reduction phase is organized so that a skolem variable is introduced only when absolutely necessary.

At the heart of the algorithm is a use of monotonicity in the predicate variables to determine a 'kernel' substitution which, in a certain sense, generates the whole class of simplifying substitutions. This idea goes back to Ackermann in [5] and [6]. The desired normal forms of the reduction phase are designed so that these 'kernel' substitutions are almost immediate.

Here is an overview of the algorithm.

- The global strategy is to search for a 'kernel' substitution with which the predicate variables may be witnessed. For certain 'nice' sentences such a 'kernel' is immediately recognizable. These sentences have a 'source' from which the 'kernel' is extracted, and a 'target' into which this 'kernel' is sent. The notion of a 'kernel' is made precise in Definition 3, and a description of the effect of the substitution is given in Lemma 4.

- To find a kernel substitution for an arbitrary sentence we perform several rephrasings which take the shape of the sentence towards that of a 'nice' sentence. This is the reduction phase of the algorithm. During this reduction process different parts of the sentence are treated differently (for they have different jobs to do). To handle the reduction steps it is convenient to label these parts. This is done through the notion of a 'special interest sentence' given in Definition 6.

- The reduction steps are designed to take the given sentence to a normal form. If successful this reduction achieves a 'kernel normal form' as given in Definition 11(b). Any sentence in kernel normal form is 'nice' and the method of extracting a kernel is given in Theorem 12.

- The reduction process is described in Section 7. Each reduction step is a simple rephrasing effected by one of the standard manipulations used to produce a skolem normal form. The most complicated of these involves the introduction of a skolem variable to transfer a quantifier prefix

  $$(\forall x \cdots)(\exists y \cdots)[\cdots x \cdots y \cdots] \quad \text{into} \quad (\exists f \cdots)(\forall x, y \cdots)[fx = y \rightarrow \cdots x \cdots y \cdots]$$

  in the usual way. Unlike the skolem normal form, the kernel normal form does not require all quantifiers to be at the front. As the reduction proceeds it becomes clear that the internal structure of certain parts of the sentence need not be analysed. These parts are then left intact and deposited in certain 'configurations' or the target. The notion of a special interest sentence is, to some extent, designed to cope with these undisturbed parts.

The paper is written in two parts. In the first part I describe the general features of the algorithm. To some extent these descriptions can be read independently; in fact I recommend that you do not read them in a linear fashion, but skip from one to another until you get a global view of the whole algorithm. This first part also contains several preliminary examples illustrating the algorithm.

In the second part I give a batch of examples to show how the algorithm works when applied to modal logic. When considered in the abstract the algorithm can look rather dry and technical. A study of these or similar examples will help you understand the finer points of the algorithm.

For your convenience here is a brief description of the contents of the remaining sections.

PART A: generalities

2. Language preliminaries

   The algorithm can be used in a vast collection of different languages. These can have quite high order features. To fix ideas I describe a family of languages for which I expect the algorithm to be most frequent used.

3. Kernel substitution

   This gives the basic notions and technique of the substitution method and proves the fundamental result which supports the whole algorithm.

4. Special interest sentences

   The algorithm requires that sentences are written in a certain format. This section describes this shape and gives several examples.

5. Kernel extraction

   Every sentence in a certain normal form has a kernel substitution which is easily extracted from its shape.

6. An example in circumscription

   A simple example that doesn't require the details of the reduction phase.

7. Reduction phase

   The full details of how sentences are reduced to normal form.

8. Skolem free equivalents

   If the initial sentence has a certain restricted shape, then the algorithm is guaranteed to produce a kernel substitution. Furthermore, the resulting predicate-free equivalent does not contained any introduced skolem variables.

PART B: applications to modal logic

9. Modal background

This contains a brief description of the modal language and how the semantics of a modal formula produces a sentence to which the algorithm can be applied. This gives a concrete example of a language of the family described in Section 2.

10. Sahlqvist's result

This shows how Sahlqvist's result is an immediate consequence of the observation of Section 8 (or how the whole algorithm is a generalization of Sahlqvist's method).

11. Modal examples

This is an extensive list of examples from modal logic which are used to illustrate the various features of the algorithm.

12. Final comments

Here I give a few remarks which may help stimulate further research on this topic.

Acknowledgments

The impulse that prompted me to write down the algorithm was a conversation with Dov Gabbay. He told me of his joint work with Hans Jürgen Ohlbach concerning an implemented algorithm SCAN, [7], which also produces equivalents to Π^1_1 sentences (1). This algorithm takes the negation

$$(\exists P_1, \dots, P_m)[\neg M]$$

of the given sentence and, by the introduction of skolem functions, transforms the body $\neg M$ into clausal form. SCAN then uses resolution techniques to eliminate (where possible) the predicate variables $P_1 \dots, P_m$. Of course, like all these algorithms, SCAN is only partially successful. Another algorithm with the same objective is described in [8]. This eliminates the predicate variables using a version of the Ackermann trick. Both of these algorithms have influenced the design of the one presented here.

I am grateful to Dov Gabbay, Hans Jürgen Ohlbach, and Andrzej Szalas for providing me with preprints of [7] and [8], and for stimulating my interest in these matters. In particular these preprints informed me of the history of Ackermann's trick (which previously I had known only in the context of correspondence theory). I am also grateful to Johan van Benthem for several pertinent comments on an earlier version of this work. These enabled me to improve the content and presentation of the algorithm.

This paper is a revised version of the technical report [9]. For the most part these revisions are the result of the referees' comments on that submitted version of the paper.

PART A: generalities

2 Language preliminaries

Let

$$\mathsf{Lang}$$

be a language suitable for describing some of the properties of structures of a given signature. Lang may be the canonical first order language associated with this signature, or could be any of its several enrichments, or it could be something more exotic. To help you fix your ideas I will restrict the nature of Lang, however, it is possible that there are applications of the algorithm where a broader kind of language is needed.

A signature is given by a collection of sorts and specified operation and relation symbols between these sorts. We are working in a classical, naive, set-theoretic setting, so these sorts are to be interpreted by sets and the operation and relation symbols interpreted by their set theoretic counterparts. The first order language associated with this signature will have names for these operation and relation symbols, and stocks of variables for each sort where each such variable is constrained to range over that sort.

The full finitary type extension of this signature is obtained by closing off the sorts under cartesian products and function spaces (to produce the types), and then adding in the various associated operations (projection maps, evaluation maps, etc). This type extension has an associated higher order language which has stocks of variables constrained to range over the various types and names for the associated operations. I will refer to this kind of variable as an 'inhabitant variable' (for each is constrained to range over the inhabitants of a particular type). Notice that in this higher order language skolem functions are easily handled.

The language Lang may be the associated first order language, or it may be the full higher order language, or it may be something in between (obtained, for instance, by restricting the allowable types).

Whatever Lang is, we may enrich it in a different direction. Thus for certain selected types, we may add stocks of predicate variables constrained to range over subsets of those types. Thus for each such predicate variable P and inhabitant variable u, both associated with the same type, we have a new atomic formula $P(u)$. These can be combined with existing formulas to produce new formulas M containing several predicate variables, and possibly some free inhabitant variables. Suppose that M does not have any such free inhabitant variables (i.e. all occurring inhabitant variables are quantified), we may then quantify out the predicate variables to obtain a sentence (1). In general such sentences capture properties of structures which are not expressible in Lang.

Let

$$\mathsf{PLang}$$

be this predicate enrichment of Lang. I am concerned here with the question of finding ways of rephrasing (1) without the use of predicate variables. In some cases (1) is equivalent to a Lang sentence. In other cases (1) is equivalent to a sentence of a language

$$\mathsf{SLang}$$

obtained from Lang by adding appropriate skolem constructs (so that SLang is a part of the full higher order language described above). In other cases there may be no useful way to eliminate the predicate variables of (1). I will describe an algorithm with the following properties.

> Given an appropriate sentence Π of Plang, as in (1), the algorithm will reduce Π to an equivalent normal form Π^*, and then, if the shape of Π^* is of a suitable restricted kind, will produce a SLang sentence Σ which is equivalent to Π. In many cases Σ is actually a Lang sentence.

(In fact, for technical reasons the algorithm deals with finite collections of sentence (1), but this need not distract us here.)

Assuming that the PLang sentence Π does have a SLang equivalent Σ, where should we look for it? Well, to begin with, we look in the easiest places.

By a *predicate witness* for a predicate variable P I mean a formula $K(u)$ of SLang with a distinguished free inhabitant variable u of the same type as P. Suppose

$$K_1(u_1), \dots, K_m(u_m)$$

is a sequence of predicate witnesses for the predicate variables

$$P_1, \dots, P_m.$$

Take the matrix M of Π and replace each occurrence of

$$P_i(u_i) \text{ by the corresponding witness } K_i(u_i)$$

where, of course, care is taken to avoid variable capture. This (after doing something about the free inhabitant variables which may be introduced by this substitution) will produce a SLang sentence Σ for which the implication

$$\Pi \Rightarrow \Sigma$$

is universally valid. But now we find that, under certain circumstances and with a suitable choice of $K_1, \dots, K_m$, the implicant Σ is actually an equivalent of Π. This is the essence of what is called the 'substitution method'.

The algorithm I describe will, when successful, automatically produce these 'kernel' substitutions. When the algorithm fails, it will still give some information about the obstructions to such a substitution.

Notice that the skolem language SLang is actually a candidate for the base language Lang, and so has its own predicate enrichment PSLang. In fact, most of the manipulations will take place within this amalgam PSLang of Slang and PLang.

3 Kernel substitution

To use the algorithm and understand when and why it works we need to treat different parts of sentence (1) in different ways. To do this we format (1) in a special way. I will describe this in terms of various restrictions on the sentence, although every sentence (1) can be formatted in a way that meets these restrictions.

I will approach these restrictions step by step.

We fix once and for all the sequence

$$\mathsf{P} \ ::= \ P_1, \ldots, P_m$$

of predicate variables. At times it is convenient to display the occurrences of these variables in a formula. Thus we write

$$M(P_1, \ldots, P_m) \quad \text{or} \quad M(\mathsf{P}) \quad \text{for} \quad M.$$

In particular, we are concerned with sentences

$$\Pi \ ::= \ (\forall \mathsf{P})[M(\mathsf{P})].$$

The matrix M of this sentence must conform to a certain pattern.

Firstly, M will begin with an initial sequence of universally quantified inhabitant variables a. We often refer to these as the *parameters* of Π. These variables will be restricted by some finite collection $\mathcal{A}$ of SLang formulas, i.e. formulas containing skolem variables but not containing predicate variables. Thus Π has the shape

$$\Pi \ ::= \ (\forall \mathsf{P})(\forall \mathsf{a})[\bigwedge \mathcal{A} \to N(\mathsf{P})]$$

for some PSLang formula N (whose shape I will deal with later). It will help us if we write this sentence as

$$\Pi \ := \ (\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[N(\mathsf{P})] \tag{2}$$

to indicate that a is restricted by $\mathcal{A}$. Much of the algorithm is concerned with manipulating these restrictions $\mathcal{A}$. (Because of its position in Π, I call $\mathcal{A}$ an 'outer configurations'. A precise definition of this notion is given later.) Notice that the configuration $\mathcal{A}$ or the whole quantification prefix $(\forall \mathsf{a}$ with $\mathcal{A})$ may be empty. Thus particular cases of (2) are

$$(\forall \mathsf{P})(\forall \mathsf{a})[N(\mathsf{P})] \quad \text{and} \quad (\forall \mathsf{P})[N(\mathsf{P})]$$

i.e. the existence of this prefix is no restriction at all, merely a device for handling the sentence.

The formula N is written as an implication

$$S \to T$$

for certain formulas S and T called the 'source' and 'target' of Π. The algorithm is driven by the shape of the source S. As it proceeds, the algorithm will collect suitable parts of S and deposit them in the target T (using the obvious transfer across an implication), and so extract a certain 'core' from S. It is this core that will give us the required 'kernel' substitution.

To describe the required property of $T = T(\mathsf{P})$ we need a bit of notation and terminology.

Let

$$\mathsf{Q} \ ::= \ Q_1, \ldots, Q_m$$

be a sequence of fresh predicate variables matching P (in length and in types of components). We write

$$[\mathsf{Q} \leq \mathsf{P}]$$

to abbreviate the sentence

$$\bigwedge\{(\forall u_i)[Q_i(u_i) \to P_i(u_i) \mid 1 \le i \le m].$$

Given the formula $T(\mathsf{P})$ let $T(\mathsf{Q})$ be the result of substituting Q for P throughout T. Notice that both $T(\mathsf{P})$ and $T(\mathsf{Q})$ depend on the parameters a.

1 DEFINITION. The formula $T(\mathsf{P})$ is *monotone* if the PSLang sentence

$$(\forall \mathsf{Q}, \mathsf{P})(\forall \mathsf{a})[[\mathsf{Q} \le \mathsf{P}] \to [T(\mathsf{Q}) \to T(\mathsf{P})]]$$

is universally valid. The formula is *antitone* if its negation is monotone. ■

Certain monotone formulas can be recognized by their syntactic shape, namely, those that are positive (i.e. contain no negative effects of negation and implication). It is not necessary to make this later notion precise, however, observe that monotone formulas are closed under conjunction, disjunction, and quantification of inhabitant variables.

Of course the object of the exercise is not to replace predicate variables by other predicate variables, but by predicate witnesses. Thus let

$$\mathsf{K} \ ::= \ K_1, \dots, K_m$$

be a sequence of predicate witnesses matching P. We write

$$[\mathsf{K} \le \mathsf{P}]$$

to abbreviate the sentence

$$\bigwedge\{(\forall u_i)[K_i(u_i) \to P_i(u_i) \mid 1 \le i \le m].$$

and denote the result of substituting K for P throughout T by $T(\mathsf{K})$. Thus $T(\mathsf{K})$ is a SLang formula in the parameters a.

2 LEMMA. *Suppose $T(\mathsf{P})$ is monotone and let K be a sequence of predicate witnesses matching P. Then the sentence*

$$(\forall \mathsf{P})(\forall \mathsf{a})[[\mathsf{K} \le \mathsf{P}] \to [T(\mathsf{K}) \to T(\mathsf{P})]] \tag{3}$$

is universally valid. ■

We are now in a position to describe the essence of the 'substitution' method. (However, the details of the algorithm will require further restrictions on the source S.)

3 DEFINITION. A *kernel* for the formula $S(\mathsf{P})$ (the source in Π) is a sequence K of predicates (as above) such that both

$$(\forall \mathsf{a})[S(\mathsf{K})] \tag{4}$$

and

$$(\forall \mathsf{P})(\forall \mathsf{a})[S(\mathsf{P}) \to [\mathsf{K} \le \mathsf{P}]] \tag{5}$$

are universally valid. We also speak informally of a kernel of a sentence when we formally mean a kernel of its source. ■

Observe that (4) and (5) are both implied by the single sentence

$$(\forall \mathsf{P})(\forall \mathsf{a})[S(\mathsf{P}) \leftrightarrow [\mathsf{K} \leq \mathsf{P}]]. \tag{6}$$

The aim of the reduction phase of the algorithm will be to rephrase the source in the form $[\mathsf{K} \leq \mathsf{P}]$ for appropriate K (and hence achieve this last equivalence). Once this has been done we may apply the following result.

This is the pivot around which the whole algorithm swings.

4 LEMMA. *For a sentence Π, suppose that the source $S(\mathsf{P})$ has a kernel K. Let*

$$\Sigma \ ::= \ (\forall \mathsf{a} \text{ with } \mathcal{A})[T(\mathsf{K})]$$

(so that Σ is a SLang *sentence.) Then the equivalence*

$$\Pi \ \Leftrightarrow \ \Sigma$$

is universally valid.

Proof. Firstly, we have

$$\Pi \ \Rightarrow \ (\forall \mathsf{a} \text{ with } \mathcal{A})[S(\mathsf{K}) \rightarrow T(\mathsf{K})] \ \Rightarrow \ \Sigma$$

where the first implication follows by instantiation and the second follows by (4).

For the converse, since K is a kernel of S and since T is monotone, we have both (5) and (3) and these combine to give

$$(\forall \mathsf{P})(\forall \mathsf{a})[S(\mathsf{P}) \rightarrow [T(\mathsf{K}) \rightarrow T(\mathsf{P})]]$$

which may be rephrased as

$$(\forall \mathsf{P})(\forall \mathsf{a})[T(\mathsf{K}) \rightarrow [S(\mathsf{P}) \rightarrow T(\mathsf{P})]].$$

By some simple quantification manipulation this may be weakened to

$$(\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[T(\mathsf{K})] \ \Rightarrow \ (\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[S \rightarrow T]$$

i.e. to

$$(\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[T(\mathsf{K})] \ \Rightarrow \ \Pi.$$

But P does not occur in $T(\mathsf{K})$ nor in $\mathcal{A}$, so this sentence can be simplified to

$$(\forall \mathsf{a} \text{ with } \mathcal{A})[T(\mathsf{K})] \ \Rightarrow \ \Pi.$$

which is

$$\Sigma \ \Rightarrow \ \Pi$$

as required. ■

When does such a kernel exists and how can we find it? The aim of the reduction phase is to make the existence and shape of the kernel obvious.

4 Special interest sentences

To find a kernel substitution for a sentence Π (provided one exists) we proceed by a sequence of simple rephrasings towards a sentence

$$(\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[S \to T]$$

where for the source S an equivalence (6) can be read off (or it can be seen that there is no such equivalence). The rephrasings are effected by attacking the source with standard tools for obtaining a prenex normal form (i.e. pulling quantifiers to the front, introducing skolem functions, using a conjunctive or a disjunctive normal form, etc), but with two modifications.

- Firstly, wherever possible a part of the source is transferred into the target (with the obligation that the resulting target must be monotone). This part is then left to reside undisturbed in the target (no matter how complicated the part may be).
- Secondly, various other parts of the source, in particular many P-free parts, are absorbed into one of several configurations (as part of a quantifier prefix), and then left undisturbed. Also, the outer configuration will house the required restrictions on the introduced skolem variables.

The formatting of the sentence Π in a certain way is designed to facilitate the handling of the various parts. In the absence of a more colourful terminology I will refer to this kind of sentence as one of 'special interest'. Before I can define this notion I need to say precisely what a configuration is.

5 DEFINITION. For a given sequence b of inhabitant variables, a *configuration over* b is a finite collection $\mathcal{X}$ of SLang formulas all of whose free variables are taken from b. In particular, no predicate variables appear in $\mathcal{X}$. ■

Given such a configuration $\mathcal{X}$, a subsequence x of b, and a PSLang formula H, we may form the PSLang formulas

$$(\forall \mathsf{x})[\bigwedge \mathcal{X} \to H] \quad , \quad (\exists \mathsf{x})[\bigwedge \mathcal{X} \wedge H]$$

As above, it will be suggestive if we write these as

$$(\forall \mathsf{x} \text{ with } \mathcal{X})[H] \quad , \quad (\exists \mathsf{x} \text{ with } \mathcal{X})[H]$$

respectively.

During the workings of the algorithm we will meet both

outer configurations and inner configurations

where here 'outer' and 'inner' refer to the positions in which these configurations occur in Π. In particular, Π has just one outer configuration, the one shown in (2), but the source formula S will contain several subformulas L each with its own inner configuration.

There is a qualitative difference between inner and outer configurations which is a little hard to pin down. Perhaps it is best not to attempt to describe this distinction, but leave you to observe the difference when the reduction phase of the algorithm is described.

We are now in a position to describe the required shape of Π.

6 DEFINITION. A *special interest* sentence is a PSLang sentence of the shape

$$\Pi ::= (\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[S \to T] \tag{7}$$

where:

- P is a sequence of predicate variables;
- a is a possibly empty sequence of inhabitant variables (called the *parameters* of Π), and $\mathcal{A}$ is an outer configuration in these variables;
- the formula S (called the *source* of Π) is the conjunction

  $$S ::= \bigwedge \mathcal{L}$$

  of a finite set $\mathcal{L}$ of formulas (called the *laminar components* of Π) where each such component L has the shape

  $$L ::= (\forall \mathsf{x} \text{ with } \mathcal{X})[\bigvee \mathcal{D}] \tag{8}$$

  where

  - x is a possibly empty sequence of inhabitant variables, and $\mathcal{X}$ is an inner configuration in these variables and the parameters a,
  - $\mathcal{D}$ is a finite set of PSLang formulas (called the *disjuncts* of L) in which no predicate variable occurs bound;
- the formula T (called the *target* of Π) is monotone.

(In practice the target T will not contain bound predicate variables.) ■

The shape of special interest sentences has been designed to help organize the workings of the algorithm. Typically we will start with a sentence

$$\Pi ::= (\forall \mathsf{P})(\forall \mathsf{a})[D \to T]$$

where D and T are PLang formulas with T monotone. This is a special interest sentence with an empty outer configuration, where the source has just one laminar component, this having an empty prefix and just one disjunct, namely D. The algorithm proceeds by rephrasing D. As this happens the shape of the sentence will become more like that of a special interest sentence as described in the definition. At certain stages skolem variables will be introduced; these and other variables will be collected together to form the outer configuration.

Notice that the restriction 'special interest' is no restriction at all. Any sentence $(\forall \mathsf{P})[M])$ can be rephrased as

$$(\forall \mathsf{P})[D \to \bot]$$

where $D = \neg M$ and $\bot$ is a trivially false sentence.

At this stage it will be useful if you see some examples of special interest sentences. Later I will use the same examples to illustrate the working of the reduction phase of the algorithm.

We look at four examples, A, B, C, D, each of the shape

$$\Pi \ ::= \ (\forall P, Q)(\forall a)[\Theta \to \Psi] \tag{9}$$

where Θ, Ψ are built up from the predicate variables P, Q and certain binary relations

$$i(\cdot,\cdot)\ ,\ j(\cdot,\cdot)\ ,\ k(\cdot,\cdot)\ ,\ l(\cdot,\cdot)\ ,\ m(\cdot,\cdot)\ ,\ n(\cdot,\cdot)$$

which are part of the underlying signature. Thus there are just two predicate variables P, Q and the outer configuration is empty. In all these examples I will assume that Ψ is monotone, but we will not look at its internal structure. However, we will look at the internal structure of Θ.

7 RUNNING EXAMPLE (A). Let

$$\Theta \ ::= \ (\exists b \text{ with } l(a,b))\Theta'$$

where

$$\Theta' \ ::= \ (\forall x \text{ with } m(b,x))P(x) \ \wedge \ (\exists c \text{ with } n(b,c))Q(c).$$

Here (9) is a special interest sentence with an empty outer configuration and just one laminar component. This component has an empty prefix. Notice how the quantified inhabitant variables b, x, c are restricted by the configurations

$$l(a,b)\ ,\ m(b,x)\ ,\ n(b,c)$$

respectively. ■

8 RUNNING EXAMPLE (B). Let

$$\Theta \ ::= \ \begin{pmatrix} \forall x \text{ with} \\ i(a,x) \end{pmatrix} \begin{pmatrix} \exists u \text{ with} \\ j(x,u) \end{pmatrix} P(u) \ \wedge \ \begin{pmatrix} \forall y \text{ with} \\ k(a,y) \end{pmatrix} \begin{pmatrix} \exists v \text{ with} \\ l(y,v) \end{pmatrix} Q(v).$$

This version of (9) is a special interest sentence with two laminar components both of which have a non-empty quantifier prefix with an inner configuration, namely

$$\begin{pmatrix} \forall x \text{ with} \\ i(a,x) \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \forall y \text{ with} \\ k(a,y) \end{pmatrix}.$$

Inside these there is a restricted existential quantifier and the reduction phase will eventually have to handle this alternating quantifier prefix. ■

9 RUNNING EXAMPLE (C). Let

$$\Theta \ ::= \ (\exists b \text{ with } i(a,b))[(\Theta_1 \vee \Theta_2) \wedge \neg P(b)]$$

for some Θ_1, Θ_2. This special interest sentence (9) has just one laminar component, and this component has an empty prefix and just one disjunct. We will see that part of this disjunct, namely $\neg P(b)$, will eventually be transferred into the target. ■

10 RUNNING EXAMPLE (D). Let

$$\Theta \ ::= \ \left(\begin{array}{c} \exists b \text{ with} \\ i(a,b) \end{array} \right) \left(\begin{array}{c} \forall x \text{ with} \\ j(b,x) \end{array} \right) \Theta'$$

where

$$\Theta' \ ::= \ \Theta_1 \vee \left(\begin{array}{c} \forall y \text{ with} \\ m(x,y) \end{array} \right) \Theta_2$$

where

$$\Theta_1 \ ::= \ (\exists v)[k(x,v) \wedge \Xi_1] \quad , \quad \Theta_2 \ ::= \ (\exists w)[n(y,w) \wedge \Xi_2]$$

for some Ξ_1, Ξ_2. Here (9) is a special interest sentence with just one laminar component, and this component has just one disjunct. However, the internal structure of this disjunct is more complicated and will require several steps to reduce. The two clauses $k(x,v)$ and $n(y,w)$ (of Θ_1 and Θ_2) will combine to contribute to the eventual target and to certain inner configurations. ■

Before I describe the reduction phase and show how it applies to the examples, you might like to work out how you would rephrase these sentences.

5 Kernel extraction

The job of the reduction phase is to transform the sentence into a certain normal form from which either

- we can immediately read off the required kernel, or
- we can see that the reduction fails.

The second alternative can arise when the original sentence is too complicated for this particular approach to work. Remember also that there are some sentences (1) from which the predicate variables simply cannot be eliminated.

11 DEFINITION. (a) A special interest sentence Π (as in (7)) is in *literal normal form* if each disjunct of each of its laminar components is a literal of the form either

$$P(y) \quad \text{or} \quad \neg P(y) \tag{10}$$

for some predicate variable P (taken from the initial block P) and some inhabitant variable y (which will be quantified earlier in the sentence).

(b) A special interest sentence is in *kernel normal form* if each of its laminar components has the shape

$$(\forall \mathsf{x} \text{ with } \mathcal{X})[P(y)] \tag{11}$$

for some predicate variable P (taken from the initial block P) and some inhabitant variable y (which will be quantified earlier in the sentence). (In particular, each sentence in kernel normal form is also in literal normal form.)■

The reduction phase of the algorithm will automatically produce a literal normal form. The algorithm is *successful* when this is a kernel normal form, for then we may apply the following result.

12 THEOREM. *Each special interest sentence*

$$\Pi \ ::= \ (\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[S(\mathsf{P}) \rightarrow T(\mathsf{P})]$$

which is in kernel normal form has a kernel K*, and hence setting*

$$\Sigma \ ::= \ (\forall \mathsf{a} \text{ with } \mathcal{A})[T(\mathsf{K})]$$

produces an equivalent Σ *of* Π.

Proof. We consider each of the laminar components of Π in turn. Since Π is in kernel normal form this means that each such component has the shape (11) for some predicate variable P and inhabitant variable y (taken from x and the parameters a of Π). For this component we introduce a fresh inhabitant variable u and rephrase L first as

$$(\forall u, \mathsf{x} \text{ with } \mathcal{X})[y = u \rightarrow P(u)]$$

and then as

$$(\forall u)[Z(u) \rightarrow P(u)]$$

where

$$Z(u) \ ::= \ (\exists \mathsf{x})[\bigwedge \mathcal{X} \wedge (y = u)].$$

(This step is merely to ensure that certain inhabitant variables are the same.)

Consider now all the laminar components of the source $S(\mathsf{P})$ which use the one predicate variable P. Let this collection be

$$\mathcal{L}(P) \ = \ \{L_k \mid 1 \leq k \leq p\}$$

and set

$$S(P) \ ::= \ \bigwedge \mathcal{L}(P).$$

From above, each L_k is equivalent to a formula

$$(\forall u)[Z_k(u) \rightarrow P(u)]$$

so that $S(P)$ is equivalent to

$$(\forall u)[K_P(u) \rightarrow P(u)]$$

where

$$K_P(u) \ ::= \ \bigvee \{Z_k(u) \mid 1 \leq k \leq p\}.$$

Note that $K_P(u)$ is a predicate witness for P, and may contain other free variables taken from the parameters a.

If we now use the abbreviation introduced earlier, we see that we have rephrased $S(P)$ to make

$$(\forall \mathsf{P})(\forall \mathsf{a})[S(P) \ \leftrightarrow \ [K_P \leq P]]$$

universally valid.

Finally, let

$$\mathsf{K} \;=\; (K_P \mid P \in \mathsf{P})$$

be the sequence of predicate witnesses so constructed and, as before, let

$$[\mathsf{K} \leq \mathsf{P}] \quad \text{abbreviate} \quad \bigwedge\{[K_P \leq P] \mid P \in \mathsf{P}\}.$$

Then, since

$$S(\mathsf{P}) \;::=\; \bigwedge\{S(P) \mid P \in \mathsf{P}\}$$

we see that

$$(\forall \mathsf{P})(\forall \mathsf{a})[S(\mathsf{P}) \;\leftrightarrow\; [\mathsf{K} \leq \mathsf{P}]]$$

is universally valid. In particular, the sequence K is a kernel for $S(\mathsf{P})$, and the required result follows immediately. ■

Let us see how this result applies to two of the Running Examples

13 RUNNING EXAMPLE (A). We will see later that this sentence Π reduces to

$$\Pi'' \;::=\; (\forall P, Q)\begin{pmatrix} \forall a, b, c \text{ with} \\ l(a,b)\,,\; n(b,c) \end{pmatrix}[\Theta'' \to \Psi]$$

where

$$\Theta'' \;::=\; \begin{pmatrix} \forall x \text{ with} \\ m(b,x) \end{pmatrix}[P(x)] \;\wedge\; Q(c).$$

(The notation $(\,\cdot\,)''$ indicates that the reduction will require two steps.) This sentence is in kernel normal form and gives

$$K_P(u) \;::=\; m(b,u) \quad , \quad K_Q(v) \;::=\; (c = v)$$

as the predicate witnesses for P and Q. Now suppose that

$$\Psi \;::=\; P(a) \vee Q(a).$$

Then Π is equivalent to

$$\Sigma \;::=\; \begin{pmatrix} \forall a, b, c \text{ with} \\ l(a,b)\,,\; n(b,c) \end{pmatrix}[K_P(a) \vee K_Q(a)]$$

which expands to

$$(\forall a, b, c)[l(a,b) \wedge n(b,c) \;\to\; m(b,a) \vee (c = a)].$$

You should try writing out the original sentence Π for this particular Ψ, and comparing it with this obtained equivalent. ■

14 RUNNING EXAMPLE (D). We will find that, after five steps the sentence Π reduces to a shape

$$\Pi^{(5)} \;::=\; (\forall P, Q)(\forall \mathsf{a} \text{ with } \mathcal{A})[\Delta_1 \wedge \Delta_2 \wedge \Delta_3 \to \Phi]$$

where

$$\Delta_1 \ ::= \ (\forall \mathsf{x}_1 \text{ with } \mathcal{X}_1)\Xi_1$$

$$\Delta_2 \ ::= \ (\forall \mathsf{x}_2 \text{ with } \mathcal{X}_2)\Xi_2$$

$$\Delta_3 \ ::= \ (\forall \mathsf{x}_3 \text{ with } \mathcal{X}_3)[\Xi_1 \vee \Xi_2]$$

for appropriate configurations $\mathcal{A}, \mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ and some monotone Φ. Suppose now that each of Ξ_1, Ξ_2, Ξ_3 is an instance of $P(\cdot)$ or $Q(\cdot)$ at some variable. Then $\Pi^{(5)}$ is in literal normal form. However, because of the Δ_3 clause, $\Pi^{(5)}$ is in kernel normal form *only if* Ξ_1 *and* Ξ_2 *are identical*, i.e. instances of the same predicate variable at the same inhabitant variable. This illustrates how the shape of the kernel normal form is a much more severe restriction than that of the literal normal form. ■

At this juncture there appears to be no alternative but to get stuck into a description of the reduction phase of the algorithm. Not quite; let's take a look at a result in circumscription.

6 An example in circumscription

The topic of circumscription arises when we are interested in certain structures for which part of the data is a distinguished predicate constant C. These structures are required to model certain global restrictions given by a sentence $T(C)$, and amongst these models we wish to isolate those for which the predicate C is minimal. In general this minimality is a higher order requirement, but in particular cases it may be captured in a lower order way.

Suppose we add to the underlying language Lang a predicate constant C to obtain the language $\mathsf{Lang}(C)$. Every formula $\phi(C)$ of $\mathsf{Lang}(C)$ can then be regarded as the instantiation of a formula $\phi(P)$ of PLang obtained by replacing the predicate variable P by C. In particular, the global restrictions $T(C)$ can be viewed as the instantiation of some formula $T(P)$ of PLang.

The required minimality can be phrased informally as

> any predicate P which satisfies $T(P)$ and which is included in C must, in fact, be equal to C.

Formally this is the sentence

$$T^{\text{circ}} \ ::= \ (\forall P)\left[\begin{pmatrix} T(P) \\ \wedge \\ (\forall u)[P(u) \rightarrow C(u)] \end{pmatrix} \rightarrow (\forall a)[C(a) \rightarrow P(a)]\right]$$

in the language $\mathsf{PLang}(C)$. We are interested in finding a P-free equivalent of T^{circ}.

A result given by Lifschitz in [10, 11] is that such an equivalent does exist when $T(P)$ is monotone (in P). Van Benthem pointed out to me that this result follows by a simple application of the substitution method, and provides a nice illustration of Theorem 12 without the need to go into the details of the reduction process.

We need a bit of notation. For the formula $T(P)$ we look at two substitution instances

$$T[\bullet \text{ for } P]$$

obtained by replacing the variable P throughout T by $\bullet$ (whatever this is). The two instances are

$$\begin{array}{lll} T^{\neg}(P) & \text{given by} & T[\neg P \text{ for } P] \\ T^{*}(C) & \text{given by} & T[C(\cdot) \wedge (\cdot \neq a) \text{ for } P]. \end{array}$$

Note that $T^{*}(C)$ has a as a free inhabitant variable. With this we can state and prove Lifschitz's result.

15 THEOREM. *When the formula $T(P)$ is monotone, the minimality requirement T^{circ} is equivalent to $(\forall a)[C(a) \rightarrow \neg T^{*}(C)]$.*

Proof. Some simple rearrangements inside the matrix shows that T^{circ} is equivalent to

$$(\forall P)\left[\left(\begin{array}{l} (\forall u)[\neg C(u) \rightarrow \neg P(u)] \\ \wedge \\ (\exists a)[C(a) \wedge \neg P(a)] \end{array}\right) \rightarrow \neg T(P)\right]$$

and by replacing P by $\neg P$ this becomes

$$(\forall P)\left[\left(\begin{array}{l} (\forall u)[\neg C(u) \rightarrow P(u)] \\ \wedge \\ (\exists a)[C(a) \wedge P(a)] \end{array}\right) \rightarrow \neg T^{\neg}(P)\right].$$

After pulling out the quantifier $(\exists a)$ this can be rephrased as

$$(\forall P)(\forall a)\left[\left(\begin{array}{l} (\forall u)[\neg C(u) \rightarrow P(u)] \\ \wedge \\ P(a) \end{array}\right) \rightarrow \neg C(a) \vee \neg T^{\neg}(P)\right]$$

and this is in kernel normal form.

The source of this sentence can be rephrased as

$$(\forall u)[\neg C(u) \vee (u = a) \ \rightarrow \ P(u)]$$

so that T^{circ} is equivalent to

$$(\forall P)(\forall a)[[K \leq P] \ \rightarrow \ \neg C(a) \vee \neg T^{\neg}(P)]$$

where

$$K(u) \ ::= \ \neg C(u) \vee (u = a).$$

Next observe that since $T(P)$ is monotone, the formula $T^{\neg}(P)$ is antitone, and hence $\neg T^{\neg}(P)$ is monotone. The result of replacing $P(\cdot)$ in $T^{\neg}(P)$ by $K(\cdot)$ is just the formula $T^{*}(C)$, and hence an application of Theorem 12 produces the equivalent of T^{circ} required to complete the proof. ■

The syntactic rearrangements performed in this proof are all simple examples of possible steps in the reduction phase.

7 Reduction phase

For technical reasons (which will become clear later) we don't reduce a single special interest sentence but a finite collection of such sentences.

16 DEFINITION. (a) A *general interest sentence* is a finite conjunction

$$\Gamma \ ::= \ \Pi_1 \wedge \cdots \wedge \Pi_n$$

of special interest sentences $\Pi_1, \ldots, \Pi_n$.

(b) A general interest sentence Γ is in

literal , kernel

normal form if each of its special conjuncts is in the corresponding normal form. ■

I will describe a reduction step

$$\Gamma \mapsto \Gamma'$$

from a general interest sentence Γ to a general interest sentence Γ' where the equivalence

$$\Gamma \ \Leftrightarrow \ \Gamma'$$

is universally valid (for fairly trivial reasons). When iterated

$$\Gamma \mapsto \Gamma' \mapsto \Gamma'' \mapsto \cdots$$

this reduction will eventually terminate in a certain normal form equivalent Γ^* of Γ. This reduction process has three aims:

- To absorb as much as possible into the various targets (which, of course, must remain monotone);
- To eliminate many of the quantifiers from the various disjuncts;
- To isolate the occurrences of the various predicate variables.

This will be achieved by a refined version of the usual procedure for converting a sentence to skolem normal form; but skolem functions will be introduced only when absolutely necessary. The shape of a special interest sentence has been designed to help organize this reduction. In particular, the various inner configurations will be used to gather together the predicate free parts of the sentence, and the outer configuration will house the restrictions on the introduced skolem variables.

To operate the reduction step

$$\Gamma \mapsto \Gamma'$$

choose a special conjunct Π of Γ, choose a laminar component L of Π, and attack L to produce a suitable replacement for Π to obtain Γ'. This step is repeated until a literal normal form is achieved (when, in fact, no more reduction steps are possible).

Let us look at the details of the attack on L.

We know that Π has the shape

$$\Pi \ ::= \ (\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[L \wedge S^+ \to T]$$

where S^+ is the conjunction of the remaining laminar components not under immediate attack. We know also that the shape of L is as in (8). We now perform the following manipulations where, in cases of conflicting possibilities, priority is given to the earliest mentioned choice.

As you go through these manipulations you might like to keep in front of you the Running Examples. You will find it instructive to test the applicability of each manipulation on each Example. Descriptions of the resulting reductions are given at the end of this section.

Monotone collection

If L is antitone, i.e. has the form

$$L \ \equiv \ \neg M$$

for some monotone M, then we transfer M from the source to the target by setting

$$\Pi' \ ::= \ (\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[S^+ \to M \vee T]$$

to get $\Pi \Leftrightarrow \Pi'$.

Empty prefix-$\bigvee$

Suppose the initial prefix of L is empty, so that

$$L \ ::= \ \bigvee\{L_\ell \mid 1 \leq \ell \leq n\}$$

where $\mathcal{D} \ = \ \{L_\ell \mid 1 \leq \ell \leq n\}$. For each $1 \leq \ell \leq n$ set

$$\Pi_\ell \ ::= \ (\forall \mathsf{P})(\forall \mathsf{a} \text{ with } \mathcal{A})[L_\ell \wedge S^+ \to T]$$

and observe that

$$\Pi \ \Leftrightarrow \ \bigwedge\{\Pi_\ell \mid 1 \leq \ell \leq n\}.$$

When $n > 1$, this step increases the number of special conjuncts of Γ but decreases the overall complexity. When $n = 1$, this step is vacuous. (This step is the reason for working with general interest sentences.)

Empty prefix-$\exists$

Suppose that the initial prefix of L is empty, that L has just one disjunct, and that this disjunct begins with an existential quantifier. In these circumstances we may rephrase L as

$$L \ ::= \ (\exists \mathsf{y} \text{ with } \mathcal{Y}[L']$$

where $\mathcal{Y}$ is an inner configuration in the fresh variables y (and earlier variables) and L' is an appropriate formula. This, of course, is a suggestive way of writing

$$(\exists \mathsf{y})[\bigwedge \mathcal{Y} \wedge L'].$$

We may assume that no variable from y appears free in S^+ or T, so that $\Pi \Leftrightarrow \Pi'$ where

$$\Pi' ::= (\forall \mathsf{a}, \mathsf{y} \text{ with } \mathcal{A}, \mathcal{Y})[L' \wedge S^+ \to T].$$

This increases the number of parameters and increases the size of the outer configuration but does not introduce any new skolem constructs.

When none of these steps is applicable we attack one of the non-literal disjuncts D of $\mathcal{D}$. Thus we have

$$L ::= (\forall \mathsf{x} \text{ with } \mathcal{X})[D \vee F]$$

where F is the disjunction of the remaining disjuncts not under attack. We assume that D is written in *standard* form, that is, with no occurrences of $\to$ and $\leftrightarrow$, and with $\neg$ occurring only immediately in front of an atomic formula. (Thus, to rephrase a formula in standard form, first eliminate $\to$ and $\leftrightarrow$ in favour of $\vee$ and $\neg$, and then drive all the $\neg$ down onto the atoms eliminating $\neg\neg$ wherever possible.) Having achieved this standardization we use whichever of the following manipulations is possible.

Predicate free

If D is predicate free (i.e. is a SLang formula) then we rephrase L as

$$(\forall \mathsf{x} \text{ with } \mathcal{X}, \neg D)[F]$$

so absorbing D into the inner configuration where it is immune from attack. (In the case where D is the only disjunct of this component we set $F = \bot$.)

Disjunction transfer

If D is itself a disjunction then we simply relabel the disjuncts of L to deal with these new disjuncts separately.

Conjunction transfer

If D is a conjunction,

$$D ::= \bigwedge\{D_\ell \mid 1 \leq \ell \leq n\}$$

then we set

$$L_\ell ::= (\forall \mathsf{x} \text{ with } \mathcal{X})[D_\ell \vee F]$$

(for each $1 \leq \ell \leq n$) so that

$$L \Leftrightarrow \bigwedge\{L_\ell \mid 1 \leq \ell \leq n\}$$

and then we may relabel the laminar components of Π.

$\forall$ transfer

Suppose that

$$D ::= (\forall \mathsf{y} \text{ with } \mathcal{Y})[E]$$

for some sequence y of fresh inhabitant variables and inner configuration $\mathcal{Y}$ in these and earlier variables. We can arrange that no variable from y is free in F. Then we set

$$L' ::= (\forall \mathsf{x}, \mathsf{y} \text{ with } \mathcal{X}, \mathcal{Y})[E \vee F]$$

to get

$$L \ \Leftrightarrow \ L'.$$

This increases the initial prefix of this laminar component but simplifies its matrix.

Observe that the manipulations so far have not introduced any skolem variables. In particular, if Γ is a PLang sentence (rather than the potentially more complicated PSLang sentence), then the equivalent Γ' is also a PLang sentence. Skolem variables are required only for the remaining case.

This final case is, of course, the case where D begins with an existential quantifier. This splits into two subcases.

Outer extension-without

Suppose that F is empty, i.e. D is without a companion. Then we rephrase L in the form

$$L \ ::= \ (\forall \mathsf{x} \text{ with } \mathcal{X})(\exists y \text{ with } \mathcal{Z})[E]$$

for some configuration $\mathcal{Z}$ in the new variable y and the earlier variables (x and a), and for some formula E.

We introduce a fresh (skolem) variable of the correct arity with defining restrictions $\mathcal{F}$ given by

$$f \in \mathcal{F} \ ::= \ \left(\begin{array}{c} \forall \mathsf{x}, y \text{ with} \\ \mathcal{X}, f(\mathsf{x}) = y \end{array} \right) [\textstyle\bigwedge \mathcal{Z}]$$

which by setting

$$\mathsf{y} \ ::= \ \mathsf{x}, y \quad , \quad \mathcal{Y} \ ::= \ \mathcal{X}, f(\mathsf{x}) = y.$$

we can abbreviate to

$$f \in \mathcal{F} \ ::= \ (\forall \mathsf{y} \text{ with } \mathcal{Y}[\textstyle\bigwedge \mathcal{Z}].$$

A use of the Axiom of Choice now gives

$$L \ \Leftrightarrow \ (\exists f \in \mathcal{F})L'$$

where

$$L' \ ::= \ (\forall \mathsf{y} \text{ with } \mathcal{Y})[E]$$

and a routine argument shows that

$$\Pi \ \Leftrightarrow \ \Pi'$$

where

$$\Pi' \ ::= \ (\forall \mathsf{P}) \left(\begin{array}{c} \forall \mathsf{a}, f \text{ with} \\ \mathcal{A}, f \in \mathcal{F} \end{array} \right) [L' \wedge S^+ \to T].$$

Notice how this step extends the outer configuration.

Outer extension-with

If F is not empty, i.e. D occurs with a companion, then we have

$$L \ ::= \ (\forall \mathsf{x} \text{ with } \mathcal{X})[(\exists y)E \vee F]$$

for some formula E. By a suitable renaming we may assume that the variable y is not free in F. We introduce a fresh (skolem) variable f and set

$$\mathsf{y} ::= \mathsf{x}, y \quad , \quad \mathcal{Y} ::= \mathcal{X}, f(\mathsf{x}) = y.$$

$$L' ::= (\forall \mathsf{y} \text{ with } \mathcal{Y})[E \vee F]$$

$$\Pi' ::= (\forall \mathsf{P}) \begin{pmatrix} \forall \mathsf{a}, f \text{ with} \\ \mathcal{A} \end{pmatrix} [L' \wedge S^+ \to T]$$

and then a simple calculation shows that

$$\Pi \Leftrightarrow \Pi'.$$

Notice that this step introduces a new parameter, but does not extend the outer configuration.

This completes the description of the possible reduction steps.

At a first reading of these steps you may think that the Empty prefix-$\exists$ case could be subsumed under the Outer extension cases. So it could, but there are good reasons for not doing this. The Empty prefix-$\exists$ case does *not* introduce any skolem variables, whereas the Outer extension reductions do, and we wish to avoid unnecessary skolem variables whenever possible.

This is also the reason for putting a priority on the possible manipulations in cases of conflicting choices. This priority preserves an empty inner prefix for as long as possible, and so delays the introduction of skolem variables (by having to apply an Outer extension case) until absolutely necessary.

Where an Outer extension rephrasing has to be applied, it seems that it is more efficient to use the without case wherever possible. There are various tricks for converting a simple case into a complex case and some of these are used later in Examples 35 and 36.

As you worked through these reduction steps you should have been trying them out on the Running Examples. Let's see how you got on.

17 RUNNING EXAMPLE (A). This is a special interest sentence with an empty outer configuration and just one laminar component. This component has an empty prefix so an application of Empty prefix-$\exists$ is required. Such an application produces

$$\Pi' ::= (\forall P, Q)(\forall a, b \text{ with } l(a, b))[\Theta' \to \Psi].$$

This is another special interest sentence this time with two laminar components. The first of these components is immune from attack, but another application of Empty prefix-$\exists$ on the second component gives the sentence Π'' as was considered in item 13. ■

18 RUNNING EXAMPLE (B). This is a special interest sentence with two laminar components both of which are susceptible to applications of Outer extension-without. We thus introduce new skolem variables f, g of the appropriate type together the following restrictions

$$f \in F(a) \Leftrightarrow \begin{pmatrix} \forall x, u \text{ with} \\ i(a, x)\,,\ f(x) = u \end{pmatrix} [j(x, u)]$$

$$g \in G(a) \Leftrightarrow \begin{pmatrix} \forall y, v \text{ with} \\ k(a, y)\,,\ g(y) = v \end{pmatrix} [l(y, v)]$$

on these variables. This produces

$$\Pi' \ ::= \ (\forall P, Q)\left(\begin{array}{c}\forall a, f, g \text{ with}\\ f \in F(a)\,,\, g \in G(a)\end{array}\right)[\Theta' \to \Psi]$$

where

$$\Theta' \ ::= \ \left(\begin{array}{c}\forall u, x \text{ with}\\ i(a,x)\,,\, f(x) = u\end{array}\right)[P(u)] \ \wedge \ \left(\begin{array}{c}\forall v, y \text{ with}\\ k(a,y)\,,\, g(y) = v\end{array}\right)[Q(v)].$$

This sentence Π' is in kernel normal form and gives

$$K_P(u) \ ::= \ (\exists x)[i(a,x) \wedge (f(x) = u)] \quad , \quad K_Q(v) \ ::= \ (\exists y)[k(a,y) \wedge (g(y) = v)]$$

as the predicate witnesses for P and Q. ■

19 RUNNING EXAMPLE (C). This special interest sentence Π has just one laminar component, and this component has an empty prefix and just one disjunct. This disjunct begins with $\exists$ so we apply Empty prefix-$\exists$ to get

$$\Pi' \ ::= \ (\forall P, Q)(\forall a, b \text{ with } i(a,b))[(\Theta_1) \vee \Theta_2) \wedge \neg P(b) \ \to \ \Psi].$$

This special interest sentence has two laminar components one of which, $\neg P(b)$, is antitone. We thus apply Monotone collection to get

$$\Pi'' \ ::= \ (\forall P, Q)(\forall a, b \text{ with } i(a,b))[\Theta_1) \vee \Theta_2 \ \to \ \Psi'']$$

where

$$\Psi'' \ = \ \Psi \vee P(b).$$

This produces a special interest sentence still with just one laminar component, and where this component has an empty prefix. However, this component has two disjuncts, Θ_1, Θ_2, so we apply Empty prefix-$\bigvee$ to get

$$\Pi \ \Leftrightarrow \ \Pi_1 \wedge \Pi_2$$

where

$$\Pi_r \ ::= \ (\forall P, Q)(\forall a, b \text{ with } i(a,b))[\Theta_r(b) \to \Psi'']$$

for $r = 1, 2$. We could now attack Π_1 and Π_2 separately, but there is no need to pursue this here. ■

20 RUNNING EXAMPLE (D). Here we have a special interest sentence with just one laminar component, where this component has just one disjunct, so an application of Empty prefix-$\exists$ produces

$$\Pi' \ ::= \ (\forall P, Q)\left(\begin{array}{c}\forall a, b \text{ with}\\ i(a,b)\end{array}\right)\left[\left(\begin{array}{c}\forall x \text{ with}\\ j(b,x)\end{array}\right)\Theta' \to \Psi\right]$$

where Θ' is as above. This special interest sentence also has just one laminar component (which has a non-empty prefix), and this component has two disjuncts. One of these disjuncts (namely the second one) is susceptible to $\forall$ transfer, and this produces

$$\Pi'' \ ::= \ (\forall P, Q)\left(\begin{array}{c}\forall a, b \text{ with} \\ i(a,b)\end{array}\right)\left[\left(\begin{array}{c}\forall x, y \text{ with} \\ j(b,x)\,,\ m(x,y)\end{array}\right)[\Theta_1 \vee \Theta_2] \to \Psi\right].$$

This sentence still has just one laminar component but now has two disjuncts. Both of these begin with existential quantifiers, so we are forced into two applications of Outer extension-with). For this we introduce two skolem variables f, g (of the appropriate type) and reduce Π'' to

$$\Pi''' \ ::= \ (\forall P, Q)(\forall a, b, f, g \text{ with } i(a,b))[\Theta''' \to \Psi]$$

where

$$\Theta''' \ ::= \ \left(\begin{array}{c}\forall x, y, v, w \text{ with} \\ \\ j(b,x)\,,\ m(x,y) \\ \\ f(x,y) = v\,,\ g(x,y) = w\end{array}\right)\left[\begin{array}{l}k(x,v) \wedge \Xi_1 \\ \vee \\ n(y,w) \wedge \Xi_2\end{array}\right].$$

This special interest sentence still has one laminar component, but now with two disjuncts, where these disjuncts are both conjunctions. Several applications of Conjunction transfer produces

$$\Pi^{(4)} \ ::= \ (\forall P, Q)(\forall a, b, f, g \text{ with } i(a,b))[\Lambda_0 \wedge \Lambda_1 \wedge \Lambda_2 \wedge \Lambda_3 \to \Psi]$$

where, for $0 \le r \le 3$,

$$\Lambda_r \ ::= \ \left(\begin{array}{c}\forall x, y, v, w \text{ with} \\ \\ j(b,x)\,,\ m(x,y) \\ \\ f(x,y) = v\,,\ g(x,y) = w\end{array}\right)\Gamma_r \quad \text{where} \quad \begin{array}{lcl}\Gamma_0 & ::= & k(x,v) \vee n(y,w) \\ \Gamma_1 & ::= & n(y,w) \vee \Xi_1 \\ \Gamma_2 & ::= & k(x,v) \vee \Xi_2 \\ \Gamma_3 & ::= & \Xi_1 \vee \Xi_2.\end{array}$$

This is a special interest sentence with four laminar components, namely, $\Lambda_0, \Lambda_1, \Lambda_2, \Lambda_3$. The first of these contains no occurrence of P nor Q and so is antitone. It can, therefore, be transferred across $\to$ to produce a new target

$$\Psi' \ ::= \ (\exists x, y, v, w)\left[\begin{array}{l}j(b,x) \wedge m(x,y) \\ \wedge \\ f(x,y) = v \wedge g(x,y) = w \\ \wedge \\ \neg k(x,v) \wedge \neg n(y,w)\end{array}\right] \wedge \Psi.$$

The middle two components, Λ_1 and Λ_2, both contain a predicate free disjuncts, so can be rephrased as

$$\Lambda_1' \ ::= \ \left(\begin{array}{c}\forall x, y, v, w \text{ with} \\ \\ j(b,x)\,,\ m(x,y) \\ \\ f(x,y) = v\,,\ g(x,y) = w \\ \\ \neg n(y,w)\end{array}\right)\Xi_1 \quad , \quad \Lambda_2' \ ::= \ \left(\begin{array}{c}\forall x, y, v, w \text{ with} \\ \\ j(b,x)\,,\ m(x,y) \\ \\ f(x,y) = v\,,\ g(x,y) = w \\ \\ \neg k(x,v)\end{array}\right)\Xi_1$$

to produce

$$\Pi^{(5)} \ ::= \ (\forall P, Q)(\forall a, b, f, g \text{ with } i(a,b))[\Lambda'_1 \wedge \Lambda'_2 \wedge \Lambda_3 \to \Psi'].$$

This is the shape that was considered in item 14. ■

A further selection of examples is given in Section 11.

8 Skolem free equivalents

Suppose we start from a general interest sentence Γ which is a PLang sentence. Under what conditions will the algorithm produce a Lang equivalent for Γ? Clearly we require that

- the reduction phase does not introduce any skolem variables, and
- the achieved literal normal form is a kernel normal form.

These can be guaranteed by appropriate extra conditions on the sentence Γ, and I have designed the algorithm so that these extra restrictions are almost immediate. Before I give them it is convenient to introduce an auxiliary notion.

21 DEFINITION. A PLang formula is *molecular* if it is either

- antitone

or of one of the two shapes

- $(\forall \mathsf{x} \text{ with } \mathcal{X})[P(y)]$
- $(\exists \mathsf{y} \text{ with } \mathcal{Y})[\bigwedge \mathcal{M}]$

where x, y are sequences of inhabitant variables; $\mathcal{X}, \mathcal{Y}$ are configurations (in earlier occurring variables); P is a predicate variable and y is a matching inhabitant variable; and $\mathcal{M}$ is a finite collection of (previously constructed) molecular formulas.) Thus molecular formulas are constructed recursively from certain basic formulas by repeated use of this last clause.) ■

The original modal case of the algorithm was analysed by Sahlqvist in [2]. As we will see later, the following definition is obtained by lifting Sahlqvist's conditions to the present more general situation.

22 DEFINITION. (a) A *special S-sentence* is a special interest sentence which is a PLang sentence (skolem free) and for which each laminar component is molecular.

(b) A *general S-sentence* is a finite conjunction of special S-sentences. ■

The analogue of Sahlqvist's result is now straight forward.

23 THEOREM. *Each general S-sentence is equivalent to a* Lang *sentence.*

Proof. Consider first the effect of the reduction phase. At each step we attack one of the laminar components of one of the special conjuncts. This component is molecular.

- If this component is antitone then it is transferred into the target formula. This step produces a new general S-sentence.
- If the component is
$$(\forall \mathsf{x} \text{ with } \mathcal{X})[P(u)]$$
then this component is not attacked.
- If the component is
$$(\exists \mathsf{y} \text{ with } \mathcal{Y})[\bigwedge \mathcal{M}]$$
then we set $L' = \bigwedge \mathcal{M}$ and deal with this by the Empty prefix-$\exists$ case. This extends the outer configuration of the special conjunct, but again produces a new general S-sentence.

This shows that the whole reduction phase never leaves the realm of general S-sentences, and so the achieved equivalent normal form is a general S-sentence.

Consider now the shape of a general S-sentence which is in literal normal form. Since no other reductions are possible, each laminar component of each special conjunct is of the form
$$(\forall \mathsf{x} \text{ with } \mathcal{X})[P(y)]$$
and hence the sentence is in kernel normal form. Thus Theorem 12 is applicable. ■

This completes the generalities. It is now time to look at some particular examples taken from the area of modal logic.

PART B: applications to modal logic

9 Modal background

Modal logic is concerned with applying propositional techniques to analyse the properties of relational structures. This is a viable technique since the propositional language used has some second order power. In particular, the canonical semantics of a modal formula is expressible as a second order (in fact, Π^1_1) sentence. Correspondence theory is that part of modal logic which is concerned with delineating the precise expressive properties of modal languages.

I have already described a general algorithm which, when supplied with a higher order sentence involving universally quantified predicate variables, will search for a predicate free equivalent. In the next section we are going to use this algorithm on the semantics of modal formulas. Before this I will give a quick survey of the relevant background.

24 DEFINITION. Let I be a fixed set.

(a) A *transition structure* of signature I is a relational structure

$$\mathfrak{A} = (A, \mathsf{A})$$

where A is a non-empty set furnished with a family A of transition relations

$$\xrightarrow{i}$$

indexed by I. These transition relations are binary relations on A.

(b) The *labels* for a signature I are generated recursively by the clauses:

- $\emptyset$ is a label;
- each $i \in I$ is a label;
- if i and j are labels then so is ij.

The *atomic labels* are the members of I.

(c) Given a transition structure $\mathfrak{A}$ of signature I, each label l has an associated transition relation constructed out of the distinguished transition relations in A. For each $a, b \in A$,

$$a \xrightarrow{l} b$$

holds if and only if one of:

- $l = \emptyset$ and $a = b$;
- $l = i \in I$ and $a \xrightarrow{i} b$;
- $l = ij$ and $a \xrightarrow{i} x \xrightarrow{j} b$ for some $x \in A$

holds. ■

For each signature I there is a canonically associated first order language

$$\mathsf{Lang}$$

which is used to describe the properties of transition structures. Thus Lang has

- a stock of element variables intended to range over the elements of the carrying set A of the interpreting transition structure $\mathfrak{A}$,
- a distinguished binary relation symbol for each atomic label $i \in I$.

As usual we will confuse parts of the language Lang with the corresponding parts of the interpreting structure. In particular we write

$$\xrightarrow{i}$$

both for the distinguished binary relation symbol of Lang and for the corresponding transition relation of the interpreting structure. Notice that each atomic formula of Lang has the form

$$u = v \quad \text{or} \quad u \xrightarrow{i} v$$

where u, v are element variables and $i \in I$.

We enrich Lang to a language

$$\mathsf{PLang}$$

by adding

- a stock of predicate variables $P_1, P_2, P_3, \ldots$ intended to range over the subsets of the carrying set A of the interpreting structure $\mathfrak{A}$.

Thus in PLang we have new atomic formulas of the form

$$P(u)$$

where P is a predicate variable and u is an element variable. (The intended interpretation of this is that 'u belongs to P'.) We combine these new atomic formulas with the formulas of Lang to generate a new class of formulas (by closing off under the boolean connectives and element quantifiers). Let

$$\Phi(P_1, \ldots, P_m)$$

be one such formula, where I have displayed the occurring predicate variables $P_1, \ldots, P_m$. By universally quantifying these predicate variables we obtain a new kind of formula

$$\Pi \ ::= \ (\forall P_1, \ldots, P_m)\Phi(P_1, \ldots, P_m).$$

These are the Π^1_1 formulas.

Suppose that Π is a sentence, i.e. has no free element variables. We write

$$\mathfrak{A} \models \Pi$$

to indicate that Π is true in $\mathfrak{A}$. This, of course means that "For all subsets $X_1, \ldots, X_m$ of A, ...". We are interested in finding predicate free equivalents to such sentences Π.

So far I have introduced nothing that is peculiar to modal logic. Let us now look at these aspects.

25 DEFINITION. (a) The *modal formulas* (of signature I) are built up in the usual way from a stock of modal variables

$$p, q, \ldots$$

using the propositional connectives

$$\top, \bot, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$$

and the modal operators

$$[i], \langle i \rangle$$

(one for each label i).

(b) A *literal* is a modal formula p or $\neg p$ where p is a modal variable. A modal formula is *standard* if it is built up from literals without the use of $\neg, \rightarrow, \leftrightarrow$. A modal formula is *positive* if it is built up from variables without the use of $\neg, \rightarrow, \leftrightarrow$, that is, if it is standard and contains no occurrences of $\neg$. ■

The relation

$$\mathfrak{A} \Vdash \phi \tag{12}$$

between transition structures $\mathfrak{A}$ and modal formulas ϕ is defined in the usual way (in the manner of Kripke). Thus we use the intermediary notion of a *valuation* α on A (the carrier of $\mathfrak{A}$). This assigns to each variable p a subset $\alpha(p)$ of A comprising the elements of A on which p is assumed true. Let $\mathcal{V}A$ be the set of all such valuations. For $\alpha \in \mathcal{V}A$ and $a \in A$ the relation

$$(\mathfrak{A}, \alpha, a) \Vdash \phi$$

is defined by recursion on ϕ (with variation of the parameter a). The characteristic modal clauses are

$$(\mathfrak{A}, \alpha, a) \Vdash [i]\phi \Leftrightarrow \text{(For all } b \in A \text{ with } a \xrightarrow{i} b)[(\mathfrak{A}, \alpha, b) \Vdash \phi]$$

and

$$(\mathfrak{A}, \alpha, a) \Vdash \langle i \rangle\phi \Leftrightarrow \text{(There is some } b \in A \text{ with } a \xrightarrow{i} b)[(\mathfrak{A}, \alpha, b) \Vdash \phi].$$

We then define (12) by

$$\mathcal{A} \Vdash \phi \Leftrightarrow \text{(For all } \alpha \in \mathcal{V}A)\text{(For all } a \in A)[(\mathcal{A}, \alpha, a) \Vdash \phi].$$

By construction, this semantics extends the usual $\mathsf{2}$-valued semantics for propositional formulas. We also find that for each modal formula ϕ, the equivalences

$$[\emptyset]\phi \leftrightarrow \phi \qquad , \qquad \langle\emptyset\rangle\phi \leftrightarrow \phi$$

$$[ij]\phi \leftrightarrow [i]\,[j]\phi \quad , \quad \langle ij \rangle\phi \leftrightarrow \langle i \rangle\langle j \rangle\phi$$

and

$$\neg\,[i]\phi \leftrightarrow \langle i \rangle\neg\phi$$

hold for all transition structures.

Notice that these ensure that every modal formula can be rephrased in an equivalent form as a standard formula. This is done by first eliminating $\rightarrow$ and $\leftrightarrow$ in favour of

$\neg, \wedge, \vee$ and then driving all negation symbols $\neg$ down onto the atomic components (the variables) and cancelling $\neg\neg$ whenever this iteration occurs. Because of this we will not distinguish between a modal formula and any of its standard rephrasings.

It should be clear that the Kripke semantics of a modal formula can be described (via the Tarski semantics) by a Π^1_1 sentence $\Pi(\phi)$ such that

$$\mathfrak{A} \Vdash \phi \ \Leftrightarrow\ \mathfrak{A} \models \Pi(\phi)$$

holds for all transition structures $\mathfrak{A}$. We need to look at the details of the construction of $\Pi(\phi)$.

Fix an element variable u. Also fix an injective assignment

$$p \mapsto P$$

from modal variables p to predicate variables P. For each modal formula ϕ we construct a PLang formula $\ulcorner\phi\urcorner$ with the following properties.

- The only predicate variables occurring in $\ulcorner\phi\urcorner$ are those of the form P for modal variables p occurring in ϕ, and non of these are bound.
- The only other free variable of $\ulcorner\phi\urcorner$ is the distinguished element variable u. There may be bound element variables in $\ulcorner\phi\urcorner$.

For any modal formula ϕ and variable v we write

$$\ulcorner\phi\urcorner(v) \quad \text{for} \quad \begin{array}{l}\text{the result of substituting } v\\ \text{for } u \text{ throughout } \ulcorner\phi\urcorner.\end{array}$$

(In particular, $\ulcorner\phi\urcorner(u)$ is just $\ulcorner\phi\urcorner$.) The crucial clauses in the definition of $\ulcorner\cdot\urcorner$ are as follows.

- $\ulcorner p\urcorner \ ::= \ P(u)$
- $\ulcorner [i]\phi\urcorner \ ::= \ \begin{pmatrix} \forall v \text{ with} \\ u \xrightarrow{i} v \end{pmatrix} [\ulcorner\phi\urcorner(v)]$
- $\ulcorner \langle i\rangle\phi\urcorner \ ::= \ \begin{pmatrix} \exists v \text{ with} \\ u \xrightarrow{i} v \end{pmatrix} [\ulcorner\phi\urcorner(v)]$

Here I have condensed the quantificational phrases to make them more readable. The remaining definitional clauses simply say that $\ulcorner\cdot\urcorner$ preserves the propositional structure.

Note that the substitution process can involve changing some bound variables to avoid variable capture. For instance we have

$$\ulcorner p\urcorner \ ::= \ P(u) \qquad \text{and} \qquad \ulcorner p\urcorner(v) \ ::= \ P(v)$$

$$\ulcorner\langle j\rangle p\urcorner \ ::= \ \begin{pmatrix} \exists v \text{ with} \\ u \xrightarrow{j} v \end{pmatrix} [P(v)] \quad \text{and} \quad \ulcorner\langle j\rangle p\urcorner(v) \ ::= \ \begin{pmatrix} \exists w \text{ with} \\ v \xrightarrow{j} w \end{pmatrix} [P(w)]$$

so that

$$\ulcorner [i]\langle j\rangle p\urcorner \ ::= \ (\begin{pmatrix} \forall v \text{ with} \\ u \xrightarrow{i} v \end{pmatrix} \begin{pmatrix} \exists w \text{ with} \\ v \xrightarrow{j} w \end{pmatrix} [P(w)].$$

Whenever this translation is employed, I will judiciously choose the variables involved so that unwanted clashes do not happen. (This is not the most efficient translation in terms of the use of element variables, however, it is the safest.)

Let ϕ be any modal formula built up from the modal variables $p_1, \ldots, p_m$. The corresponding PLang formula $\ulcorner\phi\urcorner$ has free predicate variables $P_1, \ldots, P_m$ and a free element variable u (and no other free variables). We set

$$\Pi(\phi) \ ::= \ (\forall P_1, \ldots, P_m)(\forall a)[\ulcorner\phi\urcorner(a)]$$

to obtain the Π^1_1 translation of ϕ. We are interested in taking such a sentence $\Pi(\phi)$ and attempting to eliminate the predicate variables $P_1, \ldots, P_m$. This may require the introduction of certain skolem functions. Thus we enrich Lang to a language

$$\mathsf{SLang}$$

by adding

- for each $n \in \mathsf{N}$ with $n > 0$, a stack of *variables of arity* n intended to range over the n-ary operations
$$A^n \longrightarrow A$$
on the carrying set A of the interpreting structure.

The variables of Lang which already exists, the element variables, are the variables of arity 0. I will refer to this whole family of variables (the variables of arity n for $n \in \mathsf{N}$) as the *inhabitant variables*. This distinguishes them from the predicate variables (which, of course, do not occur in SLang).

The language SLang has various term forming constructs. Thus if f is an inhabitant variable of arity n and $t_1, \ldots, t_n$ are terms (already constructed) then

$$f(t_1, \ldots, t_n)$$

is a new term. The intended meaning of this is the result of evaluating the operation f at the arguments $t_1, \ldots, t_n$.

Finally we amalgamate Plang with SLang into one language

$$\mathsf{PSLang}.$$

This is done by adding to SLang the predicate variables (ranging over the subsets of the carrying set A, *not* over subsets of higher types). The atomic formulas of PSLang have one of the three forms

$$s = t \quad , \quad s \xrightarrow{i} t \quad , \quad P(t)$$

where s, t are terms and P is a predicate variable. There are, of course, various typing restrictions lurking beneath this enrichment. I won't elucidate these for they are not exposed by the examples I give in Section 11

The general procedure is to take a modal formula ϕ, convert it into a Π^1_1 sentence $\Pi(\phi)$, and then apply the algorithm to $\Pi(\phi)$. Of course, this can be done only if $\Pi(\phi)$ is a special interest sentence, but this is not a severe limitation since $\Pi(\phi)$ can be phrased as

$$(\forall \mathsf{P})(\forall a)[\neg\ulcorner\phi\urcorner(a) \rightarrow a \neq a]$$

which is a special interest sentence with just one laminar component, this having an empty prefix and just one disjunct, namely $\neg\ulcorner\phi\urcorner(a)$.

10 Sahlqvist's result

As I remarked earlier, much of the algorithm was designed by lifting Sahlqvist's result to a more general situation. Let us see how this general algorithm deals with this original modal result.

The original monomodal version of Sahlqvist's result states that each modal formula

$$\Box^m(\sigma \to \tau)$$

where

- σ satisfies a rather complicated condition,
- τ is positive

has a first order equivalent. I will show that when the algorithm is given the Π^1_1 translation of such a modal sentence, it will automatically generate the required first order equivalent. I will describe a polymodal version of the result.

The first job is to explain the "rather complicated condition" satisfied by σ. To do this I make use of the analysis of the condition given in [4, Lemma 2.6]. Thus consider rephrasing σ in standard form, i.e built up from atoms and negated atoms using only $\wedge, \vee, \Box$, and $\Diamond$. The 'complicated condition' then ensures that the Sahlqvist formula is equivalent to a conjunction of formulas

$$\Box^m(\sigma \to \tau)$$

where now each σ has a less complicated shape. The polymodal version of this shape is as follows.

26 DEFINITION. A modal formula is *untied* if it is either

- negative, or
- $[i]p$, or
- $\langle i\rangle[\bigwedge\mathcal{U}]$

for some label i, variable p, and finite collection $\mathcal{U}$ of (previously constructed) untied formulas. ■

You may like to check that in the mononodal case the untied formulas as defined above are precisely the untied formulas of [4]. (When doing this you should remember that in the monomodal case the labels are just the powers $0, 1, 2, \ldots$ of a single atomic label.)

Using these notions the polymodal version of Sahlqvist's restrictions can be given as follows.

27 DEFINITION. A *special Sahlqvist formula* is a modal formula

$$\phi \;:=\; [i](\sigma \to \tau)$$

where

- σ is an untied formula
- τ is positive.

A *general Sahlqvist formula* is a conjunction of special Sahlqvist formulas. ■

Sahlqvist's result says that the semantics of a general Sahlqvist formula is equivalent to a first order sentence. To prove this it is clearly sufficient to check that the Π^1_1 translation of a general Sahlqvist formula is a general S-sentence (for then we may apply Theorem 23).

28 LEMMA. *The translation $\ulcorner\phi\urcorner$ of an untied formula ϕ is molecular.*

Proof. If ϕ is negative then $\ulcorner\phi\urcorner$ is antitone.
If ϕ is $[i]p$ then $\ulcorner\phi\urcorner$ is

$$\begin{pmatrix} \forall x \text{ with} \\ u \xrightarrow{i} x \end{pmatrix} [P(x)].$$

Let $\mathcal{U}$ be a collection of untied formulas and set

$$\mathcal{M} \;=\; \{\ulcorner\phi\urcorner \mid \phi \in \mathcal{U}\}.$$

By hypothesis $\mathcal{M}$ is a collection of molecular formulas. But now

$$\ulcorner\langle i\rangle[\textstyle\bigwedge\mathcal{U}]\urcorner \qquad \text{is} \qquad \begin{pmatrix} \exists x \text{ with} \\ u \xrightarrow{i} x \end{pmatrix} [\textstyle\bigwedge\mathcal{M}]$$

which is also molecular. ■

With this the following result is almost immediate.

29 THEOREM. *The Π^1_1 translation of a general Sahlqvist formula is a general S-sentence, and hence each such formula is equivalent to a first order condition.*

To complete the paper I give a selection of applications of the algorithm to the semantics of modal formulas. Some of these make use of the previous Running Examples.

11 Modal examples

To begin let's have a look at four simple examples, as follows (where $i, j, k, \ldots$ are arbitrary labels).

$$\begin{array}{ll}
(a) & \langle i\rangle\,[j]p \to [k]\langle l\rangle p \\
(b) & [i]\langle j\rangle p \to \langle k\rangle\,[l]p \\
(c) & [i](\,[j]p \to [k]q) \to [l](\,[m]p \to [n]q) \\
(d) & (\,[i]\langle j\rangle p \;\wedge\; [k]\langle l\rangle q) \to [m]\langle n\rangle(p \wedge q)
\end{array}$$

I have chosen these to illustrate some of the different properties of the algorithm. The first two examples, (a) and (b), each contain just one variable. This simplifies some of the later mechanics of the algorithm. In contrast, the examples (c) and (d) contain more than one variable. Both of the formulas (a) and (c) do have first order equivalents. This ensures that the workings of the algorithm are straight forward. Formulas (b) and (d) do not have first order equivalents, so, although the algorithm is still successful, it requires the use of skolem functions.

Before we start on these examples you may like to look at some of their monomodal instances.

In example (a) we may set

$$i = j = k = l = 1 \quad \text{(say)}$$

to obtain the confluence axiom

$$\Diamond \Box p \to \Box \Diamond p.$$

The polymodal version is hardly more complicated than this. We may also set

$$\begin{array}{rlll} \text{(i)} & i = k = \emptyset, j = l = 1 & \text{to obtain} & \Box p \to \Diamond p \\ \text{(ii)} & j = k = l = \emptyset, i = 1 & \text{to obtain} & \Box p \to p \\ \text{(iii)} & i = j = \emptyset, k = l = 1 & \text{to obtain} & p \to \Box \Diamond p \\ \text{(iv)} & i = l = \emptyset, j = 1, k = 2 & \text{to obtain} & \Box p \to \Box^2 p \\ & \vdots & & \vdots \end{array}$$

which correspond to seriality, reflexivity, symmetry, transitivity, etc. In short, this example covers many of this initial correspondence results.

Setting $i = j = k = l = 1$ in (b) produces the McKinsey formula

$$\Box \Diamond p \to \Diamond \Box p$$

which is one of the simplest non first order modal formulas.

Clearly, some of the instances of (c) are universally valid. For example, setting

$$i = l \quad , \quad j = m \quad , \quad k = n$$

produces a tautological instance. Notice also that setting

$$j = k = l = \emptyset \quad , \quad i = m = n = 1$$

produces the basic monomodal axiom K,

$$\Box(p \to q) \to (\Box p \to \Box q).$$

The full polymodal version is *not* universally valid.

Setting

$$m = \emptyset \quad , \quad i = j = k = l = n = 1$$

in (d) produces

$$(\Box \Diamond p \wedge \Box \Diamond q) \to \Diamond(p \wedge q)$$

which, like (b), is one of the formulas lying just outside the scope of Sahlqvist's result.

I will now go through these four preliminary examples.

30 EXAMPLE. For labels i, j, k, l and modal variable p let

$$\phi \ ::= \ \langle i \rangle [j]p \to [k]\langle l \rangle p.$$

A direct translation of the semantics of this formula gives the sentence

$$\Pi(\phi) \ ::= \ (\forall P)(\forall a)[\Theta(a) \to \Psi(a)]$$

where

$$\Theta(a) \ ::= \ \left(\begin{array}{c} \exists b \text{ with} \\ a \xrightarrow{i} b \end{array} \right) \left(\begin{array}{c} \forall x \text{ with} \\ b \xrightarrow{j} x \end{array} \right) [P(x)]$$

$$\Psi(a) \ ::= \ \left(\begin{array}{c} \forall c \text{ with} \\ a \xrightarrow{k} c \end{array} \right) \left(\begin{array}{c} \exists y \text{ with} \\ c \xrightarrow{l} y \end{array} \right) [P(y)].$$

This is a special interest sentence with an empty outer configuration and just one laminar component which has an empty prefix and just one disjunct, $\Theta(a)$. This disjunct begins with an existential quantifier, hence an application of Empty prefix-$\exists$ produces

$$\Pi'(\phi) \ ::= \ (\forall P) \left(\begin{array}{c} \forall a, b \text{ with} \\ a \xrightarrow{i} b \end{array} \right) [\Theta'(a) \to \Psi(a)]$$

where

$$\Theta'(a) \ ::= \ \left(\begin{array}{c} \forall x \text{ with} \\ b \xrightarrow{j} x \end{array} \right) [P(x)].$$

This sentence is in kernel normal form and by inspection we see that its kernel is

$$K(u) \ ::= \ b \xrightarrow{j} u.$$

We thus set

$$\Sigma(\phi) \ ::= \ \left(\begin{array}{c} \forall a, b, c \text{ with} \\ a \xrightarrow{i} b, a \xrightarrow{k} c \end{array} \right) \left(\begin{array}{c} \exists y \text{ with} \\ c \xrightarrow{l} y \end{array} \right) [K(y)]$$

which may be expanded to

$$\Sigma(\phi) \ ::= \ \left(\begin{array}{c} \forall a, b, c \text{ with} \\ a \xrightarrow{i} b, a \xrightarrow{k} c \end{array} \right) (\exists d)[b \xrightarrow{j} d \ \wedge \ c \xrightarrow{l} d]$$

to obtain the usual equivalent. ■

31 EXAMPLE. For labels i, j, k, l and modal variable p let

$$\phi \ ::= \ [i]\langle j \rangle p \to \langle k \rangle [l]p.$$

The Π^1_1 translation of this is

$$\Pi(\phi) \ ::= \ (\forall P)(\forall a)[\Theta(a) \to \Psi(a)]$$

where

$$\Theta(a) \ ::= \ \left(\begin{array}{c} \forall x \text{ with} \\ a \xrightarrow{i} x \end{array} \right) \left(\begin{array}{c} \exists u \text{ with} \\ x \xrightarrow{j} u \end{array} \right) [P(u)]$$

$$\Psi(a) \ ::= \ \left(\begin{array}{c} \exists y \text{ with} \\ a \xrightarrow{k} y \end{array} \right) \left(\begin{array}{c} \forall v \text{ with} \\ y \xrightarrow{l} v \end{array} \right) [P(v)].$$

This is a special interests sentence with an empty outer configuration and just one laminar component, $\Theta(a)$, which has just one disjunct, namely

$$\left(\begin{array}{c}\exists u \text{ with} \\ x \xrightarrow{j} u\end{array}\right)[P(u)].$$

Thus we have to apply **Outer extension-without**. To do this we need to introduce a skolem function.

For each element a and labels i and j let

$$Sel(a,i;j)$$

be the set of all function f given by

$$f \in Sel(a,i;j) \;\Leftrightarrow\; \left(\begin{array}{c}\forall x, u \text{ with} \\ a \xrightarrow{i} x, f(x) = u\end{array}\right)[x \xrightarrow{j} u]$$

and, for convenience, set $F(a) = Sel(a,i;j)$. A use of the Axiom of Choice now shows that

$$\Theta(a) \;\Leftrightarrow\; (\exists f \in F(a))\left(\begin{array}{c}\forall x, u \text{ with} \\ a \xrightarrow{i} x, f(x) = u\end{array}\right)[P(u)]$$

and hence we obtain

$$\Pi'(\phi) \;::=\; (\forall P)\left(\begin{array}{c}\forall a, f \text{ with} \\ f \in F(a)\end{array}\right)[\Theta'(a) \to \Psi(a)]$$

where

$$\Theta'(a) \;::=\; \left(\begin{array}{c}\forall u, x \text{ with} \\ a \xrightarrow{i} x, f(x) = u\end{array}\right)[P(u)].$$

This sentence is in kernel normal form with kernel

$$K(u) \;::=\; \left(\begin{array}{c}\exists x \text{ with} \\ a \xrightarrow{i} x\end{array}\right)[f(x) = u]$$

and hence we obtain the equivalent

$$\Sigma(\phi) \;::=\; (\forall a)(\forall f \in F(a))\Xi(a,f)$$

where

$$\Xi(a,f) \;::=\; \left(\begin{array}{c}\exists y \text{ with} \\ a \xrightarrow{k} y\end{array}\right)\left(\begin{array}{c}\forall v \text{ with} \\ y \xrightarrow{l} v\end{array}\right)[K(v)].$$

This sentence $\Sigma(\phi)$ is the official result of the algorithm, however, in this case we can obtain a more understandable equivalent by introducing further skolem functions.

Let

$$G(a) = Sel(a,k;l)$$

and observe that

$$\Psi(a) \;\Leftrightarrow\; (\forall g \in G(a))\left(\begin{array}{c}\exists y, v \text{ with} \\ a \xrightarrow{k} y, g(y) = v\end{array}\right)[P(v)]$$

and similarly

$$\Xi(a,f) \;\Leftrightarrow\; (\forall g\in G(a))\begin{pmatrix} \exists y,v \text{ with} \\ a \xrightarrow{k} y, g(y)=v \end{pmatrix}[K(v)].$$

After some tidying up we find that $\Sigma(\phi)$ is equivalent to

$$\begin{pmatrix} \forall a,f,g \text{ with} \\ f\in F(a), g\in G(a) \end{pmatrix}\begin{pmatrix} \exists x,y \text{ with} \\ a \xrightarrow{i} x, a \xrightarrow{k} y \end{pmatrix}[f(x)=g(y)]$$

which has a more symmetrical shape but, of course, is *not* first order. ■

32 EXAMPLE. For labels i,j,k,l,m,n and distinct modal variables p and q let

$$\phi \;::=\; [i]([j]p \to [k]q) \to [l]([m]p \to [n]q).$$

By replacing q by $\neg q$ and taking the contrapositive, this formula may be rephrased as

$$\phi \;::=\; \langle l\rangle([m]p \;\wedge\; \langle n\rangle q) \to \langle i\rangle([j]p \;\wedge\; \langle k\rangle q).$$

The Π^1_1 translation of this is

$$\Pi(\phi) \;::=\; (\forall P,Q)(\forall a)[\Theta(a)\to\Psi(a)]$$

where

$$\Theta(a) \;::=\; \begin{pmatrix} \exists b \text{ with} \\ a \xrightarrow{l} b \end{pmatrix}\left[\begin{pmatrix} \forall x \text{ with} \\ b \xrightarrow{m} x \end{pmatrix} P(x) \;\wedge\; \begin{pmatrix} \exists c \text{ with} \\ b \xrightarrow{n} c \end{pmatrix} Q(c)\right]$$

$$\Psi(a) \;::=\; \begin{pmatrix} \exists y \text{ with} \\ a \xrightarrow{i} y \end{pmatrix}\left[\begin{pmatrix} \forall u \text{ with} \\ y \xrightarrow{j} u \end{pmatrix} P(u) \;\wedge\; \begin{pmatrix} \exists v \text{ with} \\ y \xrightarrow{k} v \end{pmatrix} Q(v)\right].$$

This is the shape of the sentence considered in Running Example A. From item 13 we know that the required predicate witnesses for P and Q are

$$K_P(u) \;::=\; (b \xrightarrow{m} u) \quad , \quad K_Q(v) \;::=\; (c=v)$$

and hence we obtain the equivalent $\Sigma(\phi)$ given by

$$\begin{pmatrix} \forall a,b,c \text{ with} \\ a \xrightarrow{l} b \xrightarrow{n} c \end{pmatrix}\begin{pmatrix} \exists y \text{ with} \\ a \xrightarrow{i} y \end{pmatrix}\left[\begin{pmatrix} \forall u \text{ with} \\ y \xrightarrow{j} u \end{pmatrix} K_P(u) \;\wedge\; \begin{pmatrix} \exists v \text{ with} \\ y \xrightarrow{k} v \end{pmatrix} K_Q(v)\right].$$

This can be expanded to

$$\begin{pmatrix} \forall a,b,c \text{ with} \\ a \xrightarrow{l} b \xrightarrow{n} c \end{pmatrix}\begin{pmatrix} \exists y \text{ with} \\ a \xrightarrow{i} y \end{pmatrix}\left[\begin{pmatrix} \forall u \text{ with} \\ y \xrightarrow{j} u \end{pmatrix}[b \xrightarrow{m} u] \;\wedge\; (y \xrightarrow{k} c)\right]$$

i.e. as

$$\begin{pmatrix} \forall a,b,c \text{ with} \\ a \xrightarrow{l} b \xrightarrow{n} c \end{pmatrix}\begin{pmatrix} \exists y \text{ with} \\ a \xrightarrow{i} y \xrightarrow{k} c \end{pmatrix}\begin{pmatrix} \forall u \text{ with} \\ y \xrightarrow{j} u \end{pmatrix}[b \xrightarrow{m} u].$$

Note that this *is* first order. ■

33 **EXAMPLE**. For labels i, j, k, l, m, n and distinct modal variables p and q let

$$\phi \ ::= \ (\ \boxed{i}\langle j\rangle p \wedge \ \boxed{k}\langle l\rangle q) \ \to \ \boxed{m}\langle n\rangle(p \wedge q).$$

The Π^1_1 translation of this ϕ is

$$\Pi(\phi) \ ::= \ (\forall P, Q)(\forall a)[\Theta(a) \to \psi(a)]$$

where

$$\Theta(a) \ ::= \ \begin{pmatrix} \forall x \text{ with} \\ a \xrightarrow{\ i\ } x \end{pmatrix} \begin{pmatrix} \exists u \text{ with} \\ x \xrightarrow{\ j\ } u \end{pmatrix} P(u) \ \wedge \ \begin{pmatrix} \forall y \text{ with} \\ a \xrightarrow{\ k\ } y \end{pmatrix} \begin{pmatrix} \exists v \text{ with} \\ y \xrightarrow{\ l\ } v \end{pmatrix} Q(v)$$

$$\Psi(a) \ ::= \ \begin{pmatrix} \forall b \text{ with} \\ a \xrightarrow{\ m\ } b \end{pmatrix} \begin{pmatrix} \exists z \text{ with} \\ b \xrightarrow{\ n\ } z \end{pmatrix} [P(z) \wedge Q(z)].$$

This is the shape of sentence considered in Running Example B. From item 18 we know that we must introduce two skolem variables f, g with associated restrictions $F(a), G(a)$ to obtain witnesses for P and Q given by

$$K_P(u) \ ::= \ \begin{pmatrix} \exists x \text{ with} \\ a \xrightarrow{\ i\ } x \end{pmatrix} [f(x) = u] \quad , \quad K_Q(v) \ ::= \ \begin{pmatrix} \exists y \text{ with} \\ a \xrightarrow{\ k\ } y \end{pmatrix} [g(y) = v].$$

Thus we obtain the equivalent

$$\Sigma(\phi) \ ::= \ \begin{pmatrix} \forall a, b, f, g \text{ with} \\ \\ a \xrightarrow{\ m\ } b \\ \\ f \in F(a), g \in G(a) \end{pmatrix} \begin{pmatrix} \exists z \text{ with} \\ b \xrightarrow{\ n\ } z \end{pmatrix} [K_P(z) \wedge K_Q(z)]$$

which, on expansion, becomes

$$\begin{pmatrix} \forall a, b, f, g \text{ with} \\ \\ a \xrightarrow{\ m\ } b \\ \\ f \in F(a), g \in G(a) \end{pmatrix} \begin{pmatrix} \exists x, y, z \text{ with} \\ \\ a \xrightarrow{\ i\ } x, a \xrightarrow{\ k\ } y \\ \\ b \xrightarrow{\ n\ } z \end{pmatrix} [f(x) = g(y) = z].$$

This is not first order. ■

This completes the preliminary examples. Let us now begin to test the effectiveness of the algorithm on more complicated formulas.

One of the ways to do this is to try the algorithm on a sequence of examples of increasing complexity and the polymodal version of the modal reduction principles of van Benthem [12] are just such a sequence. For a given modal variable p, these are the formulas

$$\begin{array}{ll} (0) & p \to \psi \\ (1) & \langle i\rangle p \to \psi \\ (2) & \langle i\rangle\ \boxed{j} p \to \psi \\ (3) & \langle i\rangle\ \boxed{j}\langle k\rangle p \to \psi \\ \ \vdots & \end{array}$$

where ψ is a formula of the shape

$$O_1 O_2 \cdots O_n p$$

where each O is either a box or a diamond operator. It is known that (0,1,2) have first order equivalents, but $(3, \cdots)$ do not. Fitch made an initial skirmish into this area in [13] where he considered the monomodal version of shape (1). Van Benthem gave a full analysis of the monomodal cases in [12] and earlier papers. You will find that the algorithm easily copes with these formulas (where, from (3) onwards, skolem functions are necessary). Let us look at the details of case (5).

34 EXAMPLE. For arbitrary labels i, j, k, l, and m, and a modal variable p let

$$\phi \ ::= \ \langle i \rangle\, [j]\, \langle k \rangle\, [l]\, \langle m \rangle p \rightarrow \psi$$

where ψ is any positive formula built up from p. After some preliminary reductions we find that the Π^1_1 translation $\Pi(\phi)$ of ϕ can be rephrased as

$$(\forall P)\begin{pmatrix} \forall a, b \text{ with} \\ a \xrightarrow{i} b \end{pmatrix}\left[\begin{pmatrix} \forall x \text{ with} \\ b \xrightarrow{j} x \end{pmatrix}\begin{pmatrix} \exists y \text{ with} \\ x \xrightarrow{k} y \end{pmatrix}\Theta(y) \rightarrow \Psi(b)\right]$$

where $\Theta ::= \ulcorner\, [l]\langle m \rangle p \urcorner$ and $\Psi ::= \ulcorner \psi \urcorner$. To deal with this consider the set $F(b)$ of skolem functions defined by

$$f \in F(b) \ \Leftrightarrow \ \begin{pmatrix} \forall x, y \text{ with} \\ b \xrightarrow{j} x, f(x) = y \end{pmatrix}[x \xrightarrow{k} y]$$

and rephrase the sentence as

$$(\forall P)\begin{pmatrix} \forall a, b, f \text{ with} \\ a \xrightarrow{i} b, f \in F(b) \end{pmatrix}\left[\begin{pmatrix} \forall x, y \text{ with} \\ b \xrightarrow{j} x, f(x) \xrightarrow{l} y \end{pmatrix}\begin{pmatrix} \exists u \text{ with} \\ y \xrightarrow{m} u \end{pmatrix} P(u) \rightarrow \Psi(b)\right].$$

To deal with this we introduce the set $G(b, f)$ of higher order skolem functions g defined by

$$g \in G(c, f) \ \Leftrightarrow \ \begin{pmatrix} \forall x, y, u \text{ with} \\ \\ b \xrightarrow{j} x, f(x) \xrightarrow{l} y \\ \\ g(x, y) = u \end{pmatrix}[y \xrightarrow{m} u].$$

(Note the type of this g.) We may now rephrase $\Pi(\phi)$ as

$$(\forall P)\begin{pmatrix} \forall a, b, f, g \text{ with} \\ \\ a \xrightarrow{i} b, f \in F(c) \\ \\ g \in G(c, f) \end{pmatrix}\left[\begin{pmatrix} \forall x, y, u \text{ with} \\ \\ b \xrightarrow{j} x, f(x) \xrightarrow{l} y \\ \\ g(x, y) = u \end{pmatrix} P(u) \rightarrow \Psi(b)\right]$$

which is in kernel normal form. The kernel is

$$K(u) \ ::= \ (\exists x, y)[(b \xrightarrow{j} x) \wedge (f(x) \xrightarrow{l} y) \wedge (g(x, y) = u)]$$

so that, for the particular case where $\psi ::= p$, we find that

$$\Sigma(\phi) \ ::= \ \begin{pmatrix} \forall a, b, c, f, g \text{ with} \\ \\ a \xrightarrow{i} b \xrightarrow{j} c \\ \\ f \in F(c), g \in G(c, f) \end{pmatrix} (\exists x, y)[(b \xrightarrow{j} x) \wedge (f(x) \xrightarrow{l} y) \wedge (g(x, y) = b)]$$

is the required equivalent to the semantics of ϕ. ■

The next example shows that under certain circumstances a use of Outer extension-with can be replaced by a use of Outer extension-without. This appears to achieve a more efficient use of skolem variables.

35 EXAMPLE. For labels i, j, k, l, m, and n, and modal variable p, consider the formula

$$\phi \ ::= \ \theta \to \psi$$

where

$$\theta ::= \langle i \rangle \boxed{j}(\langle k \rangle \boxed{l} p \ \vee \ \boxed{m} \langle n \rangle \boxed{l} p)$$

and ψ is any positive formula built up from p. For instance, setting $l = m = 0$ and $n = k$ gives

$$\phi \ ::= \ \langle i \rangle \boxed{j} \langle k \rangle p \to \psi$$

which, for appropriate choice of ψ, provides a common generalization of Examples 30 and 31. Note also that setting $i = l = m = n = 0$ gives

$$\phi \ ::= \ \Box(\Diamond p \vee p) \to \psi$$

which displays another obstruction to an application of Sahlqvist's result.

The Π^1_1 translation of this ϕ is, of course

$$\Pi(\phi) \ ::= \ (\forall P)(\forall a)[\Theta(a) \to \Psi(a)]$$

where $\Theta = \ulcorner \theta \urcorner$ and $\Psi = \ulcorner \psi \urcorner$. After some preliminary reductions we find that this is equivalent to

$$(\forall P) \begin{pmatrix} \forall a, b \text{ with} \\ a \xrightarrow{i} b \end{pmatrix} \left[\begin{pmatrix} \forall x \text{ with} \\ b \xrightarrow{j} x \end{pmatrix} \left[\Theta_1(x) \vee \begin{pmatrix} \forall y \text{ with} \\ x \xrightarrow{m} y \end{pmatrix} \Theta_2(y) \right] \ \to \ \Psi(a) \right]$$

where

$$\Theta_1(x) \ ::= \ \begin{pmatrix} \exists v \text{ with} \\ x \xrightarrow{k} v \end{pmatrix} E(v) \quad , \quad \Theta_2(y) \ ::= \ \begin{pmatrix} \exists w \text{ with} \\ y \xrightarrow{n} w \end{pmatrix} E(w)$$

where $E = \ulcorner \boxed{l} p \urcorner$. This is precisely the shape of sentence considered in Running Example D where the predicate variable Q doesn't occur in the matrix so can be omitted in the prefix. For the beginning of item 20 we know that $\Pi(\phi)$ is equivalent to

$$(\forall P) \begin{pmatrix} \forall a, b \text{ with} \\ a \xrightarrow{i} b \end{pmatrix} [\Theta''(b) \to \Psi(a)]$$

where $\Theta''(b)$ is

$$\left(\begin{array}{c}\forall x, y \text{ with} \\ b \xrightarrow{j} x \xrightarrow{m} y\end{array}\right)\left[\left(\begin{array}{c}\exists v \text{ with} \\ x \xrightarrow{k} v\end{array}\right) E(v) \vee \left(\begin{array}{c}\exists w \text{ with} \\ y \xrightarrow{n} w\end{array}\right) E(w)\right].$$

The official algorithm as carried out in item 20 now makes two separate attacks on these existential quantifiers (using $\mathsf{Outer\ extension\text{-}with}$) and so introduces two skolem variables. It then performs some rather tedious manipulations inside the matrix to achieve some appropriate sentence $\Pi^{(5)}(\phi)$. However, a simple trick at this stage leads to a much simpler calculation.

Notice that by a change of variable the matrix of $\Theta''(b)$ can be written as

$$(\exists z)[((x \xrightarrow{k} z) \wedge E(z)) \ \vee \ ((y \xrightarrow{n} z) \wedge E(z))]$$

and this is equivalent to

$$\left(\begin{array}{c}\exists z \text{ with} \\ (x \xrightarrow{k} z) \vee (y \xrightarrow{n} z)\end{array}\right)[E(z)]$$

so that $\Theta''(b)$ is equivalent to

$$\left(\begin{array}{c}\forall x, y \text{ with} \\ b \xrightarrow{j} x \xrightarrow{m} y\end{array}\right)\left(\begin{array}{c}\exists z \text{ with} \\ (x \xrightarrow{k} z) \vee (y \xrightarrow{n} z)\end{array}\right)[E(z)].$$

Let $F(b)$ be the set of skolem functions given by

$$f \in F(b) \ \Leftrightarrow \left(\begin{array}{c}\forall x, y, z \text{ with} \\ \\ b \xrightarrow{j} x \xrightarrow{m} y \\ \\ f(x,y) = z\end{array}\right)[(x \xrightarrow{k} z) \vee (y \xrightarrow{n} z)].$$

Then, by an application of $\mathsf{Outer\ extension\text{-}without}$, we see that $\Pi(\phi)$ is equivalent to

$$(\forall P)\left(\begin{array}{c}\forall a, b, f \text{ with} \\ a \xrightarrow{i} b, f \in F(b)\end{array}\right)\left[\left(\begin{array}{c}\forall x, y, z \text{ with} \\ \\ b \xrightarrow{j} x \xrightarrow{m} y \\ \\ f(x,y) = z\end{array}\right) E(z) \to \Psi(a)\right].$$

Since $E = \ulcorner [l]p \urcorner$, this may be rephrased as

$$(\forall P)\left(\begin{array}{c}\forall a, b, f \text{ with} \\ a \xrightarrow{i} b, f \in F(b)\end{array}\right)\left[\left(\begin{array}{c}\forall u, x, y \text{ with} \\ \\ b \xrightarrow{j} x \xrightarrow{m} y \\ \\ f(x,y) \xrightarrow{l} u\end{array}\right) P(u) \to \Psi(a)\right]$$

which is in kernel normal with kernel

$$K(u) \ ::= \left(\begin{array}{c}\exists x, y \text{ with} \\ b \xrightarrow{j} x \xrightarrow{m} y\end{array}\right)[f(x,y) \xrightarrow{l} u].$$

In the case where $\psi ::= p$, this produces the equivalent

$$\left(\begin{array}{c}\forall a, b, f \text{ with} \\ a \xrightarrow{i} b, f \in F(b)\end{array}\right)\left(\begin{array}{c}\exists x, y \text{ with} \\ b \xrightarrow{j} x \xrightarrow{m} y\end{array}\right)[f(x,y) \xrightarrow{l} a]$$

You may like to check that the particular case $k = m = n = 0$ produces a first order equivalent to ϕ, and work out how this can be obtained from the given general equivalent. I suggest that you also consider the variant of this Example obtained by replacing θ by

$$\theta ::= \langle i \rangle [j](\langle k \rangle [l]p \vee [m]\langle n \rangle [r]p)$$

for some label r. You will find that the algorithm breaks down, even for the apparently simpler case of $r = \emptyset$. ■

For the next example I have chosen a modal formula which, as far as I can discern, has no modal significance, but it does help to illustrate many of the facilities of the algorithm. Again we also use a trick to achieve a more efficient use of skolem variables.

36 EXAMPLE. For labels i, j, k, l, m and n, and a modal variable p, let

$$\phi ::= [i][[j](\theta_1 \vee [l](p \vee \theta_2)) \to p]$$

where

$$\theta_1 ::= \langle k \rangle p \quad , \quad \theta_2 ::= \langle m \rangle p$$

The Π^1_1 translation of ϕ is

$$\Pi(\phi) ::= (\forall P)\left(\begin{array}{c}\forall a, b \text{ with} \\ a \xrightarrow{i} b\end{array}\right)[\Xi(b) \to P(b)]$$

where

$$\Xi(b) ::= \left(\begin{array}{c}\forall x \text{ with} \\ b \xrightarrow{j} x\end{array}\right)\left[\Theta_1(x) \vee \left(\begin{array}{c}\forall y \text{ with} \\ x \xrightarrow{l} y\end{array}\right)[P(y) \vee \Theta(y)]\right]$$

where $\Theta_1 = \ulcorner\theta_1\urcorner$ and $\Theta_2 = \ulcorner\theta\urcorner$. Using $\forall$ **transfer** $\Xi(b)$ is equivalent to

$$\left(\begin{array}{c}\forall x, y \text{ with} \\ b \xrightarrow{j} x \xrightarrow{l} y\end{array}\right)[\Theta(x) \vee P(y) \vee \Theta_2(y)]$$

which we may rephrase as

$$\left(\begin{array}{c}\forall x, y \text{ with} \\ b \xrightarrow{j} x \xrightarrow{l} y\end{array}\right)(\exists u)\left[\begin{array}{l}(x \xrightarrow{k} u) \wedge P(u) \\ \vee \\ (y = u) \wedge P(u) \\ \vee \\ (y \xrightarrow{m} u) \wedge P(u)\end{array}\right]$$

or, equivalently, as

$$\left(\begin{array}{c}\forall x, y \text{ with} \\ b \xrightarrow{j} x \xrightarrow{l} y\end{array}\right)\left(\begin{array}{c}\exists u \text{ with} \\ (x \xrightarrow{k} u) \vee (y = u) \vee (y \xrightarrow{m} u)\end{array}\right)[P(u)].$$

Now let $F(b)$ be the set of skolem functions defined by

$$f \in F(b) \;\Leftrightarrow\; \begin{pmatrix} \forall u, x, y \text{ with} \\ \\ b \xrightarrow{j} x \xrightarrow{l} y \\ \\ f(x,y) = u \end{pmatrix} [(x \xrightarrow{k} u) \vee (y = u) \vee (y \xrightarrow{m} u)].$$

Then $\Xi(b)$ is equivalent to

$$(\exists f \in F(b)) \begin{pmatrix} \forall u, x, y \text{ with} \\ \\ b \xrightarrow{j} x \xrightarrow{l} y \\ \\ f(x,y) = u \end{pmatrix} [P(u)]$$

and hence $\Pi(\phi)$ is equivalent to

$$(\forall P) \begin{pmatrix} \forall a, b, f \text{ with} \\ a \xrightarrow{i} b, f \in F(b) \end{pmatrix} \left[\begin{pmatrix} \forall u, x, y \text{ with} \\ \\ b \xrightarrow{j} x \xrightarrow{l} y \\ \\ f(x,y) = u \end{pmatrix} [P(u)] \;\rightarrow\; P(b) \right]$$

which is in kernel normal form. The kernel is

$$K(u) \;::=\; \begin{pmatrix} \exists x, y \text{ with} \\ b \xrightarrow{j} x \xrightarrow{l} y \end{pmatrix} [f(x,y) = u]$$

and hence we obtain the equivalent

$$\begin{pmatrix} \forall a, b, f \text{ with} \\ a \xrightarrow{i} b, f \in F(b) \end{pmatrix} \begin{pmatrix} \exists x, y \text{ with} \\ b \xrightarrow{j} x \xrightarrow{l} y \end{pmatrix} [f(x,y) = b]$$

of ϕ. ■

For the next example I have chosen a formula which lies within the range of the Sahlqvist algorithm. To illustrate his account of Sahlqvist's result in [3, pages 105 - 107], van Benthem uses the monomodal formula

$$\Diamond(P \wedge \Box \Diamond \neg P) \rightarrow (\Diamond \Box P \vee \Box^2 \neg P)$$

which can be rephrased as

$$\Diamond(P \wedge \Box \Diamond \neg P) \wedge \Diamond^2 P \rightarrow \Diamond \Box P.$$

I deal with a polymodal generalization of this as follows.

37 EXAMPLE. For labels $i, j, k, l, m, n, r, s, t$ and modal variable p let

$$\phi \ ::= \ [i][[\langle j\rangle(\langle k\rangle\,[l]p \wedge \ [m]\langle n\rangle\neg p) \wedge \langle r\rangle p] \ \to \ \langle s\rangle\,[t]p].$$

A direct translation of the semantics of ϕ gives the Π^1_1 sentence

$$\Pi(\phi) \ ::= \ (\forall P)\left(\begin{array}{c}\forall a, b \text{ with} \\ a \xrightarrow{i} b\end{array}\right)[\Theta(b) \to \Psi(b)]$$

where

$$\Theta(b) \ ::= \ \left(\begin{array}{c}\exists c \text{ with} \\ b \xrightarrow{j} c\end{array}\right)[L(c) \wedge M(c)] \wedge N(b)$$

$$\Psi(b) \ ::= \ \left(\begin{array}{c}\exists x \text{ with} \\ b \xrightarrow{s} x\end{array}\right)\left(\begin{array}{c}\forall u \text{ with} \\ x \xrightarrow{t} u\end{array}\right)[P(u)]$$

where

$$L(c) \ ::= \ \left(\begin{array}{c}\exists e \text{ with} \\ c \xrightarrow{k} e\end{array}\right)\left(\begin{array}{c}\forall u \text{ with} \\ e \xrightarrow{l} u\end{array}\right)[P(u)]$$

$$M(c) \ ::= \ \left(\begin{array}{c}\forall x \text{ with} \\ c \xrightarrow{m} x\end{array}\right)\left(\begin{array}{c}\exists u \text{ with} \\ x \xrightarrow{n} u\end{array}\right)[\neg P(u)]$$

$$N(b) \ ::= \ \left(\begin{array}{c}\exists d \text{ with} \\ b \xrightarrow{r} d\end{array}\right)[P(d)].$$

We reduce $\Pi(\phi)$ to a first order sentence.

Several applications of Empty prefix-$\exists$ with a Monotone collection at the appropriate stage produces

$$\Pi'(\phi) \ ::= \ (\forall P)\left(\begin{array}{c}\forall a, b, c, d, e \text{ with} \\ \\ a \xrightarrow{i} b \xrightarrow{j} c \xrightarrow{k} e \\ \\ b \xrightarrow{r} d\end{array}\right)[\Theta'(d, e) \to \Psi'(b, c)]$$

where

$$\Theta'(d, e) \ ::= \ P(d) \wedge \left(\begin{array}{c}\forall u \text{ with} \\ e \xrightarrow{l} u\end{array}\right)[P(u)]$$

$$\Psi'(b, c) \ ::= \ \begin{array}{l}\left(\begin{array}{c}\exists x \text{ with} \\ b \xrightarrow{s} x\end{array}\right)\left(\begin{array}{c}\forall u \text{ with} \\ x \xrightarrow{t} u\end{array}\right)[P(u)] \\ \vee \\ \left(\begin{array}{c}\exists x \text{ with} \\ c \xrightarrow{m} x\end{array}\right)\left(\begin{array}{c}\forall u \text{ with} \\ x \xrightarrow{n} u\end{array}\right)[P(u)].\end{array}$$

This can be rephrased as

$$\Pi''(\phi) \ ::= \ (\forall P)\left(\begin{array}{c}\forall a, b, c, d, e \text{ with} \\ \\ a \xrightarrow{i} b \xrightarrow{j} c \xrightarrow{k} e \\ \\ b \xrightarrow{r} d\end{array}\right)[\Theta''(d, c) \to \Psi'(b, c)]$$

where

$$\Theta''(d,c) \ ::= \ \left(\begin{array}{c}\forall u \text{ with}\\ (d=u)\vee(e\xrightarrow{l}u)\end{array}\right)[P(u)].$$

The kernel of this is given by

$$K(u) \ ::= \ (d=u)\vee(e\xrightarrow{l}u)$$

and hence the required equivalent is

$$\Sigma(\phi) \ ::= \ \left(\begin{array}{c}\forall a,b,c,d,e \text{ with}\\ \\ a\xrightarrow{i}b\xrightarrow{j}c\xrightarrow{k}e\\ \\ b\xrightarrow{r}d\end{array}\right)\Psi''(b,c,d,e)$$

where

$$\Psi''(b,c,d,e) \ ::= \ \begin{array}{l}\left(\begin{array}{c}\exists x \text{ with}\\ b\xrightarrow{s}x\end{array}\right)\left(\begin{array}{c}\forall u \text{ with}\\ x\xrightarrow{t}u\end{array}\right)[(d=u)\vee(e\xrightarrow{l}u)]\\ \vee\\ \left(\begin{array}{c}\exists x \text{ with}\\ c\xrightarrow{m}x\end{array}\right)\left(\begin{array}{c}\forall u \text{ with}\\ x\xrightarrow{n}u\end{array}\right)[(d=u)\vee(e\xrightarrow{l}u)].\end{array}$$

This equivalent may be be expanded to:

For all elements a,b,c,d,e with

$$a\xrightarrow{i}b\xrightarrow{j}c\xrightarrow{k}e \quad , \quad b\xrightarrow{r}d$$

there is an element x such that one of

- $b\xrightarrow{s}x$ and $\left(\begin{array}{c}\forall u \text{ with}\\ x\xrightarrow{t}u\end{array}\right)[d=u \text{ or } e\xrightarrow{l}u]$
- $c\xrightarrow{m}x$ and $\left(\begin{array}{c}\forall u \text{ with}\\ x\xrightarrow{n}u\end{array}\right)[d=u \text{ or } e\xrightarrow{l}u]$

holds.

This equivalent is first order. ■

Notice that we may retrieve the original van Benthem example by setting

$$i=k=l=\emptyset \quad , \quad j=m=n=s=t=1 \quad , \quad r=2.$$

If we look at the corresponding instance of $\Sigma(\phi)$ then we get:

For all elements b,c,d with

$$b\longrightarrow c \quad , \quad b\longrightarrow\bullet\longrightarrow d$$

there is an element x with either

$$b\longrightarrow x \quad \text{or} \quad c\longrightarrow x$$

and such that the only elements u with $x\longrightarrow u$ are $u=c$ and $u=d$.

The last example again lies within the range of the Sahlqvist result, but this one contains more than one modal variable and makes use of Empty prefix-$\bigvee$.

38 EXAMPLE. For labels i, j, k, l, m and n, and modal variables p, q, let

$$\phi \ ::= \ \theta \to \psi$$

where

$$\theta \ ::= \ \langle i \rangle((\theta_1 \vee \theta_2) \wedge \neg p)$$

where

$$\theta_1 \ ::= \ \langle j \rangle(\langle k \rangle p \wedge [l]q) \quad , \quad \theta_2 \ ::= \ \langle m \rangle([n]p \wedge q)$$

and where ψ is a positive formula built up from p and q. As usual, the Π^1_1 translation of this formula is

$$\Pi(\phi) \ ::= \ (\forall P, Q)(\forall a)[\Theta(a) \to \Psi(a)]$$

where $\Theta = \ulcorner\theta\urcorner$ and $\Psi = \ulcorner\psi\urcorner$. This is the shape of sentence considered in Running Example C. From item 19 we know that

$$\Pi(\phi) \ \Leftrightarrow \ \Pi_1 \wedge \Pi_2$$

where

$$\Pi_r \ ::= \ (\forall P, Q)\left(\begin{array}{c} \forall a, b \text{ with} \\ a \xrightarrow{i} b \end{array}\right)[\Theta_r(b) \to \Psi'(a, b)]$$

for $r = 1, 2$. We attack Π_1 and Π_2 separately.

Firstly, Π_1 reduces to

$$(\forall P, Q)\left(\begin{array}{c} \forall a, b, c, d \text{ with} \\ \\ a \xrightarrow{i} b \xrightarrow{j} c \\ \\ c \xrightarrow{k} d \end{array}\right)\left[P(d) \wedge \left(\begin{array}{c} \forall u \text{ with} \\ c \xrightarrow{l} u \end{array}\right) Q(u) \ \to \ \Psi'(a, b)\right]$$

which is in kernel normal form. The kernels for P and Q are

$$K_{1,P}(u) \ ::= \ (d = u) \quad , \quad K_{1,Q}(u) \ ::= \ (c \xrightarrow{l} u).$$

Secondly, Π_2 reduces to

$$(\forall P, Q)\left(\begin{array}{c} \forall a, b, e \text{ with} \\ a \xrightarrow{i} b \xrightarrow{m} e \end{array}\right)\left[\left(\begin{array}{c} \forall u \text{ with} \\ e \xrightarrow{n} u \end{array}\right) P(u) \wedge Q(e) \ \to \ \Psi'(a, b)\right]$$

which is in kernel normal form. Here the kernels for P and Q are

$$K_{2,P}(u) \ ::= \ (e \xrightarrow{n} u) \quad , \quad K_{2,Q}(u) \ ::= \ (e = u).$$

For the sake of completeness, suppose that $\psi ::= p \wedge q$. Then

$$\Psi'(a, b) \ ::= \ (P(a) \wedge Q(a)) \vee P(b).$$

Let

$$\Sigma_1 \quad ::= \quad \begin{pmatrix} \forall a, b, c, d \text{ with} \\ \\ a \xrightarrow{i} b \xrightarrow{j} c \\ \\ c \xrightarrow{k} d \end{pmatrix} [(d = a \wedge c \xrightarrow{l} a) \vee d = b]$$

$$\Sigma_2 \quad ::= \quad \begin{pmatrix} \forall a, b, e \text{ with} \\ a \xrightarrow{i} b \xrightarrow{m} e \end{pmatrix} [(e \xrightarrow{n} a \wedge e = a) \vee e \xrightarrow{n} b]$$

so that

$$\Pi(\phi) \ \Leftrightarrow \ \Sigma_1 \wedge \Sigma_2$$

which, no doubt, can be cleaned up a little. ■

This is enough examples to keep you going for a while.

12 Final comments

The algorithm as presented here is by no means the finished article. What are its limitations and, more importantly, what are the limitations of the general method on which the algorithm is based? The crucial components of the method are the use of monotonicity and the existence of a kernel substitution. Just how extensive is this general method, and to what extent does this particular algorithm cover all these cases? I do not know an example of a sentence which has a predicate-free equivalent but this equivalent is not produced by the algorithm. Howeever, I can not believe that this is the full story.

Two particular questions need to be looked at and, with luck, answered.

Is there a Π^1_1 sentence which has a first order equivalent but no such equivalent arises as the result of witnessing the predicate variables in some way? (I suspect there are such sentences.)

For those Π^1_1 sentences which are equivalent to the conjunction of finitely many witnessed instances, is it the case the the algorithm will produce the equivalent?

The efficiency of the algorithm should be improved.

There are various tricks which it may be possible to incorporate into the the algorithm. I have not done this here because I don't fully understand what these tricks are doing (I can merely recognize certain situations when they might be useful). One such trick is to change a bound predicate variable from P to $\neg P$. This doesn't change the semantics of the parent sentence but does change its internal structure. (For instance, it can change an antitone part into a monotone part).

Another trick which I can't fully get to grips with is the one used in Example 35. This, you will recall, is related to the Running Example D. The official algorithm as carried out on this example in item 20 produce a lot of unhelpful shuffling. This was avoided in 35 by observing the two disjuncts could be coalesced. A fuller explanation of what is going on here is required.

It may be possible to iterate the use of the algorithm. Thus the first application could eliminate one batch of predicate variables in favour of another batch, and then a second application could eliminate these remaining variables. It is not clear to me whether or not this produces anything different then eliminating all the predicate variables at once. If it is not, then this opens up many possibilities.

So far I have tested the algorithm mainly on modal examples (as given in Section 11). This, of course, may be giving a wrong impression of what the algorithm can achieve (for there may be special features of such modally generated sentences which are helping, or hindering, the workings of the algorithm). Several other families of sentences should be investigated.

As present the algorithm is concerned with *logical* equivalents. It would be interesting to see a version which searches for equivalents relative to a given theory.

Again all the equivalents sought are classical. I have not investigated the extent to which the equivalents are constructive. This might be a fruitful path to follow, for the family of languages described in Section 2 fits naturally into constructive type theory.

There are some equivalents which the algorithm simply will not deal with because of a basic obstruction in the language. For instance, consider the modal formula

$$\Box(\Box p \to p) \to \Box p$$

known as Löb's formula. This is known not to have a first order equivalent but does have a perfectly clear and simple second order equivalent (which is not just the translation of its semantics). The algorithm as described here will never produce this equivalent. This is because the equivalent refers to gadgets which are not a part of the machinery of SLang, namely *sequences* of elements. It would be interesting to see how the algorithm can be extended to encompass this.

References

[1] J. F. K. van Benthem. Correspondence theory. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic, vol II*, chapter II.4, pages 167 – 247. D. Reidel and Co, 1984.

[2] H. Sahlqvist. Completeness and correspondence in the first- and second-order semantics for modal logic. In S. Kanger, editor, *Proceedings of the 3rd Scandinavian Logic Symposium*, pages 110 – 143. North Holland, 1975.

[3] J. F. K. van Benthem. *Modal and Classical Logic*. Bibliopolis, Naples, 1983.

[4] G. Sambin and V. Vaccaro. A new proof of Sahlqvist's theorem on modal definability and completeness. *J. Symbolic Logic*, 54:992 – 999, 1989.

[5] W. Ackermann. Untersuchung über das Eliminationsproblem der mathematischen Logik. *Math Annalen*, 110:390 – 413, 1935.

[6] W. Ackermann. Zum Eliminationsproblem der mathematischen Logik. *Math Annalen*, 111:61 – 63, 1935.

[7] D. Gabbay and H. J. Ohlbach. Quantifier elimination in second order predicate logic. In *Proceedings of KR 92*, pages 425 – 436, 92.

[8] A. Szalas. On the correspondence between modal and classical logic: an automated approach. *Logic and Computation*, 99.

[9] H. Simmons. How to eliminate predicate variables. Technical Report UMCS-93-1-2, Department of Computer Science, University of Manchester, 1993.

[10] V. Lifschitz. Computing circumscription. In *Proceedings of the ninth international joint conference on artificial intelligence*, pages 121 – 127, 1985.

[11] V. Lifschitz. Pointwise circumscription. In *Proceedings AAAI-86:1*, pages 406 – 410, 1985.

[12] J. F. K. van Benthem. Modal reduction principles. *J. Symbolic Logic*, 41:301 – 312, 1976.

[13] F. B. Fitch. A correlation between modal reduction priciples and properties of relations. *J. Philosopohical Logic*, 2:97 – 101, 1973.