Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

Syllogistics = monotonicity + symmetry + existential import

D.J.N. van Eijck

REPORT SEN-R0512 JULY 2005

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

Syllogistics = monotonicity + symmetry + existential import

ABSTRACT
Syllogistics reduces to only two rules of inference: monotonicity and symmetry, plus a third if one wants to take existential import into account. We give an implementation that uses only the monotonicity and symmetry rules, with an addendum for the treatment of existential import. Soundness follows from the monotonicity properties and symmetry properties of the Aristotelean quantifiers, while completeness for syllogistic theory is proved by direct inspection of the valid syllogisms. Next, the valid syllogisms are decomposed in terms of the rules they involve. The implementation uses Haskell, and is given in 'literate programming' style.

2000 Mathematics Subject Classification: 03A05, 68T01.
1998 ACM Computing Classification System: I.2.4
Keywords and Phrases: Knowledge representation, theory of reasoning, monotonic inference, syllogistics.

Syllogistics = Monotonicity + Symmetry + Existential Import

Jan van Eijck

July 4, 2005

Abstract

Syllogistics reduces to only two rules of inference: monotonicity and symmetry, plus a third if one wants to take existential import into account. We give an implementation that uses only the monotonicity and symmetry rules, with an addendum for the treatment of existential import. Soundness follows from the monotonicity properties and symmetry properties of the Aristotelean quantifiers, while completeness for syllogistic theory is proved by direct inspection of the valid syllogisms. Next, the valid syllogisms are decomposed in terms of the rules they involve. The implementation uses Haskell [8], and is given in 'literate programming' style [9].

Keywords: Knowledge representation, theory of reasoning, monotonic inference, syllogistics.

ACM Classification (1998) I.2.4.

2000 Mathematics Subject Classification 03A05, 68T01.

1 Introduction

Monotonicity reasoning with a binary quantifier Q that is monotonicity preserving (or: upward monotone) in its first argument uses the following rule of inference:

$$\frac{\mathrm{Quant}(P,Q) \qquad P \subseteq R}{\mathrm{Quant}(R,Q)}\ \mathrm{Quant}(\uparrow,\_)$$

The monotonicity rule for a binary quantifier that is monotonicity preserving in its second argument is similar:

$$\frac{\mathrm{Quant}(P,Q) \qquad Q \subseteq R}{\mathrm{Quant}(P,R)}\ \mathrm{Quant}(\_,\uparrow)$$

And here are the monotonicity rules for binary quantifiers that are monotonicity reversing:

$$\frac{\mathrm{Quant}(P,Q) \qquad R \subseteq P}{\mathrm{Quant}(R,Q)}\ \mathrm{Quant}(\downarrow,\_)$$

$$\frac{\mathrm{Quant}(P,Q) \qquad R \subseteq Q}{\mathrm{Quant}(P,R)}\ \mathrm{Quant}(\_,\downarrow)$$

Symmetry reasoning is the rule that infers $\mathrm{Quant}(Q,P)$ from $\mathrm{Quant}(P,Q)$ for symmetric quantifiers. In this paper, we will show that this is all one needs for the implementation of a system for syllogistic reasoning.

2 The Basic System

Declare a module, and define three terms, for the minor, major and middle term of a syllogistic judgement, respectively. The minor premise of a syllogistic judgement is the premise in which the subject term of the conclusion occurs, the major premise the one in which the predicate term of the conclusion occurs. For syllogistic terminology, see e.g. [11].

```
module Syllogic where

import List

data Term = A | B | C deriving (Eq,Show)
```

Propositions are expressed using the four quantifiers from the square of opposition:

```
data Proposition = All    Term Term
                 | Some   Term Term
                 | No     Term Term
                 | NotAll Term Term
                 deriving (Eq,Show)
```

The monotonicity properties of the Aristotelean quantifiers are given by:

$$\begin{array}{ccc} \mathrm{All}(\downarrow,\uparrow) & = & \mathrm{No}(\downarrow,\downarrow) \\ \downarrow & & \downarrow \\ \mathrm{Some}(\uparrow,\uparrow) & = & \mathrm{NotAll}(\uparrow,\downarrow) \end{array}$$

There are two ways to trigger a monotonicity inference: by means of a premise 'All P Q', interpreted as $P \subseteq Q$, or by means of a premise 'No P Q', interpreted as $P \subseteq \overline{Q}$. If a trigger of the form 'No P Q' is used, we have to take into account that the inferences from $P \subseteq \overline{Q}$ have a negated term.

```
infer :: Proposition -> Proposition -> [Proposition]
infer (All p q) prop = monAll prop p q
infer prop (All p q) = monAll prop p q
infer (No p q) prop  = monNo prop p q
infer prop (No p q)  = monNo prop p q
infer _ _            = []
```

Triggers of the form 'All R T' give rise to monotonicity premise $R \subseteq T$. The following function deals with the monotonicity rule with premises of the form Quant(P, Q) and $R \subseteq T$. This covers the case where there are two premises Quant(P, Q) and All(R, T), where these premises may have been presented in either order.

The second and third argument are for the terms R and T, respectively, of which it is assumed that the first is related by set inclusion to the second.

```
monAll :: Proposition -> Term -> Term -> [Proposition]
```

Since terms range over the three element set $\{A, B, C\}$, the arguments P, Q, R, T in the premises Quant(P, Q) and $R \subseteq T$ cannot all be different. We do assume, however, that P and Q denote different terms and that R and T denote different terms. Thus there are the following four possibilities for what is the middle term in the inference:

1. if $P = R$ then P is the middle terms, and the two premises are Quant(P, Q) and $P \subseteq T$,
2. if $P = T$ then P is the middle term, and the two premises are Quant(P, Q) and $R \subseteq P$,
3. if $Q = R$ then Q is the middle terms, and the two premises are Quant(P, Q) and $Q \subseteq T$,
4. if $Q = T$ then Q is the middle term, and the two premises are Quant(P, Q) and $R \subseteq Q$.

The code for `monAll` treats the four possible values for *Quant* in the premise of the form Quant(P, Q) one by one, and for each case gives the list of possible monotonicity conclusions, for each of the four ways of identifying terms.

If the quantifier Quant(P, Q) equals 'All P Q', we can draw a monotonicity consequence if $P = T$ (employing the downward monotonicity of 'All' in the first argument), or if $Q = R$ (employing the upward monotonicity of 'All' in the second argument):

```
monAll (All p q) r t
  | p == t    = [All r q]
  | q == r    = [All p t]
  | otherwise = []
```

If the quantifier Quant(P, Q) equals 'Some P Q', we can draw a monotonicity consequence if $P = R$ (employing the upward monotonicity of 'Some' in the first argument), or if $Q = R$ (employing the upward monotonicity of 'Some' in the second argument). Note that the rule also takes the symmetry of 'Some' into account:

```
monAll (Some p q) r t
  | p == r    = [Some t q, Some q t]
  | q == r    = [Some p t, Some t p]
  | otherwise = []
```

If the quantifier Quant(P, Q) equals 'No P Q', we can draw a monotonicity consequence if $P = T$ (employing the downward monotonicity of 'No' in the first argument), or if $Q = T$ (employing the downward monotonicity of 'No' in the second argument): Again, the rule takes the symmetry of 'No' into account.

```
monAll (No p q) r t
  | p == t    = [No q r, No r q]
  | q == t    = [No p r, No r p]
  | otherwise = []
```

If the quantifier Quant(P, Q) equals 'NotAll P Q', we can draw a monotonicity consequence if $P = R$ (employing the upward monotonicity of 'NotAll' in the first argument), or if $Q = T$ (employing the downward monotonicity of 'NotAll' in the second argument):

```
monAll (NotAll p q) r t
  | p == r    = [NotAll t q]
  | q == t    = [NotAll p r]
  | otherwise = []
```

From the monotonicity inferences with trigger 'No R T', we do not need to cover those with a second premise of the form 'All _ _', for these are already covered by the trigger 'All'. Of the

other ones, only those with a second premise of the form 'Some _ _' yield new conclusions by monotonicity. This is due to the restricted expressive power of our syllogistic format, where we can only deal with patterns of the form Quant$(P, \overline{Q})$ in conclusions in the case where 'Quant' equals 'Some'. Since the term in the trigger is negated, in the conclusions the 'Some' has to be replaced by its subcontrary 'Not All'. Again, if we have two premises of the forms Some(P, Q) and $R \subseteq \overline{T}$, where we assume P to be different from Q and R to be different from T, it follows from the fact that there are just three terms that there are the following four possibilities:

1. if $P = R$, we can employ the upward monotonicity of 'Some' in the first argument, and conclude Some$(\overline{T}, Q)$, or, by symmetry, Some$(Q, \overline{T})$, i.e., NotAll(Q, T).
2. if $P = T$, we can use the contraposition $T \subseteq \overline{R}$ of $R \subseteq \overline{T}$ (or, in other words, use the symmetry of 'No'), plus the upward monotonicity of 'Some' in the first argument, and conclude Some$(\overline{R}, Q)$, or, by symmetry, Some$(Q, \overline{R})$, i.e., NotAll(Q, R).
3. if $Q = R$, we can employ the upward monotonicity of 'Some' in the second argument, and conclude Some$(P, \overline{T})$, i.e., NotAll(P, T).
4. if $Q = T$, we can can use the contraposition $T \subseteq \overline{R}$ of $R \subseteq \overline{T}$ (or, in other words, use the symmetry of 'No'), plus the upward monotonicity of 'Some' in the second argument, and conclude Some$(P, \overline{R})$, i.e., NotAll(P, R).

```
monNo :: Proposition -> Term -> Term -> [Proposition]
monNo (Some p q) r t
  | p == r    = [NotAll q t]
  | p == t    = [NotAll q r]
  | q == r    = [NotAll p t]
  | q == t    = [NotAll p r]
  | otherwise = []
```

In all other cases, i.e., in the cases where the quantifier has one of the forms `All p q`, `No p q` or `Some p q`, there are no new consequences:

```
monNo _ _ _ = []
```

3 Existential Import

If we want to take existential import into account (everything that follows from the fact that all terms and their complements are assumed to have non-empty denotations), then we have to close off premises and consequences under the following rules:

$$\frac{P \subseteq Q}{P \cap Q \neq \emptyset}$$

$$\frac{P \subseteq \overline{Q}}{P \cap \overline{Q} \neq \emptyset}$$

Here is the implementation (note that, again, we take care of the symmetry of 'Some'):

```
existentialImport :: Proposition -> [Proposition]
existentialImport (All p q) = [Some p q, Some q p]
existentialImport (No  p q) = [NotAll p q]
existentialImport _         = []
```

Expanding a list of propositions with their existential imports:

```
withEI :: [Proposition] -> [Proposition]
withEI []     = []
withEI (p:ps) = (p: existentialImport p) ++ withEI ps
```

Inference with existential import is just a matter of applying existential import expansion to the syllogistic inference results, and to the syllogistic inference premises. Inference with existential import expansion of the conclusions:

```
inferCEI ::  Proposition -> Proposition -> [Proposition]
inferCEI p1 p2 = withEI (infer p1 p2)
```

Inference with existential import expansion of the premises:

```
inferPEI :: Proposition -> Proposition -> [Proposition]
inferPEI p1 p2 =
  nub (concat
        [ infer p1' p2' |
            p1' <- withEI [p1],
            p2' <- withEI [p2] ])
```

4 Soundness and Completeness

The syllogistic interpretation of the syllogistic propositions is the interpretation where the terms denote proper and non-empty subsets of a domain of discourse. The first order interpretation of the syllogistic propositions is the interpretation where the terms denote arbitrary subsets of a domain of discourse.

Soundness of the monotonicity and symmetry rules, for both the syllogistic and the first order interpretation, follows from the monotonicity properties of the quantifier denotations, plus the fact that the denotations of 'Some' and 'No' are indeed symmetric. The existential import rules are obviously sound for the syllogistic interpretation of terms. The soundness of the monotonicity and symmetry rules implies that no syllogism that is invalid for first order logic can be derived by means of monotonicity and symmetry alone. The soundness of the full calculus for the syllogistic interpretation implies that no invalid syllogism is derivable in the full system.

For completeness, we have to inspect the valid syllogistic patterns. We do this by exhaustion of the list. To understand the mnemonics, recall that the affirmative quantifiers *All* and *Some* take their names *a* and *i* from latin *affirmo* ('I affirm'), while the negating quantifiers *No* and *NotAll* derive their names *e* and *o* from latin *nego* ('I deny'). Thus, *celarent* is the mnemonic for the syllogism that has a *No* quantifier in its major premise, an *All* quantifier in its minor premise, and a *No* quantifier in its conclusion.

The syllogisms are classified according to the position of the middle term (the term that occurs in both premises but does not occur in the conclusion) in the minor and the major premise. The first figure has the middle term as subject in the major premise, as predicate in the minor premise, the second figure has the middle term as predicate in both premises, the third figure has the middle term as subject in both premises, and the fourth figure has the middle term as predicate in the major premise and as subject in the minor one.

Here are the valid syllogisms of the first figure:

```
barbara   = infer (All B C) (All A B)
celarent  = infer (No  B C) (All A B)
darii     = infer (All B C) (Some A B)
ferio     = infer (No  B C) (Some A B)
```

Of these, the ones with a universal or universal negative conclusion can also yield particular conclusions by existential import:

```
barbari  = inferCEI (All B C) (All A B)
celaront = inferCEI (No  B C) (All A B)
```

The valid syllogisms of the second figure:

```
cesare     = infer (No C B) (All A B)
camestres  = infer (All C B) (No A B)
festino    = infer (No C B) (Some A B)
baroco     = infer (All C B) (NotAll A B)
```

Again, the ones with a universal negative conclusion also yield particular conclusions by existential import:

```
cesaro     = inferCEI (No C B) (All A B)
camestrop  = inferCEI (All C B) (No A B)
```

The valid syllogisms of the third figure:

```
darapti   = inferPEI (All B C) (All B A)
disamis   = infer    (Some B C) (All B A)
datisi    = infer    (All B C) (Some B A)
felapton  = inferPEI (No B C) (All B A)
bocardo   = infer    (NotAll B C) (All B A)
ferison   = infer    (No B C) (Some B A)
```

Note that the validity of *darapti* and *felapton* hinges on the existential import of their premises.

The valid syllogisms of the fourth figure:

```
bramantip  = inferPEI (All C B) (All B A)
camenes    = infer    (All C B) (No  B A)
dimaris    = infer    (Some C B) (All B A)
fesapo     = inferPEI (No C B) (All B A)
fresison   = infer    (No C B) (Some B A)
```

The validity of *bramantip* and *fesapo* hinges on existential import.

Camenes can also yield a particular conclusion, by existential import of the universal negative conclusion:

```
camenop = inferCEI (All C B) (No  B A)
```

These are all the valid syllogistic inference forms. Inspection of the yield of the above functions for the syllogistic figures makes clear that monotonicity, symmetry and existential import are indeed a complete inference system for syllogistics.

5 Syllogism Decomposition

Every valid syllogism involves exactly one application of the monotonicity rule, either triggered by 'All' or by 'No'. Arguably, the syllogisms that just involve monotonicity are the simplest ones. A syllogism may or may not involve an application of the following rules:

1. symmetry of a premise,
2. symmetry of the conclusion,
3. existential import of a premise
4. existential import of the conclusion.

Existential import of premise and conclusion was already treated above. Below we give a new version that leaves a mark of the rule application.

First we implement a modification of the inference engine that lists whether monotonicity was triggered by 'All' or by 'No', and that also keeps track of whether 'symmetry of a premise' or 'symmetry of the conclusion' was used in an inference. The marks for indicating the rule applications are kept in a string.

```
infer' :: Proposition -> Proposition -> [(Proposition,String)]
infer' p1 p2 = infr (p1,"") (p2,"")

infr :: (Proposition,String) -> (Proposition,String)
             -> [(Proposition,String)]
infr (All p q,str1) (prop,str2) = monAll' (prop,str1 ++ str2) p q
infr (prop,str1) (All p q,str2) = monAll' (prop,str1 ++ str2) p q
infr (No p q,str1) (prop,str2)  = monNo' (prop,str1 ++ str2) p q
infr (prop,str1) (No p q,str2)  = monNo' (prop,str1 ++ str2) p q
infr _ _                        = []
```

```
monAll' :: (Proposition,String) -> Term -> Term -> [(Proposition,String)]
monAll' (All p q, str) r t
  | p == t    = [(All r q, str ++ "Ma")]
  | q == r    = [(All p t, str ++ "Ma")]
  | otherwise = []
```

```
monAll' (Some p q, str) r t
  | p == r    = [(Some t q, str ++ "Ma"), (Some q t, str ++ "MaSs")]
  | q == r    = [(Some p t, str ++ "Ma"), (Some t p, str ++ "MaSs")]
  | otherwise = []
```

```
monAll' (No p q, str) r t
  | p == t    = [(No q r, str ++ "Ma"), (No r q, str ++ "MaSn")]
  | q == t    = [(No p r, str ++ "Ma"), (No r p, str ++ "MaSn")]
  | otherwise = []
```

```
monAll' (NotAll p q, str) r t
  | p == r    = [(NotAll t q, str ++ "Ma")]
  | q == t    = [(NotAll p r, str ++ "Ma")]
  | otherwise = []
```

```
monNo' :: (Proposition,String) -> Term -> Term -> [(Proposition,String)]
monNo' (Some p q, str) r t
  | p == r    = [(NotAll q t, str ++ "MnSs")]
  | p == t    = [(NotAll q r, str ++ "SnMnSs")]
  | q == r    = [(NotAll p t, str ++ "Mn")]
  | q == t    = [(NotAll p r, str ++ "SnMn")]
  | otherwise = []
```

```
monNo' _ _ _ = []
```

Next, we program annotated versions of the existential import rules, distinguishing between existential import triggered by 'All' and existential import triggered by 'No'.

```
existentialImport' :: (Proposition,String) -> [(Proposition,String)]
existentialImport' (All p q,str) =
   [(Some p q,str ++ "Ea"),(Some q p, str ++ "EaSs")]
existentialImport' (No  p q,str) =
   [(NotAll p q, str ++ "En")]
existentialImport' _             =
   []
```

```
withEI' :: [(Proposition,String)] -> [(Proposition,String)]
withEI' []     = []
withEI' (p:ps) = (p : existentialImport' p) ++ withEI' ps
```

```
inferCEI' ::  Proposition -> Proposition -> [(Proposition,String)]
inferCEI' p1 p2 = withEI' (infer' p1 p2)
```

```
inferPEI' :: Proposition -> Proposition -> [(Proposition,String)]
inferPEI' p1 p2 =
  nub (concat
        [ infr p1' p2' |
            p1' <- withEI' [(p1,"")],
            p2' <- withEI' [(p2,"")] ])
```

Here are some examples of the generation of rule annotatations:

```
Syllogic> infer' (No  B C) (All A B)
[(No C A,"Ma"),(No A C,"MaSn")]
Syllogic> inferCEI' (All B C) (All A B)
[(All A C,"Ma"),(Some A C,"MaEa"),(Some C A,"MaEaSs")]
```

Applying this to further example syllogisms we find that *fesapo* is the most complex syllogism, in the sense that it involves two applications of symmetry (either to a premise and to a conclusion, or twice to a premise) and existential import of a premise.

```
Syllogic> infer' (No C B) (All B A)
[]
Syllogic> inferPEI' (No C B) (All B A)
[(NotAll A C,"EaSnMnSs"),(NotAll A C,"EaSsSnMn")]
```

This shows that we can decompose the syllogism *fesapo* in two ways:

$$\dfrac{\dfrac{\text{No } C\ B}{\text{No } B\ C}\ \text{Sn} \qquad \dfrac{\dfrac{\text{All } B\ A}{\text{Some } B\ A}\ \text{Ea}}{\text{Some } A\ B}\ \begin{matrix}\text{Ss}\\ \text{Mn}\end{matrix}}{\text{Some } A\ \overline{C}}$$

$$\dfrac{\dfrac{\dfrac{\text{No } C\ B}{\text{No } B\ C}\ \text{Sn} \qquad \dfrac{\text{All } B\ A}{\text{Some } B\ A}\ \text{Ea}}{\text{Some } \overline{C}\ A}\ \text{Mn}}{\text{Some } A\ \overline{C}}\ \text{Ss}$$

In an empirical set-up of [4], the inference

> No B C, All B A, therefore NotAll A C

is only recognized as valid in 8 percent of the cases, while in a staggering 61 percent of the cases, subjects think, erroneously, that the conclusion *No A C* follows from the premises. The only cases where the scores are still lower for endorsement of a valid conclusion are cases where the conclusion follows by existential import from a universal negative conclusion that is *also* valid, and that is recognized in a majority of cases as being valid.

The annotated versions of the valid syllogisms of the first figure:

```
barbara'   = infer' (All B C) (All A B)
celarent'  = infer' (No  B C) (All A B)
darii'     = infer' (All B C) (Some A B)
ferio'     = infer' (No  B C) (Some A B)
barbari'   = inferCEI' (All B C) (All A B)
celaront'  = inferCEI' (No  B C) (All A B)
```

The annotated versions of the valid syllogisms of the second figure:

```
cesare'     = infer' (No C B) (All A B)
camestres'  = infer' (All C B) (No A B)
festino'    = infer' (No C B) (Some A B)
baroco'     = infer' (All C B) (NotAll A B)
cesaro'     = inferCEI' (No C B) (All A B)
camestrop'  = inferCEI' (All C B) (No A B)
```

The annotated versions of the valid syllogisms of the third figure:

```
darapti'   = inferPEI' (All B C) (All B A)
disamis'   = infer'    (Some B C) (All B A)
datisi'    = infer'    (All B C) (Some B A)
felapton'  = inferPEI' (No B C) (All B A)
bocardo'   = infer'    (NotAll B C) (All B A)
ferison'   = infer'    (No B C) (Some B A)
```

The annotated versions of the valid syllogisms of the fourth figure:

```
bramantip'  = inferPEI' (All C B) (All B A)
camenes'    = infer'    (All C B) (No  B A)
dimaris'    = infer'    (Some C B) (All B A)
fesapo'     = inferPEI' (No C B) (All B A)
fresison'   = infer'    (No C B) (Some B A)
camenop'    = inferCEI' (All C B) (No  B A)
```

6 Related Work

Smiley [13] gives a rational reconstruction of syllogistics where there are four rules of inference: (i) From All P Q and All Q R infer All P Q, (ii) from All P Q and No Q R infer No P Q, (iii) from No Q P infer No P Q, (iv) from All Q P infer Some P Q, and where outermost negation is treated in the metatheory. He then shows that his system derives precisely the valid Aristotelean syllogisms.

An important heuristics in traditional logic is the doctrine of distribution, consisting of the following two rules:

1. the middle term of a valid syllogism has to be distributed in at least one of the premises,

2. if a term of a valid syllogism is distributed in the conclusion it has to be distributed in at least one of the premises.

Prior [11] gives the following explanation of what 'distributed' means in these rules:

> It is often said [...] that a distibuted term refers to all, and an undistributed term to only a part, of its extension. But in what way does "Some men are mortal", for example, refer to only a part of the class of men? Any man whatever will do to verify it: if any man whatever turns out to be mortal, "Some men are mortal" is true. What the traditional writers were trying to express seems to be something of the following sort: a term t is distributed in a proposition $f(t)$ if and only if it is replaceable in $f(t)$, without loss of truth, by any term "falling under it" in the way that a species falls under a genus.

This suggestion of a modern version of the doctrine of distribution is taken up in Van Benthem [3]. In Van Eijck [5] the relations between traditional logic (syllogistic theory) and generalized quantifier theory [10, 1, 2] are worked out further, with due attention to the role of monotonicity in syllogistic reasoning, and with the observation that the square of opposition generalizes to quantifiers defined from *At least n*.

Sanchez [12] is an extensive study of the role of monotonicity in 'natural reasoning', with as main contribution an algorithm for monotonicity marking, and a system for monotonicity reasoning in terms of monotonicity markings.

Hodges [7] relates the doctrine of distribution to monotonicity (just as [11, 3, 5] had done before), and gives a semantic argument to show that the correctness of the two rules of distribution follows from the interpretation of 'distributed term' as 'term in a downward monotone position'. The doctrine of distribution also follows from our completeness result. Consider the first rule of distribution, saying that the middle term has to be distributed in at least one of the premises. If the trigger of the monotonicity rule is 'No P Q', then this condition is always fulfilled, for both P and Q are in downward position. If the trigger of the monotonicity rule is 'All P Q', then the condition is fulfilled if P is the middle term, for P is in a downward position in 'All P Q', and it is also fulfilled if Q is the middle term, for the monotonicity rule allows substitution of Q by P in the other premise only if Q is in downward position in that premise. Hodges shows that the second rule of distribution follows from the first rule, as follows. Let φ and ψ be the two premises, and assume P is in downward position in $\chi(P)$, where

$$\varphi,\ \psi,\ \text{therefore}\ \chi(P)$$

is a valid syllogism. Assume, without loss of generality, that P is a term in φ, and suppose that P is in upward position in $\varphi(P)$. Then

$$\varphi(P),\ \overline{\chi}(P),\ \text{therefore}\ \overline{\psi}$$

is also a valid syllogism. But in this syllogism P is the middle term. Moreover, the effect of wide scope negation is that P is in upward position in $\overline{\chi}(P)$, and we have a contradiction with the first rule of distribution.

In Geurts [6] a monotonicity based system of reasoning for syllogistics is sketched, in terms of Sanchez-style monotonicity markings. The claim is made that monotonicity, symmetry and existential import account for all syllogistic inference, but the presentation of the rules is too informal to admit a proof of this. Geurts' intention is to explain empirical findings about accomplishment in syllogistic reasoning tasks in terms of complexity of inference in his reasoning system.

Code The Haskell code in this paper can be downloaded from `http://www.cwi.nl/~jve/papers/05/syllogic/`.

Acknowledgement Thanks to Fabian Battaglini for his help in improving the presentation.

References

[1] Barwise, J., and Cooper, R. Generalized quantifiers and natural language. *Linguistics and Philosophy 4* (1981), 159–219.

[2] Benthem, J. v. Questions about quantifiers. *Journal of Symbolic Logic 49* (1984), 443–466.

[3] Benthem, J. v. *Essays in Logical Semantics*. Reidel, Dordrecht, 1986.

[4] Chater, N., and Oaksford, M. The probability heuristics model of syllogistic reasoning. *Cognitive Psychology*, 38 (1999), 191–258.

[5] Eijck, J. v. Generalized quantifiers and traditional logic. In *Generalized Quantifiers, Theory and Applications*, J. van Benthem and A. ter Meulen, Eds. Foris, Dordrecht, 1985.

[6] Geurts, B. Reasoning with quantifiers. *Cognition*, 86 (2003), 223–251.

[7] Hodges, W. The laws of distribution for syllogisms. *Notre Dame Journal of Formal Logic 39* (1998), 221–230.

[8] Jones, S. P., Hughes, J., et al. Report on the programming language Haskell 98. Available from the Haskell homepage: `http://www.haskell.org`, 1999.

[9] Knuth, D. *Literate Programming*. CSLI Lecture Notes, no. 27. CSLI, Stanford, 1992.

[10] Mostowski, A. On a generalization of quantifiers. *Fundamenta Mathematica 44* (1957), 12–36.

[11] Prior, A. Traditional logic. In *The Encyclopedia of Philosophy*, P. Edwards, Ed., vol. 5. Macmillan, 1967, pp. 34–45.

[12] Sánchez, V. *Studies on Natural Logic and Categorial Grammar*. PhD thesis, University of Amsterdam, 1991.

[13] Smiley, T. What is a syllogism? *Journal of Philosophical Logic*, 2 (1973), 136–154.