COMPUTATIONS IN FRAGMENTS OF INTUITIONISTIC PROPOSITIONAL LOGIC

D. de Jongh
University of Amsterdam, Department of Mathematics and Computer Science

Lex Hendriks
Millingenhof 102, 1106 KG Amsterdam, the Netherlands

G.R. Renardel de Lavalette
University of Utrecht, Department of Philosophy

Logic Group
Preprint Series
No. 39
July 1988

Department of Philosophy
University of Urecht
Heidelberglaan 2
3584 CS Utrecht
The Netherlands

COMPUTATIONS IN FRAGMENTS OF INTUITIONISTIC PROPOSITIONAL LOGIC

Dick de Jongh
University of Amsterdam, Department of Mathematics and Computer Science
Roetersstraat 15, 1018 WB Amsterdam, the Netherlands

Lex Hendriks
Millingenhof 102, 1106 KG Amsterdam, the Netherlands

Gerard R. Renardel de Lavalette
University of Utrecht, Department of Philosophy
Heidelberglaan 2, 3584 CS Utrecht, The Netherlands

Abstract
This article is a report on research in progress into the structure of finite diagrams of intuitionistic propositional logic with the aid of automated reasoning systems for larger calculations. A *fragment* of a propositional logic is the set of formulae built up from a finite number of propositional variables by means of a number of connectives of the logic, among which possibly non-standard ones like $\neg\neg$ or $\leftrightarrow$ which are studied here. The *diagram* of that fragment is the set of equivalence classes of its formulae partially ordered by the derivability relation. N.G. de Bruijn's concept of exact model has been used to construct subdiagrams of the $[p,q,\wedge,\rightarrow,\neg]$-fragment.

1980 Mathematical Subject Classification: 03B04, 03B20, 68T15
1982 Computer Reviews Classification System: F.4.1 (Mechanical Theorem Proving)

Key words and phrases: intuitionistic propositional logic, fragment, diagram, mechanical theorem proving

July 1988

1. Introduction

This article is a report on research in progress into the structure of diagrams of fragments of intuitionistic propositional logic IpL with the aid of automated reasoning systems for larger calculations. A *fragment* of a propositional logic is the set formulae built up from a finite number of propositional variables by means of a number of connectives of the logic. The *diagram* of that fragment is the set of the equivalence classes of its formulae partially ordered by the derivability relation. Another way of looking at it is as a free algebra on a finite number of generators. When, in the case of intuitionistic logic, one takes all connectives, the fragment becomes a free pseudo-Boolean algebra (Rasiowa-Sikorski, 1963) on a finite number of generators. It is well-known that the free pseudo-Boolean algebra on even one generator is infinite; it is called the *Rieger-Nishimura-lattice* (Rieger 1949, Nishimura 1960). Diagrams without $\vee$ are always finite, as was in essence first proved by Diego (1966). Besides the connectives used standardly, fragments may contain more complex connectives; in the following e.g. fragments with $\neg\neg$ and $\leftrightarrow$ are studied.
Before describing more precisely the content of the research a few historical remarks are in order. In 1963, D. de Jongh and H. Kamp developed programs for deciding derivability in IpL using Kripke semantics (Kripke, 1965) and Beth's semantic tableaux (Beth, 1955). These programs were too time-consuming to be of any real use in studying diagrams. L. Hendriks (1980) picked up the study of diagrams with an Algol 68 program, and improved results were obtained by H. van Riemsdijk (1985) with the aid of a Pascal program, both using Kripke semantics and Beth tableaux. The latter was able to construct diagrams of up to a hundred elements which is not yet completely outside of the range of manual calculations. The amount of time and memory needed made it almost impossible to extend the technique to larger fragments. Larger models can be attacked with a method developed by N.G. de Bruijn. In 1975 he has introduced the concept of exact model for the study of diagrams in intuitionistic logic for fragments with implication, conjunction and negation (De Bruijn, 1975a). He implemented his ideas in an Algol 60 program, describing a decision procedure for derivability in the fragment of three propositional variables with $\rightarrow$ and $\wedge$ (De Bruijn, 1975b).
In the present article, in Sections 2 and 3, de Bruijn's concept of exact model is developed in the more familiar context of Kripke-models and simplified. Exact models are immediately fit to obtain the diagrams of $[\wedge, \rightarrow]$-fragments. However, the 2-variable $[\wedge, \rightarrow]$-diagram contains 18 elements and hence can be produced manually, whereas the 3-variable $[\wedge, \rightarrow]$-diagram already contains more than $6 \cdot 10^{14}$ elements and cannot be completely produced by a computer program, so, although the structure of the 3-variable $[\wedge, \rightarrow]$-diagram is clarified by its 61-point exact model, it is useful to study subdiagrams of it. De Bruijn (1975a) gave some ideas and results concerning the diagrams of $[\rightarrow]$-fragments and $[\wedge, \rightarrow, \neg]$-fragments within the diagrams of $[\wedge, \rightarrow]$-fragments. His ideas are transposed to Kripke-models and extended to

fragments with $\leftrightarrow$ and $\neg\neg$. In Section 4, the theory developed is used to give algorithms to compute the exact models of several fragments contained in $[p, q, \wedge, \rightarrow, \neg]$ and to construct the corresponding diagrams. Since even the smaller of these diagrams are relatively large (the smallest one, $[p, q, \leftrightarrow, \neg\neg]$, has 169 elements) it is necessary to develop methods for obtaining a global overview of the diagrams. One method is to determine in what manner the intuitionistic diagram is a refinement of the diagram in classical logic for the same fragment. Algorithms to do the latter are described. The results obtained are given in Section 5. The programs themselves have been added as an appendix. Section 6 sketches the plans for the continuation of the investigations.

It will be obvious from this introduction that we are heavily indebted to the mathematical work of N.G. de Bruijn. We thank Henk van Riemsdijk of the Mathematics and Computer Science Department of the Free University of Amsterdam for his contributions in exploring the algorithms and computer programs for computing diagrams, and also John Tromp of the Centre of Mathematics and Computer Science in Amsterdam who helped us by making the first programs using the exact models as a tool. Finally we thank Albert Visser of the Department of Philosophy of the University of Utrecht for suggesting lemma 3.1.1 to us.

2. Preliminaries.

2.1. Fragments.

The language of intuitionistic propositional logic is defined as usual, starting with the propositional variables (atoms) $p, q, r, p_1, \ldots,$ and using the connectives $\wedge, \vee, \rightarrow$ and $\neg$. Formulae of IpL are denoted by $A, B, C, A_1, \ldots$. For an axiomatization of IpL, we refer to Troelstra (1973).

A *fragment* $[p_1, \ldots, p_m, c_1, \ldots, c_n]$ of IpL is a subset of the set of formulae of IpL built up from the propositional variables $p_1, \ldots, p_m$ and using only the connectives $c_1, \ldots, c_n$. We shall consider here only the connectives $\wedge, \vee, \rightarrow, \neg, \neg\neg$ and $\leftrightarrow$, where the last two are defined by $\neg\neg A = \neg(\neg A)$ and $A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$. Sometimes, when the generating propositional variables are irrelevant, we denote a fragment by $[c_1, \ldots, c_n]$.

The *diagram* $<\mathrm{diag}(F), \vdash>$ of a fragment F is the collection of equivalence classes of F under the relation $\equiv$ (logical equivalence, defined by $A \equiv B$ iff $A \vdash B$ and $B \vdash A$), partially ordered by the derivability relation $\vdash$. We say that fragment F *contains* fragment G $(F \subseteq G)$ if the diagram of F contains (modulo logical equivalence) the diagram of G. Two fragments are called *equivalent* $(F \equiv G)$ if they contain each other. As an example, we give the diagram of $[p, q, \wedge, \rightarrow]$ (fig.1).

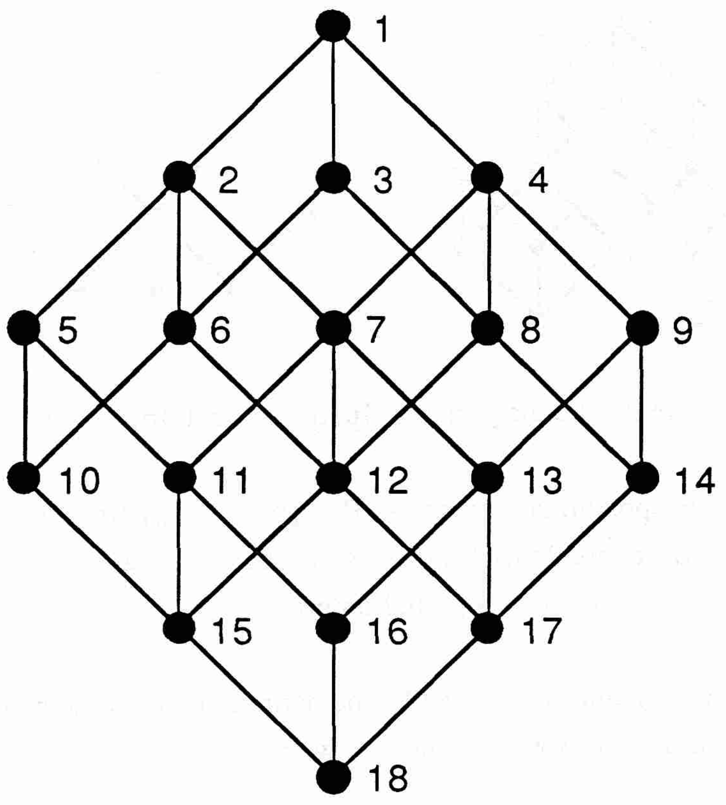

Figure 1. The diagram of $[p, q, \wedge, \to]$.

The numbers of the nodes refer to the equivalence classes of the following formulae:

1: $p \to p$
2: $((q \to p) \to q) \to q$
3: $(p \to q) \to ((q \to p) \to p)$
4: $((p \to q) \to p) \to p$
5: $q \to p$
6: $(p \to q) \to q$
7: $(((q \to p) \to q) \to q) \wedge (((p \to q) \to p) \to p)$
8: $(q \to p) \to p$
9: $p \to q$

10: $(p \to q) \to p$
11: $((p \to q) \to q) \to p$
12: $((p \to q) \to q) \wedge ((q \to p) \to p)$
13: $((q \to p) \to p) \to q$
14: $(q \to p) \to q$
15: p
16: $(p \to q) \wedge (q \to p)$
17: q
18: $p \wedge q$

The diagram of $[p, q, \to]$ is a subdiagram of fig. 1, obtained by omitting the nodes 7, 12, 16 and 18. Another subdiagram of Fig. 1 is the diagram of $[p, q, \leftrightarrow]$, which consists of the points 1, 2, 4, 7, 10, 14, 15, 16 and 17, with the following new formulae:

2: $((p \leftrightarrow q) \leftrightarrow p) \leftrightarrow q$ (analogously for 4)
7: $(((p \leftrightarrow q) \leftrightarrow p) \leftrightarrow q) \leftrightarrow (((q \leftrightarrow p) \leftrightarrow q) \leftrightarrow p)$
10: $(p \leftrightarrow q) \leftrightarrow q$ (analogously for 14)

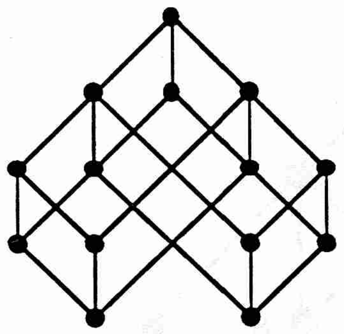
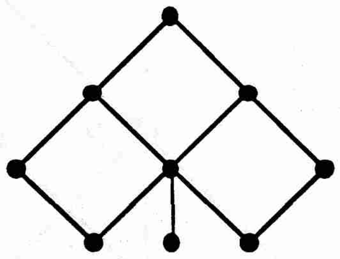

Figure 2. The diagrams of $[p, q, \rightarrow]$ and $[p, q, \leftrightarrow]$.

Given a collection of propositional constants $P = \{p_1, \ldots, p_m\}$ $(m \geq 2)$, there are *twenty-seven* different (i.e. non-equivalent) fragments F based on a subset of $\{\wedge, \vee, \rightarrow, \neg, \neg\neg, \leftrightarrow\}$. They are shown in fig. 3, as a partial ordering.

Equivalences between fragments are caused by the definability of some connectives in terms of others; besides the definitions for $\neg\neg$ and $\leftrightarrow$, we have

$$A \wedge B \equiv (A \leftrightarrow B) \leftrightarrow (A \vee B)$$
$$A \wedge B \equiv A \leftrightarrow (A \rightarrow B)$$
$$A \rightarrow B \equiv (A \wedge B) \leftrightarrow B$$
$$A \rightarrow B \equiv A \leftrightarrow (A \vee B)$$

In fragments containing $\rightarrow$, the presence of $\neg$ is equivalent to that of $\perp$, for $\neg A \equiv A \rightarrow \perp$ and $\perp \equiv \neg(p \rightarrow p)$. For technical reasons it is sometimes convenient to consider $\perp$ as primitive and $\neg$ as defined. Finally, sometimes $\top$ is written for $\neg\perp$.

As stated above, the diagram of $[p, \perp, \vee, \rightarrow]$ is the infinite so-called Rieger-Nishimura-lattice. This lattice can be embedded in the diagram of $[p, q, \vee, \rightarrow]$, so six of the twenty-seven fragments mentioned above have infinite diagrams, viz. those containing both $\vee$ and $\rightarrow$. In this paper, we confine ourselves to some of the other fragments which do have a finite diagram. The number of equivalence classes in each of the fragments with two propositional variables is given in table 1, together with the corresponding numbers for classical logic and some known numbers for three variable case.

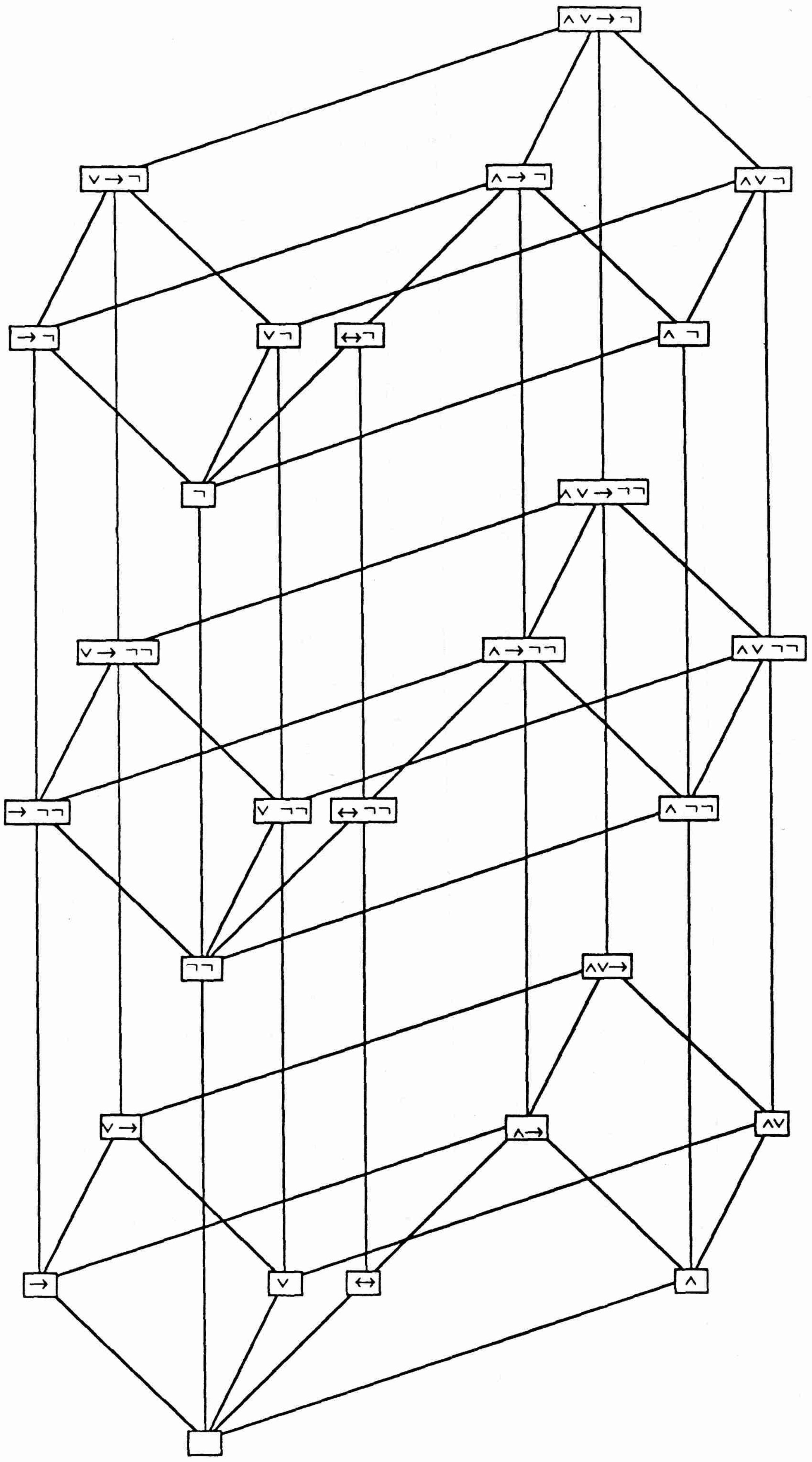

Figure 3. The inclusion diagram of twenty-seven fragments

Table 1

Connectives	classical logic			intuitionistic logic		
	number of atoms:					
	1	2	3	1	2	3
$\wedge \vee \rightarrow \neg$	4	16	256	∞	∞	∞
$\wedge \vee \neg$	4	16	256	7	683	
$\wedge \rightarrow \neg$	4	16	256	6	2134	
$\wedge \neg$	4	16	256	4	23	
$\leftrightarrow \neg$	4	8	16	6	538	
$\vee \rightarrow \neg$	4	16	256	∞	∞	∞
$\vee \neg$	4	16	256	7	385	
$\rightarrow \neg$	4	16	256	6	518	
$\neg$	2	4	6	3	6	9
$\wedge \vee \rightarrow \neg\neg$	2	8	128	5	∞	∞
$\wedge \vee \neg\neg$	1	4	18	2	19	
$\wedge \rightarrow \neg\neg$	2	8	128	4	676	
$\wedge \neg\neg$	1	3	7	2	8	
$\leftrightarrow \neg\neg$	2	4	8	4	169	
$\vee \rightarrow \neg\neg$	2	6	38	5	∞	∞
$\vee \neg\neg$	1	3	7	2	9	
$\rightarrow \neg\neg$	2	6	38	4	252	
$\neg\neg$	1	2	3	2	4	6
$\wedge \vee \rightarrow$	2	8	128	2	∞	∞
$\wedge \vee$	1	4	18	1	4	18
$\wedge \rightarrow$	2	8	128	2	18	623 662 965 552 330
$\wedge$	1	3	7	1	3	7
$\leftrightarrow$	2	4	8	2	9	
$\vee \rightarrow$	2	6	38	2	∞	∞
$\vee$	1	3	7	1	3	7
$\rightarrow$	2	6	38	2	14	25 165 802

2.2. Partially ordered sets.

Finite partially ordered sets will be used for the interpretation of formulae of IpL. From now on $X = \langle X, \leq \rangle$ is a finite p.o. set, unless stated otherwise. As usual, we write $<$ for the strict order associated to $\leq$; analogously, $\subset$ stands for strict set inclusion, i.e. $Y \subset Z$ iff $Y \subseteq Z$ and $Y \neq Z$.

We define:

$\wp^u(X) = \{Y \subseteq X \mid \forall y \in Y \forall x \geq y\ (x \in Y)\}$ (upward closed subsets)
$Y^c = \{x \mid x \in X \wedge x \notin Y\}$ (complementation)
$\mathrm{int}(Y) = \{y \mid \forall x \geq y\ (x \in Y)\}$ (upward interior)

It is easy to see that $\wp^u(X)$ is a topology on X with int as interior operation. So we have

$\mathrm{range(int)} = \mathrm{range(cl)} = \wp^u(X)$
$\mathrm{int}(Y) \subseteq Y \subseteq \mathrm{cl}(Y)$
$\mathrm{int}(Y) = Y$ iff $Y \in \wp^u(X)$
$\mathrm{int}(Y \cap Z) = \mathrm{int}(Y) \cap \mathrm{int}(Z)$

The *depth* $d(x)$ of an element of X is inductively defined by

$d(x) := \max\{d(y) \mid y > x\} + 1$, where $\max(\emptyset) = -1$.

It is obvious that $x < y$ implies $d(x) > d(y)$.

2.3. Models.

A *model* $X = \langle X, \leq, \mathrm{atom} \rangle$ of IpL is a p.o. set X together with a monotonic mapping atom which maps elements of X on sets of atoms of IpL. *Forcing* of formulae in X is defined as usual:

$x \Vdash p$ iff $p \in \mathrm{atom}(x)$
$x \Vdash A \wedge B$ iff $x \Vdash A$ and $x \Vdash B$
$x \Vdash A \vee B$ iff $x \Vdash A$ or $x \Vdash B$
$x \Vdash A \rightarrow B$ iff $\forall y \geq x\ (y \Vdash A$ implies $y \Vdash B)$
$x \Vdash \neg A$ iff $\forall y \geq x\ (y \nVdash A)$

This is the well-known Kripke forcing definition. For the defined connectives $\neg\neg$ and $\leftrightarrow$ we have:

$x \Vdash \neg\neg A$ iff $\forall y \geq x\ \exists z \geq y\ (z \Vdash A)$
$x \Vdash A \leftrightarrow B$ iff $\forall y \geq x\ (y \Vdash A$ iff $y \Vdash B)$

We also put

$X \Vdash A$ iff $\forall x \in X\ (x \Vdash A)$
$A_1, \ldots, A_n \Vdash B$ iff, for all models X: $X \Vdash A_1 \wedge \ldots \wedge A_n$ implies $X \Vdash B$

Forcing is sound and complete, i.e. $A_1, \ldots, A_n \vdash B$ iff $A_1, \ldots, A_n \Vdash B$. It is even complete w.r.t. the class of finite models, so

if $X \Vdash A$ for all finite models X, then $\vdash A$.

This fact shall be used later on, in 3.2.
Given a model X, we define the mapping $\mathrm{val}(A) := \{x \in X \mid x \Vdash A\}$ of formulae A of IpL on elements of $\wp^u(X)$. One straightforwardly verifies

$$\begin{aligned}
&\mathrm{val}(p) = \mathrm{atom}(p)\\
&\mathrm{val}(A \wedge B) = \mathrm{val}(A) \cap \mathrm{val}(B)\\
&\mathrm{val}(A \vee B) = \mathrm{val}(A) \cup \mathrm{val}(B)\\
&\mathrm{val}(A \rightarrow B) = \mathrm{int}(\mathrm{val}(A)^c \cup \mathrm{val}(B))\\
&\mathrm{val}(\neg A) = \mathrm{int}(\mathrm{val}(A)^c)
\end{aligned}$$

If X is a model, then we call $< \wp^u(X), \subseteq >$ the *diagram* of X (notation: diag(X)). By soundness, we have

(*) if $A \equiv B$, then $\mathrm{val}(A) = \mathrm{val}(B)$; if $A \vdash B$, then $\mathrm{val}(A) \subseteq \mathrm{val}(B)$.

We now consider the relation between the diagram of a fragment and the diagram of a model. Let F be a fragment and X be a model. By (*), we can consider $\mathrm{val}_F : \mathrm{diag}(F) \rightarrow \mathrm{diag}(X)$ as a well-defined order-preserving mapping. We put

$\mathrm{diag}(F) \subseteq \mathrm{diag}(X)$ if val_F is injective,
$\mathrm{diag}(F) \supseteq \mathrm{diag}(X)$ if val_F is surjective,
$\mathrm{diag}(F) = \mathrm{diag}(X)$ if val_F is bijective.

If $\mathrm{diag}(F) = \mathrm{diag}(X)$, we call X an *exact* model of F.

3. Construction of exact models.

3.1. Basic lemmas.

Not all finite fragments have an exact model: this can be seen by observing that the diagram of a model is of the form $\wp^u(X)$ and hence a (complete) lattice which is *join-representable* (i.e. where every element is the supremum of the join-irreducible elements below it), and that there are fragments whose diagram is not a join-representable lattice. Consider e.g. the diagrams of $[p, q, \rightarrow]$ and $[p, q, \leftrightarrow]$ described in 2.1: neither of them is a lattice. In general, only the fragments containing $\wedge$ have exact models, and in the sequel we focus on two of these, viz. $[\wedge, \rightarrow]$ and $[\wedge, \rightarrow, \neg]$. First we derive some general properties of formulae in these fragments. The next lemma (3.1.1) was suggested to us by A. Visser.

3.1.1. Lemma. i) Let $x \in X$ be non-maximal and let A be a formula without $\vee$. Then

(1) $\quad \mathrm{atom}(x) = \bigcap\{\mathrm{atom}(y) \mid y > x\}$ implies $\forall y > x\ (y \Vdash A) \Leftrightarrow x \Vdash A$.

ii) Let A be a formula without $\neg$. Then, for all $x \in X$:

(2) $\quad \{p \mid p \text{ occurs in } A\} \subseteq \mathrm{atom}(x) \Rightarrow x \Vdash A$.

Proof. i) Induction over the complexity of A. A prime is trivial, $A = B \wedge C$ is easy. For $A = B \rightarrow C$ we argue as follows. Assume $\forall y > x\ (y \Vdash (B \rightarrow C))$, i.e. $\forall y > x\ \forall z \geq y\ (z \Vdash B \Rightarrow z \Vdash C)$; this is equivalent to $\forall y > x\ (y \Vdash B \Rightarrow y \Vdash C)$. We want $x \Vdash (B \rightarrow C)$, i.e. $\forall y \geq x\ (y \Vdash B \Rightarrow y \Vdash C)$, so we only need $x \Vdash B \Rightarrow x \Vdash C$. But this holds, for if $x \Vdash B$, then $\forall y > x\ (y \Vdash B)$, so $\forall y > x\ (y \Vdash C)$, hence (by induction hypothesis) $x \Vdash C$. $A = \neg B$ is treated likewise.

ii) Easy. $\square$

3.1.2. Lemma. Let F be some fragment, and let $X = \langle X, \leq, \mathrm{val}_F \rangle$ be a model with val_F surjective.

i) If $\vee \notin \mathrm{con}(F)$, then

(3) $\quad \mathrm{atom}(x) \subset \bigcap\{\mathrm{atom}(y) \mid y > x\}$ for all non-maximal $x \in X$.

ii) If $\neg \notin \mathrm{con}(F)$, then

(4) $\quad \mathrm{atom}(x) \subset \mathrm{atom}(F)$ for all $x \in X$.

Proof. i) By the monotonicity of atom, we already have $\mathrm{atom}(x) \subseteq \bigcap\{\mathrm{atom}(y) \mid y > x\}$, so assume $\mathrm{atom}(x) = \bigcap\{\mathrm{atom}(y) \mid y > x\}$ for some non-maximal x. Let A be a formula of F with $\mathrm{val}_F(A) = \{y \mid y > x\}$. By 3.1.1(i), we have $x \in \{y \mid y > x\}$, so contradiction. This proves (3).

ii) Assume that F does not contain $\neg$, and let B be a formula of F with $\mathrm{val}_F(B) = \emptyset$. If there is an $x \in X$ with $\mathrm{atom}(x) = \mathrm{atom}(F)$, then (by 3.1.1(ii)) $x \in \mathrm{val}_F(B)$. Contradiction, so this x does not exist. $\square$

Remark. (4) is a consequence of (3) without the restriction to non-maximal x (reading $\mathrm{atom}(F)$ for $\bigcap\emptyset$).

3.2. Exact models for $[\wedge, \rightarrow]$.

Now let $F = [p_1, \ldots, p_n, \rightarrow, \wedge]$ be given: we define a model $X_F = < X_F, \leq_F, val_F >$ and show that it is an exact model for F. X_F is defined in n stages $X_{n-1}, \ldots, X_0$ which satisfy

$x \in X_k \Leftrightarrow \|atom(x)\| = k$

$x, y \in X_F$ and $x < y \Rightarrow x \in X_k, y \in X_l$ with $k < l$

We denote every $x \in X_F$ by $x_{Y,P}$ with $Y = \{y \in X_F \mid x <_1 y\}$, $P = atom(x)$. Now for $k = n-1, \ldots, 0$, we put

$X_k := \{x_{Y,P} \mid Y \in \wp^i(\bigcup\{X_l \mid k < l \leq n\text{-}1\}), P \subset atom(Y), \|P\| = k\}$;

here $atom(Y) := \bigcap\{atom(y) \mid y \in Y\}$, with $\bigcap\emptyset = \{p_1, \ldots, p_n\}$.

As an illustration, we give a picture of X_F for $F = [p, q, \wedge, \rightarrow]$ in fig. 4.

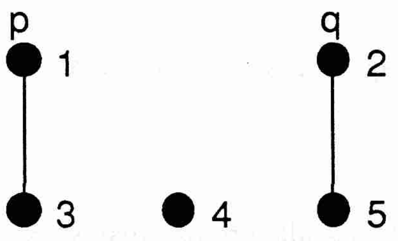

Figure 4. The exact model of $[p, q, \wedge, \rightarrow]$.

It is not hard to check that the diagram of $[p, q, \wedge, \rightarrow]$, given in fig. 1, is isomorphic to the diagram of this exact model. The exact model of $[p, q, r, \wedge, \rightarrow]$ has 61 points: the levels 2 and 1 are given in fig. 5. The diagram of this exact model has 623 662 965 552 330 points (De Bruijn 1975a), and the size of the exact model of $[p, q, r, s, \wedge, \rightarrow]$ is of the same order of magnitude.

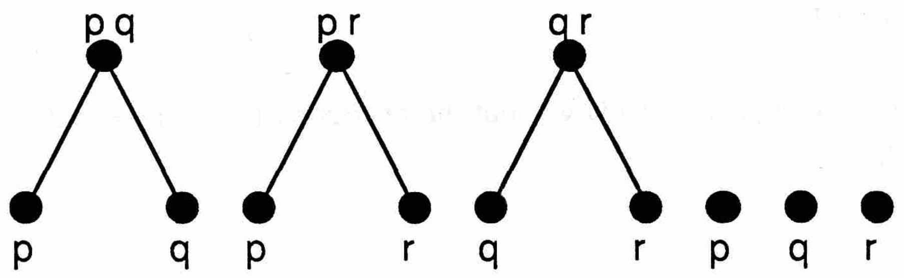

Figure 5. The first two levels of the exact model of $[p, q, r, \wedge, \rightarrow]$.

Let $F = [p_1, \ldots, p_n, \wedge, \to]$, X_F as defined above. We shall show that $\mathrm{diag}(F) \supseteq \mathrm{diag}(X_F)$, i.e. $\mathrm{val}\colon \mathrm{diag}(F) \to \mathrm{diag}(X_F)$ is surjective. This is done by giving, for every $x \in X_F$, a formula $\psi_x \in F$ which satisfies

$$\mathrm{val}(\psi_x) = \{y \in X \mid \mathrm{not}(y \leq x)\};$$

as a consequence we have, for any $Y \in \wp^u(X)$, $\bigwedge\{\psi_x \mid x \notin Y\} \in F$ and

$$\mathrm{val}(\bigwedge\{\psi_x \mid x \notin Y\}) = \bigcap\{\{y \mid \mathrm{not}(y \leq x)\} \mid x \notin Y\} = \{y \mid \forall x \geq y\ (x \in Y)\} = Y,$$

so val is surjective.
Before defining ψ_x, we introduce two abbreviations.

$$\mathrm{newatom}(x) := \bigcap\{\mathrm{atom}(y) \mid y >_1 x\} - \mathrm{atom}(x)$$
$$\Delta\ \{A_1, \ldots, A_k\} = \bigwedge\{A_i \leftrightarrow A_{i+1} \mid 1 \leq i < k\}$$

By the definition of X_F, we have $\mathrm{newatom}(x) \neq \emptyset$ for all $x \in X_F$. In the sequel, $q = q_x$ is an arbitrary element of newatom(x): the particular choice of q will be irrelevant, since it will only be used in contexts where $\Delta\mathrm{newatom}(x)$ holds, and it is easy to see that $\Delta\mathrm{newatom}(x)$ implies $q \leftrightarrow q'$ for all $q, q' \in \mathrm{newatom}(x)$.

We now simultaneously define ϕ_x and ψ_x, with induction over $n - \|\mathrm{atom}(x)\|$:

$$\phi_x := \bigwedge\mathrm{atom}(x) \wedge \Delta\mathrm{newatom}(x) \wedge \bigwedge\{\psi_y \to q \mid y >_1 x\}$$
$$\wedge \bigwedge\{\psi_z \mid \mathrm{not}(z \geq x) \text{ and } \mathrm{atom}(z) \subseteq \bigcap\{\mathrm{atom}(y) \mid y >_1 x\}\ \}$$
$$\psi_x := \phi_x \to q$$

Remarks. (1) An analogous procedure was used in De Jongh 1980, Def. 2.8 (and in De Jongh 1968, 1970) for all connectives. In De Bruijn (1975a) one finds essentially the same definition for ψ.
(2) The last conjunct of ϕ_x can be denoted by $\bigwedge\{\psi_z \mid z \in Z\}$ with $Z := \{z \mid \mathrm{not}(z \geq x)$ and $\mathrm{atom}(z) \supseteq \bigcap\{\mathrm{atom}(y) \mid y >_1 x\}\ \}$. This Z is a downward closed subset of X and can be replaced by $Z' := \{z \mid z$ is a maximal element of $Z\} \in \wp^i(X)$ in the definition of ϕ_x, since $z \geq z'$ implies $\phi_z \vdash \phi_{z'}$.

For the example $F = [p, q, \wedge, \to]$ given above in fig. 4, this comes down to

$\phi_1 = p$	$\psi_1 = p \to q$
$\phi_2 = q$	$\psi_2 = q \to p$
$\phi_3 = (p \to q) \to p$	$\psi_3 = ((p \to q) \to p) \to p$
$\phi_4 = p \leftrightarrow q$	$\psi_4 = (p \leftrightarrow q) \to p$
$\phi_5 = (q \to p) \to q$	$\psi_5 = ((q \to p) \to q) \to q$

3.2.1. Lemma. For all $x, y \in X$ we have

i) $y \Vdash \phi_x \quad \Leftrightarrow y \geq x$;

ii) $y \nVdash \psi_x \quad \Leftrightarrow y \leq x$.

Proof. Simultaneous induction over $n - \|\text{atom}(x)\|$. We denote the four conjuncts of ϕ_x by $\phi_{1,x}, \ldots, \phi_{4,x}$, respectively: so $\phi_x = \phi_{1,x} \wedge \phi_{2,x} \wedge \phi_{3,x} \wedge \phi_{4,x}$. Now, using the induction hypothesis for (7) and (8):

(5) $\quad y \Vdash \phi_{1,x} \equiv \text{atom}(y) \supseteq \text{atom}(x)$

(6) $\quad y \Vdash \phi_{2,x} \equiv \forall z \geq y\, (\text{atom}(z) \cap \text{newatom}(x) = \emptyset \vee \text{atom}(z) \supseteq \text{newatom}(x))$

(7) $\quad y \Vdash \phi_{3,x} \equiv \forall z \geq y\, (q \in \text{atom}(z) \vee \forall u >_1 x\, (z \leq u))$

(8) $\quad y \Vdash \phi_{4,x} \equiv \forall z \geq y\, (\text{atom}(z) \supseteq \text{atom}(x) \cup \text{newatom}(x) \rightarrow z \geq x)$

(i) $\Leftarrow$: it suffices to check (5-8) for $y := x$. This is easy.

$\Rightarrow$: Let $y \Vdash \phi_x$. We distinguish two cases.

a) $q \in \text{atom}(x)$: then by (6) with $z := y$ we have $\text{atom}(y) \supseteq \text{newatom}(x)$, so by (5) and (8) with $z := y$ we get $y \geq x$.

b) $q \notin \text{atom}(x)$: then $\forall u \geq x\, (y \leq u)$ by (7) with $z := y$, so $y \leq x$; by (5), we get $y=x$.

(ii) $\Rightarrow$: $x \nVdash \psi_x$ follows from $\psi_x = \phi_x \rightarrow q$, $x \Vdash \phi_x$ and $x \nVdash q$.

$\Leftarrow$: Assume $y \nVdash \psi_x$, i.e. $\exists z \geq y\, (z \Vdash \phi_x \wedge z \nVdash q)$, so by induction hypothesis $\exists z \geq y\, (z \geq x \wedge q \notin \text{atom}(z))$; but this means $\exists z \geq y\, (z=x)$, i.e. $x \geq y$. □

So we have $\text{diag}(F) \supseteq \text{diag}(X_F)$, and it remains to show that val_F is injective in order to conclude that X_F is an exact model of F. This comes down to demonstrating that X_F is complete w.r.t. F, which is a consequence of the completeness of IpL for finite models mentioned in 2.3 and the following lemma.

3.2.2. Lemma. Let X be a finite model.

i) Define (with induction over the depth of x) $x^* := < \text{atom}(x), \{y^* \mid y > x\} >$ and put $X^* := \{x^* \mid x \in X\}$. Then X^* is (isomorphic to) a subset of X, and

$X \Vdash A \Leftrightarrow X^* \Vdash A$ for all formulae A of IpL.

ii) Define $X' := \{x \in X \mid x \text{ maximal or atom}(x) \subset \bigcap\{\text{atom}(y) \mid y > x\}\}$. Then

$X \Vdash A \Leftrightarrow X' \Vdash A$ for all formulae A of IpL without $\vee$.

iii) Let P be a set of propositional variables. Define $X'' := \{x \in X \mid \text{atom}(x) \subset P\}$. Then

$X \Vdash A \Leftrightarrow X'' \Vdash A$ for all formulae A in $[P, \wedge, \vee, \rightarrow]$.

Proof. i) By proving $\forall x \in X\, (x \Vdash A \Leftrightarrow x^* \Vdash A)$ with induction over A.

ii) By lemma 3.1.1(i)

iii) By lemma 3.1.1(ii). □

3.2.3. Lemma. X_F is complete w.r.t. formulae of F.

Proof. Assume $\nvdash A$, where A is a formula of F. Then there is a finite model X with $X \nVdash A$. By lemma 3.2.2, we have $((X^*)')'' \nVdash A$, and one easily verifies that $((X^*)')''$ is a submodel of X_F, so $X_F \nVdash A$. □

Now we may conclude:

3.2.4. Theorem. X_F is the exact model of $F = [p_1, \ldots, p_n, \wedge, \rightarrow]$.

3.3. Exact models for $[\wedge, \rightarrow, \neg]$.

We now take $\neg$ into consideration and do the same as in 3.2 for $F' = [p_1, \ldots, p_n, \rightarrow, \wedge, \neg]$: we define a model $X_{F'} = < X_{F'}, \leq_{F'}, val_{F''} >$ and show that it is an exact model for F'. The treatment follows the lines of 3.3. $X = X_{F'}$ is defined exactly as in 3.3, but now in n+1 stages $X_n, \ldots, X_0$.

As an example, the model $X_{F'}$ for $F' = [p, q, \wedge, \rightarrow, \neg]$ is given in fig. 6. The numbers at the nodes are for later use (section 4). Its diagram has 2134 points (De Bruijn 1975a). The exact model of $[p, q, r, \wedge, \rightarrow, \neg]$ has 6423 points, of which 6386 in level 0; the size of its diagram still awaits computation (it is greater than 10^{1900}).

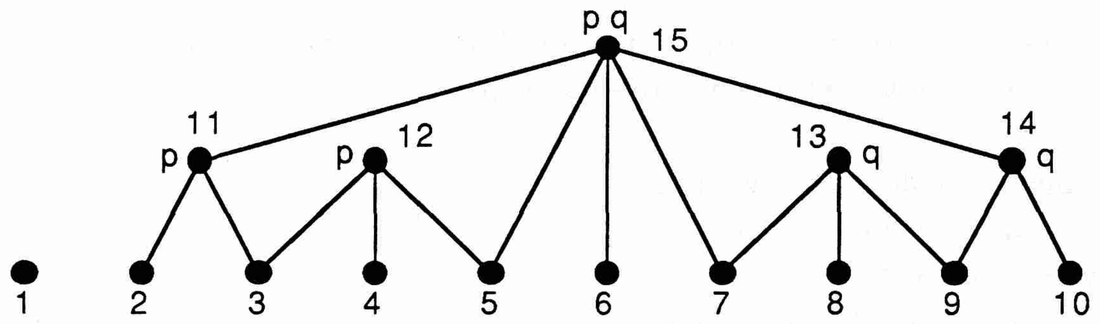

Figure 6. The exact model of $[p, q, \wedge, \rightarrow, \neg]$.

diag(F') $\supseteq$ diag($X_{F'}$) is shown using the following lemma, with ϕ_x and ψ_x defined by

$$\phi_x := \bigwedge atom(x) \wedge \bigwedge \{\neg p \mid p \notin newatom(x)\}$$
$$\psi_x := \neg\phi_x$$

for maximal x, and

$$\phi_x := \bigwedge \mathrm{atom}(x) \wedge \Delta \mathrm{newatom}(x) \wedge \bigwedge \{\psi_y \rightarrow q \mid y \in \mathrm{Pred}(x)\}$$
$$\wedge \bigwedge \{\psi_z \mid \mathrm{not}(z \geq x) \text{ and } \mathrm{atom}(z) \supseteq \bigcap \{\mathrm{atom}(y) \mid y \in \mathrm{Pred}(x)\}\ \}$$
$$\psi_x := \phi_x \rightarrow q$$

for non-maximal x.

Remark. This definition of ϕ_x and ψ_x follows directly from the one in 3.2 if we add $\perp$ to the atoms of F', with $\forall x \in X\ (\perp \notin \mathrm{atom}(x))$: observe that $\perp \in \mathrm{newatom}(x)$ iff x is maximal. We shall use this to reduce the proof of the next lemma to that of 3.2.1.

We work this out for the model in fig. 6, giving ψ_x (modulo logical equivalence) for some of the points; the other ψ's and ϕ's can be obtained easily from these. First two abbreviations.

$$\nabla p := \neg\neg p \rightarrow p$$
$$A \vee_p B := ((A \rightarrow p) \wedge (B \rightarrow p)) \rightarrow p$$

$$\psi_{15} = \neg(p \wedge q)$$
$$\psi_{11} = (p \wedge (\psi_{15} \rightarrow q)) \rightarrow q \ \equiv\ p \rightarrow \nabla q$$
$$\psi_{12} = \neg(p \wedge \neg q) \ \equiv\ p \rightarrow \neg\neg q$$
$$\psi_1 = \neg(\neg p \wedge \neg q)$$
$$\psi_2 = (\psi_{12} \wedge (\psi_{11} \rightarrow p)) \rightarrow p \ \equiv\ ((p \rightarrow q) \rightarrow p) \rightarrow ((p \rightarrow \neg\neg q) \rightarrow p)$$
$$\psi_3 = ((\psi_{12} \rightarrow p) \wedge (\psi_{11} \rightarrow p)) \rightarrow p \ \equiv\ (p \rightarrow \nabla q) \vee_p (p \rightarrow \neg\neg q)$$
$$\psi_4 = ((\psi_{12} \rightarrow p) \wedge \psi_{15}) \rightarrow p \ \equiv\ \neg q \rightarrow \nabla p$$
$$\psi_5 = ((\psi_{12} \rightarrow p) \wedge (\psi_{15} \rightarrow p) \wedge \psi_{11}) \rightarrow p \ \equiv\ (p \rightarrow \nabla q) \rightarrow ((p \rightarrow \neg\neg q) \vee_p (p \rightarrow \neg q))$$
$$\psi_6 = ((p \leftrightarrow q) \wedge (\psi_{15} \rightarrow p)) \rightarrow p \ \equiv\ (p \leftrightarrow q) \rightarrow \nabla p$$

3.3.1. Lemma. For all $x, y \in X$ we have
i) $y \Vdash \phi_x \ \Leftrightarrow\ y \geq x$;
ii) $y \nVdash \psi_x \ \Leftrightarrow\ y \leq x$.
Proof. As for 3.2.1, reading $A \rightarrow \perp$ for $\neg A$. □

3.3.2. Lemma. $X_{F'}$ is complete w.r.t. formulae of F'.
Proof. Analogously to 3.2.3: if $\nvdash A$, then $(X^*)' \nVdash A$, for some finite model X, and $(X^*)'$ is a submodel of $X_{F'}$. □

Concluding:

3.3.3. Theorem. $X_{F'}$ is the exact model of $F' = [p_1, \ldots, p_n, \wedge, \rightarrow, \neg]$.

Although many of the other finite fragments (i.e. different from $[\wedge, \rightarrow]$ and $[\wedge, \rightarrow, \neg]$) do not have exact models (see 3.1), their diagrams can often be considered as subdiagrams of the exact models constructed above. This will be exploited in the sections 4 and 5.

4. Algorithms.

4.1 Computation of a diagram using an exact model.

If F is a fragment and E is its exact model, then E can be used in the computation of the diagram of any fragment contained in F. An algorithm to perform this computation is `MakeDiagram`. The algorithm consists of two steps, an *initial step* corresponding to the valuation for the atoms in the fragment, and an *iteration step* systematically constructing the formula classes in the diagram of the fragment and the open (i.e. upward closed) sets associated to them.
In the construction a *heap* of `Elements` is used, each `Element` being a pair consisting of a formula (representing a formula class) and a subset of E. The heap will simply be called `Heap` and will be equipped with two pointers called `HeapTop` and `HeapPointer`.
In the initial step `Heap` is loaded with the atomic formulae of the fragment and the subsets of E associated to each of them. `HeapPointer` is set to the bottom of the `Heap` and `HeapTop` will point to the top of the `Heap`.
For all `Elements` below the `HeapPointer`, `Elements` are created in accordance with the connectives available in the fragment. If the subset associated to one such a newly constructed `Element` does not appear as one of the subsets associated to an `Element` in the `Heap`, the new `Element` is added to the `Heap`. Then the `HeapPointer` is reassigned to the next `Element` in the `Heap` to repeat the iteration step. This process comes to a halt when `HeapPointer` equals `HeapTop`. All formula classes of the fragment will then be represented in the `Heap` and their ordering can be read off from the inclusion relation of the associated open sets.
In the algorithm below, the a[i] ($1 \leq i \leq n$) represent the atomic formulae of the fragment and the s[i] the subsets associated to the a[i].

```
MakeDiagram:
 Begin
    ¢ Initial Step ¢
       For i:= 1 to n
          Do
             Heap[i]:= < a[i],s[i] >
             Found(s[i])
          Od
       Heap Top:= n
       HeapPointer:= 1
    ¢ Iteration Step  ¢
       While HeapPointer ≤ HeapTop
          Do
             ¢ if negation is in the fragment  ¢
                < x,y >:= NegElement(Heap[HeapPointer])
                If Not(Available(y))
                Then AddHeap(< x,y >)
                Fi
             ¢ fi ¢
             For i:= 1 to HeapPointer
                Do
                   ¢ repeat for each other connective  ∘   ¢
                     < x,y >:= ∘ Element(Heap[i]),Heap[HeapPointer])
                     If Not Available(y)
                     Then AddHeap(< x,y >)
                     Fi
                   ¢ end repeat  ¢
                Od
             HeapPointer:= HeapPointer+1
          Od
 End MakeDiagram.
```

In the iteration step the clause for ∘ is repeated for each connective ∘ in the fragment. The functions `ConElement`, `ImpElement` and `NegElement`, corresponding to conjunction, implication and negation respectively, will be spelled out below. The ∘ `-Element` functions corresponding to $\neg\neg$ and $\leftrightarrow$ are easily derived from these basic functions.
The procedure `Found` and a corresponding Boolean function `Available` keep track of the subsets found in one of the `Heap Elements`. If subsets are represented as integers (arrays of bits), `Found` might set a flag in a Boolean array F and `Available` would then test whether the flag for the set (or integer) given has already been set. The procedure `AddHeap` simply adds the element to the heap and adjusts the `HeapTop` pointer.

```
ConElement (< Forma,Seta >,< Formb,Setb >: Element) : Element
 Begin
    x:= (Forma∧Formb)
    y:= Seta∩Setb
    Return < x,y >
 End ConElement.
```

```
ImpElement (< Forma,Seta >,< Formb,Setb >: Element) : Element
  Begin
     x:= (Forma→ Formb)
     y:= Interior(Complement(Seta)∪Setb)
     Return.< x,y >
  End ImpElement.

NegElement (< Forma,Seta >: Element) : Element
  Begin
     x:= (¬Forma)
     y:= Interior(Complement(Seta))
     Return < x,y >
  End NegElement.
```

In the algorithms for these functions we assume some kind of implementation of set intersection, union and complement to be available. With the integer representation of sets in many programming languages these can be implemented with `Or`, `And` and `Not`.
The function `Interior` uses the partial ordering of the exact model E. The body of its algorithm is in fact a kind of representation of this ordering. As an example we will look at the exact model of [p, q, ∧, →, ¬] as it is given in fig. 6. The elements of E are represented by the e[i] $(1 \leq i \leq 15)$.

```
Interior(Seta: Set) : Set
  Begin
     x:= Complement(Seta)
     If e[15]∈x
     Then x:= x∪{e[5],e[6],e[7],e[11],e[14]}
     Fi
     If e[11]∈x
     Then x:=x ∪{e[2],e[3]}
     Fi
     If e[12]∈x
     Then x:=x∪ {e[3],e[4],e[5]}
     Fi
     If e[13]∈x
     Then x:=x ∪{e[7],e[8],e[9]}
     Fi
     If e[14]∈x
     Then x:=x∪ {e[9],e[10]}
     Fi
     Return Complement(x)
  End Interior.
```

Note that we used the closure operation to obtain the interior of the complemented set. The reason for this is that to construct the interior directly eleven tests would be needed instead of the five used above, and almost invariably more tests are needed to construct the interior directly than the closure; this is a consequence of the fact that many more formulas are almost intuitionistically valid then are (almost) intuitionistically contradictory. The membership test $a \in b$ could be implemented as $b = \{a\} \cup b$. In case sets are interpreted as integers, a set of elements (and hence also the union of a singleton and a set) is a simple Or-application to the corresponding integers.

4.2 Evaluating formulas, a decision procedure for derivability.

The next algorithm, MakeSet, is a function taking as its argument a formula of a fragment (not containing disjunction) and returning the corresponding open subset in an exact model. It is assumed that the fragment is contained in the fragment of the exact model. MakeSet could be used as a simple decision procedure for the formulae of the fragment. A formula A is derivable in IpL iff MakeSet(A)=E. In De Bruijn (1975b) a comparable decision procedure is implemented in an Algol 60 program.

The s[i] and a[i] are used as in MakeDiagram and the function Interior is assumed to be available. In the algorithm below we assume the exact model used to be the model of a fragment containing negation. In case negation is lacking its clause should be omitted.

```
MakeSet(Forma: Formula): Set
  Begin
     If Forma = a[i]
     Then x:= s[i]
     Elif Forma = ¬Formb
     Then x:= Interior(Complement(MakeSet(Formb)))
     Elif Forma = (Formb∧Formc)
     Then x:= MakeSet(Formb)∩MakeSet(Formc)
     Elif Forma = (Formb → Formc)
     Then x:=Interior(Complement(MakeSet(Formb)∪MakeSet(Formc)))
     Elif Forma = (Formb ↔ Formc)
     Then x:=Interior(Complement(MakeSet(Formb)∪MakeSet(Formc)))
          ∩Interior(Complement(MakeSet(Formc)∪MakeSet(Formb)))
     Else Error
     Fi
     Return x
  End MakeSet.
```

4.3 Embedding classical diagrams into the intuitionistic ones.

Diagrams tend to be large in intuitionistic logic, and hence one likes to find ways to get some global overview of the diagram. One way of doing this is by partitioning the set of formula classes by collecting those equivalence classes together which contain classically equivalent formulae. We shall call such a set of equivalence classes which are classically equivalent a *component*. In classical propositional logic all formulae are equivalent to a formula in the [∧, ¬]-fragment. We assume such a representation to be given for the classical fragment contained in the intuitionistic one we want to partition.

The algorithm ClassicalPartition will take the Heap constructed in MakeDiagram together with a formula, ClassForm, as its input and make a new heap of Elements, Component, of all formula classes in the diagram of the fragment equivalent with ClassForm in classical logic (together with the subsets associated to them in the exact model). We use the subset ClassSet of E associated with ClassForm to test for classical equivalence, using the well-known fact that a formula A is a tautology of classical logic iff $\neg\neg A$ is a theorem of intuitionistic logic.

```
ClassicalPartition(ClassForm: Formula)
  Begin
     ClassSet:=MakeSet(ClassForm)
     For i:=1 to HeapTop
        Do
           x:=Interior(Complement(Heap[i]·Set))
           x:=Interior(Complement(x))
           If x=ClassSet
           Then AddComponent(Heap[i])
           Fi
        Od
  End ClassicalPartition.
```

Here Heap[i]·Set denotes the subset of E in the Element Heap[i], and AddComponent is defined as AddHeap above. The ordering of the equivalence classes in Component can again be derived from the inclusion relation of the associated subsets in the exact model E.

4.4 Drawing a Diagram.

A heap like Heap or Component could simply be listed, but a more instructive picture of a diagram or component can be obtained by a kind of topological sorting of such a heap: levels are assigned to the elements, starting with level zero for all elements not containing any other element in the heap and working upwards increasing the level to n+1 for each element containing an element of level n. (An element Elementa is said to contain an Elementb if Elementa·Set is included in Elementb·Set.) After this sorting the elements within a level can be ordered according to their relationships with elements above or below. Using these orderings a neater picture of the diagram may be constructed. The levels and the ordering within each level can be used as co-ordinates in drawing the elements of the diagram after which the elements are joined by lines indicating the partial ordering (see figs. 1, 2). The structure of larger diagrams may however sometimes be clarified better, when one applies the above procedure to some subdiagrams and then shifts these subdiagrams around till the pattern, which is a kind of diagram too, becomes clearer (see fig. 7).

4.5 The construction of exact models.

The construction of exact models for intuitionistic fragments $[p_1, \ldots, p_n, \wedge, \rightarrow, \neg]$ is outlined in the algorithm MakeModel below. Each element x in the model will be a pair $< A, B >$ with A a subset of $\{p_1, \ldots, p_n\}$ and B the set of predecessors of x. In its initial step the algorithm introduces elements with no predecessors for all subsets of $\{p_1, \ldots, p_n\}$. In the iteration step of the algorithm *independent* subsets F of the set of elements already constructed (i.e. no ele-

ment in the subset is a predecessor of any other) give rise to new elements if the intersection of their subsets $C \subseteq \{p_1, \ldots, p_n\}$ is non-empty. For every proper subset of C a new element will be introduced, having the elements of F and their predecessors as its predecessors. An algorithm for exact models of fragments not containing negation is obtained from `MakeModel` by omitting, from the initial step, the element corresponding to $\{p_1, \ldots, p_n\}$ itself.
In the algorithm we use a heap of `Elements`, each consisting of a subset of the alphabet $\{p_1, \ldots, p_n\}$ and a subset of the set of elements (pointers). The heap will be called `Heap` again and `HeapPointer` and `HeapTop` will function as before. At the start of the iteration step `HeapPointer` will be set to the bottom of the `Heap`. `Heap[i]·Prop` will denote the subset of propositional variables of the i-th element in the heap, and `Heap[i]·Pred` will denote the set of predecessors of the i-th element.
All the subsets of $\{p_1, \ldots, p_n\}$ are constructed in the initial step, and then used again in the iteration step. (See the procedure `AddSubsetElement` below.) The part of the heap containing these 'basic' elements will be indicated by a pointer `BasicTop`.

```
MakeModel
  Begin
     ¢initial step  ¢
        For i:=1 to n
           Do
              AddHeap(<{p_i},∅>)
              For j:=1 to Heaptop-1
                 Do
                   AddHeap(<Heap[j]·Prop∪{p_i},∅>)
                 Od
              AddHeap(<∅,∅>)
              BasicTop:=Heap Top
           ¢iteration step  ¢
              While BasicTop≤Heap Top
                 Do
                   MakeSubsetElement({Heap[HeapPointer]})
                   HeapPointer:=HeapPointer+1
                 Od
  End MakeModel.
```

The `MakeSubsetElement` procedure recursively searches for all independent subsets (under some constraints) in a given subset.

```
MakeSubsetElement(Subseta: set of Element)
  Begin
     AddSubsetElement(Subseta)
        For i:=1 to Heaptop
           Do
              If Independent(Heap[i],Subseta)
              Then MakeSubsetElement(Subseta∪{Heap[i]})
              Fi
           Od
  End MakeSubsetElement.
```

The Boolean function Independent checks whether the union of an element x and an independent set A is an independent set; the procedure AddSubset adds elements for these sets.

```
Independent(Elementa:Element,Subseta:set of Element):Boolean
  Begin
    Result := Not(Elementa∈Subseta Or Subseta = ∅)
    Intersection:={p₁,…,pₙ}
    i:=1
    While Result And i<HeapTop
      Do
        If Heap[i]∈Subseta
        Then Intersection:=Intersection∩Heap[i]·Prop
          If  Elementa∈Heap[i]·Pred Or Heap[i]∈Elementa·Pred
          Then Result:=False
          Fi
        Fi
      Od
    Result:=Result And Not(Intersection ∩Elementa·Prop=∅)
    Return Result
   End Independent.
```

The elements for these independent subsets are added to the heap by the procedure AddSubsetElement.

```
AddSubsetElement(Subseta: set of Element)
  Begin
    Intersection:={p₁,…,pₙ}
    For i:=1 to BasicTop
      Do
        ToAdd:=True
        For j:=1 to Heap Top
          Do
            If Heap[j]∈Subseta
            Then
              If Heap[i]·Prop⊆Heap[j]·Prop
              Then Intersection:= Intersection ∩Heap[j]·Prop
              Else ToAdd:=False
              Fi
            Fi
          Od
        ToAdd:=ToAdd And Not(Heap[i]·Prop=Intersection)
        If ToAdd
        Then AddHeap(<Heap[i]·Prop,Subseta>)
        Fi
      Od
  End AddSubsetElement.
```

In these `MakeModel` algorithms it is assumed that the `AddHeap` procedure will not add an element twice, i.e. the element $< A, B >$ will not be added if $< A, B >$ is already present in the heap.

5. Results.

Using a program implementing the `MakeDiagram` algorithm the diagrams of a number of fragments contained in $[p, q, \wedge, \rightarrow, \neg]$ were constructed. The number of equivalence classes obtained in each of these fragments was (afterwards) proved correct 'manually', also by means of the theory developed in Section 3. The results are summarized in Table 1. The diagram of the $[p, q, \leftrightarrow, \neg\neg]$-fragment is shown in fig. 7.
The classical partitioning of the fragments is shown in Table 2. A number of features are remarkable and warrant further theoretical study. Among them are the following. The components corresponding to p and $p \rightarrow q$ (and symmetrically, to q and $q \rightarrow p$) turn out to be isomorphic in all fragments in which both are non-empty. The same holds for $p \vee q$ and T. In $[p, q, \rightarrow, \neg\neg]$ the components for $p, q, p \rightarrow q$ and $q \rightarrow p$ do have an exact model of their own, a set of five independent elements. In $[p, q, \rightarrow, \neg]$ the exact models for these components are sets of six elements. In $[p, q, \rightarrow, \neg\neg]$ and $[p, q, \rightarrow, \neg]$ the components for $p \wedge q$ and $p \leftrightarrow q$ are isomorphic copies of the diagram of $[p, q, \wedge, \rightarrow]$. In $[p, q, \rightarrow, \neg\neg]$ and $[p, q, \rightarrow, \neg]$ the components corresponding to $p \leftrightarrow q$ are isomorphic to the diagram of $[p, q, \leftrightarrow]$.

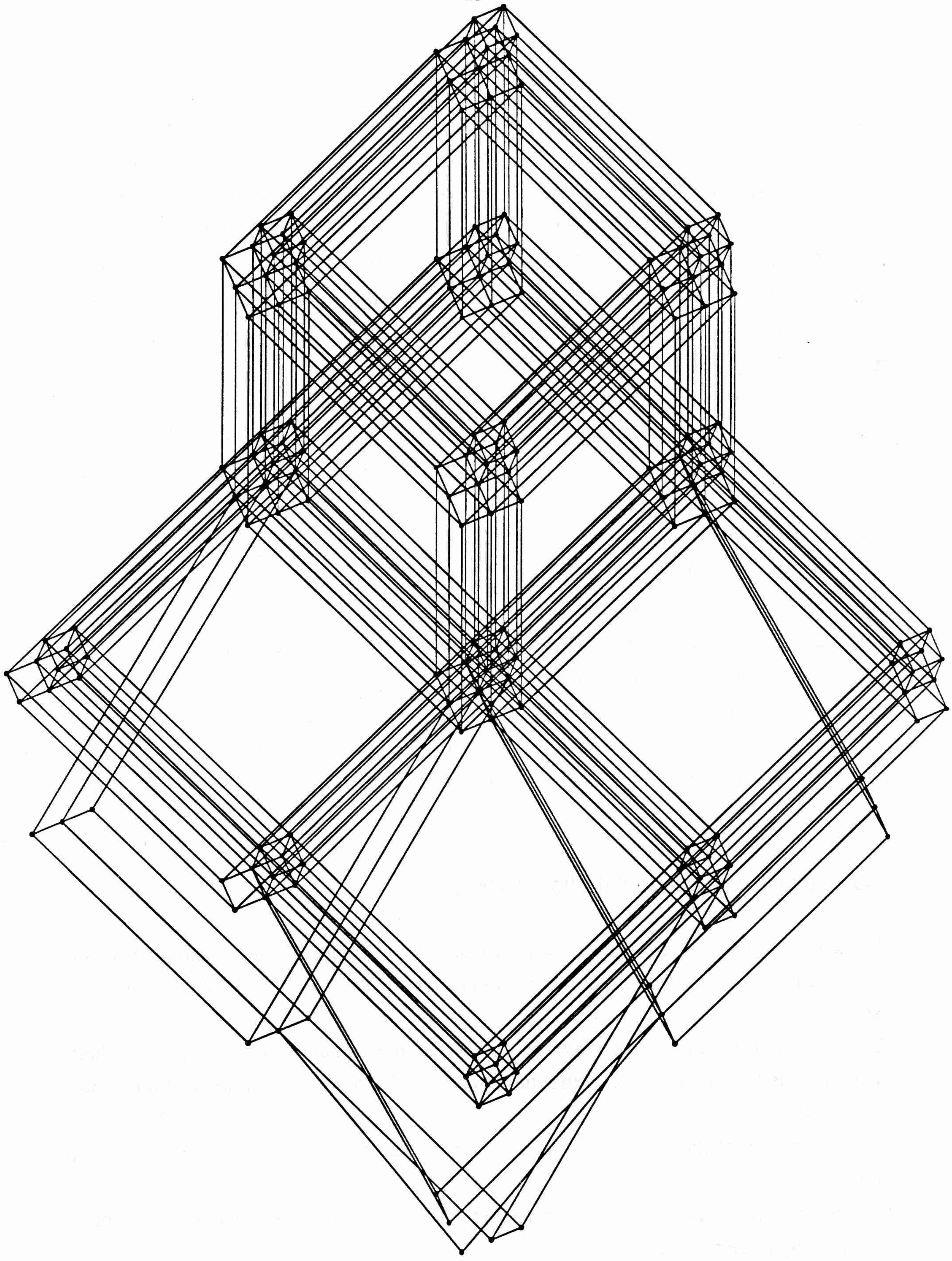

Figure 7. The diagram of the fragment $[p,q,\leftrightarrow,\urcorner]$.

Table 2

	$[\to]$	$[\leftrightarrow]$	$[\wedge]$	$[\to,\wedge]$	$[\neg\neg]$	$[\to,\neg\neg]$	$[\leftrightarrow,\neg\neg]$	$[\wedge,\neg\neg]$	$[\to,\wedge,\neg\neg]$	$[\neg]$	$[\to,\neg]$	$[\leftrightarrow,\neg]$	$[\wedge,\neg]$	$[\wedge,\to,\neg]$
#	14	9	3	18	4	252	169	8	676	6	518	538	23	2134
$\perp$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
$p\wedge q$	0	0	1	1	0	0	0	4	18	0	1	0	4	18
$p\wedge\neg q$	0	0	0	0	0	0	0	0	0	0	1	0	2	2
$\neg p\wedge q$	0	0	0	0	0	0	0	0	0	0	1	0	2	2
$\neg p\wedge\neg q$	0	0	0	0	0	0	0	0	0	0	1	0	1	1
$p\leftrightarrow q$	0	1	0	1	0	0	9	0	18	0	1	9	1	18
p	2	2	1	2	2	32	30	2	60	2	64	60	2	120
q	2	2	1	2	2	32	30	2	60	2	64	60	2	120
$\neg(p\leftrightarrow q)$	0	0	0	0	0	0	0	0	0	0	1	4	1	4
$\neg p$	0	0	0	0	0	0	0	0	0	1	1	2	1	2
$\neg q$	0	0	0	0	0	0	0	0	0	1	1	2	1	2
$p\to q$	2	0	0	2	0	32	0	0	60	0	64	0	1	120
$q\to p$	2	0	0	2	0	32	0	0	60	0	64	0	1	120
$\neg(p\wedge q)$	0	0	0	0	0	0	0	0	0	0	1	0	1	4
$p\vee q$	3	0	0	4	0	62	0	0	200	0	126	0	1	800
$\top$	3	4	0	4	0	62	100	0	200	0	126	400	1	800

6. Plans for the continuation of the Investigations.

(1) Using the 61-point exact model for the $[p, q, r, \wedge, \to]$-fragment for computations of sub-diagrams just as above the 15-point exact model for the $[p, q, \wedge, \neg, \to]$-fragment has been used.

(2) More careful study of the diagrams of the fragments obtained above to get a better theoretical understanding of their structure, perhaps with more general applications to intuitionistic logic.

(3) Construction of the 6423-point exact model for the $[p, q, r, \wedge, \neg, \to]$-fragment and applying it.

(4) Generalizing the concept of exact model to the (finite) fragments containing $\vee$ but not $\to$ and studying these fragments.

(5) Getting good theoretical descriptions of the subdiagrams of the diagrams of the $[p_1, \ldots, p_n, \wedge, \neg, \rightarrow]$-fragments, e.g. the $[p_1, \ldots, p_n,, \neg, \rightarrow]$-, $[p_1, \ldots, p_n,, \neg, \rightarrow]$-, $[p_1, \ldots, p_n, \wedge, \neg\neg, \rightarrow]$-, $[p_1, \ldots, p_n,, \leftrightarrow]$ and $[p_1, \ldots, p_n, \neg\neg, \leftrightarrow]$-fragments.

(6) The study of the validity of the interpolation property for the different fragments. In Renardel de Lavalette (1987) it has been shown for all the fragments here except the ones containing $\leftrightarrow$ (without $\rightarrow$) and the ones with $\neg\neg$ that the interpolation theorem holds. For the remaining fragments known methods fail. From the diagrams obtained, some good candidate formulae for a counterexample to the interpolation property might be found, or otherwise it may become clear that "small" counterexamples do not exist. In fact, from the diagrams with two propositional variables it has already be checked that for none of the above fragments a counterexample $A(p, q) \vdash B(q, r)$ exists; so a study of the fragments with three variables is indicated.

(7) Automatic drawing of diagrams.

7. Bibliography

Beth, E.W. (1955), Semantic Entailment and Formal Derivability, *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde,* N.R., Vol.18, no. 13, pp.309-342

De Bruijn, N.G.(1975a), Exact Finite Models for Minimal Propositional Calculus over a Finite Alphabet, Technological University Eindhoven, Report 75-WSK-02

De Bruijn, N.G.(1975b), An Algol Program for deciding Derivability in Minimal Propositional Calculus with Implication and Conjunction over a Three Letter Alphabet, Technological University Eindhoven, Memorandum 1975-06

De Jongh, D.H.J. (1968), Investigations on the Intuitionistic Propositional Calculus, *Doctoral Dissertation,* University of Wisconsin

De Jongh, D.H.J. (1970), A Characterization of the Intuitionistic Propositional Calculus, in: A.Kino, J.Myhill, R.E.Vesley eds., *Intuitionism and Proof Theory, Proceedings of the Summer Conference at Buffalo, N.Y., 1968,* pp.103-111

De Jongh, D.H.J. (1980), A Class of Intuitionistic Connectives, in: J. Barwise, H.J. Keisler and K. Kunen, eds., *The Kleene Symposium,* North-Holland Publishing Company, pp. 103-111

Diego, A. (1966), *Sur les Algèbres de Hilbert,* Gauthier-Villars, Paris

Hendriks, L. (1980), Logische Automaten, Beslissen en Bewijzen met de Computer, *Master's Thesis* (in Dutch), Department of Mathematics, University of Amsterdam

Kripke, S.A. (1965), Semantical Analysis of Intuitionistic Logic I, in: J.N.Crossley, M.A.E.Dummett eds., *Formal Systems and Recursive Functions,* Amsterdam, pp. 92-130

Nishimura, I. (1960), On Formulas of one Variable in Intuitionistic Propositional Logic, *Journal of Symbolic Logic*, 25, pp. 327-331

Rasiowa, H. and R.Sikorski (1963), *The Mathematics of Metamathematics*, Warszawa.

Renardel de Lavalette, G.R. (1981), The Interpolation Theorem in Fragments of Logics, *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen, Series A*, 84, pp. 71-86 (also published in *Indagationes Mathematicae* 43, 1981, pp. 71-86)

Renardel de Lavalette, G.R. (1987), Interpolation in Fragments of Intuitionistic Propositional Logic, *Logic Group Preprint Series* No.24, Department of Philosophy, University of Utrecht

Rieger, N.S. (1949), On the Lattice Theory of Brouwerian Propositional Logic, *Acta Fac. Rerum Nat. Univ. Carolinae*, 189, pp. 3-40

Troelstra, A.S. (ed.) (1973), *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, Lecture Notes in Mathematics 344, Springer-Verlag, Berlin, Heidelberg, New York

Van Riemsdijk, H. (1985), Het Genereren van Diagrammen, *Master's Thesis* (in Dutch), Department of Mathematics, University of Amsterdam