The Logic of Preference and Decision Supporting Systems

Dragan Cvetković

MPI–I–94–260 November 1994

Author's Address

Dragan Cvetković
Mathematical Faculty
Studentski Trg 16
11000 Belgrade
Yugoslavia
ecvetkov@yubgef51.bitnet

current address:

Dragan Cvetković
MPI Informatik
Im Stadtwald
66123 Saarbrücken
Germany
dragan@mpi-sb.mpg.de

Publication Notes

This is a translation into English and a slight modification of my thesis presented for the Master of Science Degree at the Mathematical Faculty, University of Belgrade and defended on May 19, 1993.

Acknowledgements

To Aleksandar Kron, Žarko Mijajlović and Milan Božić, members of my Master of Science thesis committee, and to Hans Jürgen Ohlbach for some of his ideas that influenced my work.

Abstract

In this thesis we are exploring some models for von Wright's preference logic. For a given (initial) set of axioms and a set of formulae, some of them valid, some of them problematic (in the sense that it is not always intuitively clear whether they should be valid or not), we investigated some matrix semantics for those formulae including semantics in relevance logics (first degree entailment and RM3), various many–valued (Kleene's, Łukasiewicz's, ...) and/or paraconsistent logics, in Sugihara matrix, and one interpretation for preference relation using modal operators $\Box$ and $\Diamond$. In each case, we also investigated dependence results between various formulae. An opposite problem (i.e. searching for a logic that satisfies given constraints) is also addressed. At the end, a `LISP` program is presented that implements von Wright's logic as a decision supporting system, i.e. that decides for a given set of preferences, what alternatives (world situation) we should choose, according to von Wright's preference logic system.

Keywords

Logic of preference, von Wright, relevance logics, paraconsistent logics, many–valued logics, decision supporting systems, LISP

Contents

1 Introduction and history **3**
- 1.1 Martin's logic of preference . . . 3
- 1.2 Logic of preference Chisholm–Sosa . . . 4
- 1.3 Halldén's logic of preference . . . 5
- 1.4 Von Wright's logic of preference . . . 5
- 1.5 Question of semantics . . . 7
- 1.6 Other approaches to semantics . . . 8
- 1.7 Our intentions . . . 8

2 In search for semantics **9**
- 2.1 System of preferences and E_{fde} . . . 9
  - 2.1.1 Von Wright's preference system . . . 9
  - 2.1.2 Intended interpretation . . . 9
  - 2.1.3 What is valid? . . . 10
  - 2.1.4 What is not valid? . . . 10
  - 2.1.5 New system of axioms . . . 11
  - 2.1.6 Axioms and rules of E_{fde} . . . 11
  - 2.1.7 Dependences between formulae . . . 12
  - 2.1.8 Slight modifications . . . 14
  - 2.1.9 Dependences in new system . . . 15
- 2.2 Interpretation in Sugihara matrix . . . 15
  - 2.2.1 Definitions . . . 15
  - 2.2.2 Why using Sugihara matrix? . . . 16
  - 2.2.3 Negative results . . . 17
- 2.3 Return to E_{fde} . . . 17
  - 2.3.1 Results . . . 18
- 2.4 Paraconsistent and many–valued logics . . . 18
  - 2.4.1 Why paraconsistent logics? . . . 18
  - 2.4.2 Definitions for many–valued logics . . . 19
  - 2.4.3 Testing method . . . 19
  - 2.4.4 System 1 . . . 19
  - 2.4.5 System 2 . . . 20
  - 2.4.6 System 3 . . . 20
  - 2.4.7 System 4 . . . 20
  - 2.4.8 System 5 . . . 21
- 2.5 Results . . . 21
  - 2.5.1 Remarks about the table of results . . . 22
- 2.6 Opposite problem . . . 23
  - 2.6.1 Definition of interpretations . . . 23
  - 2.6.2 Results . . . 24
- 2.7 New interpretations . . . 27
  - 2.7.1 Kleene's 3–valued logic . . . 27

2.7.2 Results . . . 28
2.7.3 Von Wright's extended system . . . 28
2.8 Indifference relation . . . 28
2.8.1 Definitions . . . 28
2.8.2 Weak indifference relation . . . 29
2.8.3 Strong indifference relation . . . 29
2.9 Further investigations . . . 30
2.9.1 Definition of relevance logic RM3 . . . 31
2.9.2 Dependences between formulae in RM3 . . . 32
2.10 Alternative system . . . 33
2.10.1 Definition of the system 1 . . . 34
2.10.2 Another system . . . 35
2.10.3 Dependences between formulae in those systems . . . 36
2.11 Slightly misbehaving . . . 37
2.11.1 Results . . . 38
2.11.2 Another E_{fde}–model . . . 38

3 Logic of preference and modal logics 40
3.1 Definition of interpretations . . . 40
3.1.1 Properties . . . 40
3.2 Modal logic and indifference relation . . . 44

4 Decision supporting systems 46
4.1 Simple example . . . 46
4.2 The making of decision procedure . . . 46
4.2.1 Theoretic foundations . . . 47
4.2.2 Decision procedure . . . 47
4.2.3 Remarks about the procedure . . . 49
4.2.4 Modifications for the extended system . . . 49
4.2.5 Modifications for indifference relation . . . 49
4.3 Implementation . . . 50
4.3.1 Program description . . . 51

A Source program listing 53
A.1 File `variable.lsp` . . . 53
A.2 File `include.lsp` . . . 53
A.3 File `step1.lsp` . . . 57
A.4 File `step2.lsp` . . . 58
A.5 File `step3.lsp` . . . 59
A.6 File `step4.lsp` . . . 60
A.7 File `step5.lsp` . . . 62
A.8 File `step6.lsp` . . . 63
A.9 File `postproc.lsp` . . . 64
A.10 File `main.lsp` . . . 66

Chapter 1

Introduction and history

The whole story about the logic of preference and preference relations is an attempt at the formalization of (rather fuzzy) feeling of preferring something, that is, an attempt of axiomatic explanation why we choose one thing over another. Historically speaking, this relation is already mentioned in Aristotle's work, but first modern attempts have been made in the decade of 1950–1960. But, as von Wright said in [vW72], contrary to the development in deontic logic, where authors mainly agree with basic postulates of the theory, the situation in the logic of preference is quite different. To quote ([vW72, p.141]):

> ... The 'intuitions' of various researchers into this field seem largely at variance with one another. Is the preference relation transitive? Are any two states or things comparable for preference; if one state is preferred to another is, then, the negation of the second preferred to the negation of the first ('contraposition'); can a preference between disjunctions or conjunctions be distributed, and if so, how? These are questions on which there are almost as many divergent opinions as there have been writers on the topic.

The concept of preference is also used in economics, but in a relation to 'utility' function. We can also split preferences into individual and group preferences, but we are going to consider only individual preferences.

In the sequel, we are going to present only the most important systems, and for the more complete presentation the reader is referred to [Hub72].

1.1 Martin's logic of preference

Martin presented this system, which was one of the first, in [Mar63] in 1963. He introduced two relations: X Prfr a, b, t, with intended meaning "Person X prefers a to b in time t" and X Indiff a, b, t as $\neg(X \text{ Prfr } a, b, t) \wedge \neg(X \text{ Prfr } b, a, t)$, where a and b are sentences. The main properties of those relations (slightly simplified) are:

(M1) X Prfr $a, b, t \to a \neq b$

(M2) X Indiff $a, b, t \to X$ Indiff b, a, t

(M3) X Indiff a, a, t

(M4) $(X \text{ Prfr } a, b, t) \wedge (X \text{ Prfr } b, c, t) \to X \text{ Prfr } a, c, t$

(M5) $(X \text{ Indiff } a, b, t) \wedge (X \text{ Indiff } b, c, t) \to X \text{ Indiff } a, c, t$

(M6) $(X \text{ Prfr } a, b, t) \vee (X \text{ Prfr } b, a, t) \vee (X \text{ Indiff } a, b, t)$

(M7) X Prfr $a, b, t \to \neg(X$ Prfr $b, a, t)$

(M8) X Prfr $a, b, t \leftrightarrow X$ Prfr $\neg b, \neg a, t$

(M9) $(X$ Prfr $a, c, t) \vee (X$ Prfr $b, c, t) \to X$ Prfr $(a \vee b), c, t$

(M10) X Prfr $c, (a \vee b), t \to (X$ Prfr $c, a, t) \wedge (X$ Prfr $c, b, t)$

(M11) If a is a theorem, and b a false statement, then X Prfr a, b, t.

(M12) If a is true, and b false, then X Prfr a, b, t.

(M13) If e is a non–false statement and if $c(x, y)$ is a conformance degree of statement x compared to y, then $c(a, e) > c(b, e) \leftrightarrow X$ Prfr a, b, t.

1.2 Logic of preference Chisholm–Sosa

Roderik M. Chisholm and Ernest Sosa introduced their logic of preference in [CS66b] in 1966. Except for a preference relation P, they also introduced the following relations:

(D1) $xSy \leftrightarrow \neg(xPy) \wedge \neg(yPx)$ ('same in intrinsic value as')

(D2) $Ix \leftrightarrow \neg(xP\neg x) \wedge \neg(\neg xPx)$ ('is intrinsically indifferent in value')

(D3) $Nx \leftrightarrow (\exists y)(Iy \wedge xSy)$ ('is intrinsically neutral in value')

(D4) $Gx \leftrightarrow (\exists y)(Iy \wedge xPy)$ ('is intrinsically good')

(D5) $Bx \leftrightarrow (\exists y)(Iy \wedge yPx)$ ('is intrinsically bad')

As for the axioms, their list of axioms is:

(C&S1) $(\forall x)(\forall y)(xPy \to \neg(yPx))$

(C&S2) $(\forall x)(\forall y)(\forall z)(\neg(xPy) \wedge \neg(yPz) \to \neg(xPz))$

(C&S3) $(\forall x)(\forall y)(Ix \wedge Iy \to xSy)$

(C&S4) $(\forall x)[(\forall y)(Iy \to xPy) \to xP\neg x]$

(C&S5) $(\forall x)[(\forall y)(Iy \to yP\neg x) \to xP\neg x]$

In [CS66a], authors slightly modified their logic, and instead of (C&S4) and (C&S5) they introduced axiom:

(C&S4′) $(\forall x)[(\forall y)(Iy \to xPy) \vee (\forall y)(Iy \to yP\neg x) \to xP\neg x]$

Chisholm and Sosa strongly object to von Wright's and Martin's principles $xPy \leftrightarrow (x \wedge \neg y)P(y \wedge \neg x)$ and $xPy \leftrightarrow \neg yP\neg x$, giving examples that are often cited when discussing various principles of preference logic ([CS66a, p.322]):

> Given such principles, we cannot say, as the other systems do, that if p is preferable to q then not–q is preferable to not–p. There being happy Americans (p) is intrinsically preferable to there being stones (q), since the former entails the existence of pleasure and of no displeasure and the latter entails the existence of neither. But, there being no stones (not–q) is not intrinsically preferable to there being no happy Americans (not–p) since neither of these negative states of affairs entails that any pleasure exists or that any displeasure exists.
>
> For similar reasons, we must reject the following formula:
>
> $$pPq \equiv (p\& \sim q)P(q\& \sim p)$$

> which is sometimes taken as axiomatic; indeed we must say both that the left hand expression does not imply the right, and also that the right hand expression does not imply the left. Suppose that p is the state of affairs which is its being false that there are three happy Greeks and that q is the state of affairs which is there being two unhappy Romans; then the expression on the left will be true and the expression on the right will be false (indeed, q and not–p will be preferable to p and not–q). Or suppose that p is the state of affairs which is there being stones and q is that state of affairs which is there being no happy Americans; then the expression on the right will be true, and the expression on the left will be false.

But, to all those (and some other) objections, von Wright said in [vW72, p.148]:

> ... Criticism leveled against it on the ground that it is counterintuitive have failed to convince me.

1.3 Halldén's logic of preference

Sören Halldén defined one preference system in [Hal66] that he connected with modal logics. He used notation xBy for "If we have to choose between x and y, then we should choose x", and xSy for "If we have to choose between x and y, then we can choose x or we can as well choose y".

His axiom system is:

(H1) $\Box x \to x$

(H2) $\Box(x \to y) \to (\Box x \to \Box y)$

(H3) $\Box(x \leftrightarrow y) \to \neg(xBy)$

(H4) $xBy \wedge yBz \to xBz$

(H5) $\Box(x \leftrightarrow y) \to xSy$

(H6) $xSy \to ySx$

(H7) $xSy \wedge ySz \to xSz$

(H8) $xSy \wedge zSu \to (xBz \leftrightarrow yBu)$

1.4 Von Wright's logic of preference

It is generally credited by all authors working into this area that von Wright's contribution was very important for the further developments in the logic of preference. Von Wright published his first work on that subject in 1963 in [vW63]. He says that he will consider preferences between states of affairs because all other preference types (i.e. between objects and between actions) can be reduced to this kind of preferences. For operations on states of affairs, he chooses standard boolean connectives of classical propositional logic as he said in [vW72] on page 143:

> ... States of affairs can be called proposition–like entries. This means that they can be negated and that we can form with them molecular compounds which can be handled in accordance with the laws of propositional logic (PL).

Furthermore, he distinguishes two kind of preferences: *extrinsic* – for which we can state rational reasons why we prefer one state to another and which von Wright does not investigate, and *intrinsic* for which we cannot state rational reasons why we

choose something. Von Wright investigates this kind of preferences. Furthermore, he says that all preferences should be considered relative to the person that chooses and to the moment in time (see also how Martin introduced preference relation in section 1.1 on page 3).

For his preference relation P, von Wright introduces the following axioms:

(W1) $xPy \to \neg(yPx)$

(W2) $xPy \wedge yPz \to xPz$

(W3) $xPy \leftrightarrow (x \wedge \neg y)P(y \wedge \neg x)$

(W4) $(x \vee y)P(z \vee u) \leftrightarrow (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u)$

(W5) $xPy \leftrightarrow (x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$, where α is a new variable.

Von Wright also introduces a (weak) indifference relation xIy as $\neg(xPy) \wedge \neg(yPx)$ and a strong indifference ('value–equality') relation E. We say that xEy if and only if (under no circumstances) neither state $x \wedge \neg y$ is preferred to $y \wedge \neg x$ nor vice versa. For relation I, von Wright does not require any special property to hold, and the reason for introducing E are the following two principles:

(E1) $xPy \wedge xEz \to zPy$

(E2) $xPy \wedge yEz \to xPz$

He gives the following reason for the existence of two different indifference relations (see [vW63, pp.55–57]): As the formula xPy means that x is preferred to y under *all* circumstances, according to the definition of I, xIy does not mean any more under all circumstances, but under *some* circumstances. For indifference under all circumstances, he introduces relation E.

In his another article ([vW72]) published in 1972, von Wright slightly changes the system, or, to cite (on page 142):

> Without abandoning its central tenets, I shall propose some changes which at the same time will make the theory stronger and less complex then it was in its original form.

Basically, von Wright says that preferences are *holistic*, i.e. relative to circumstances, and he introduces two different meanings for notion of *preferring state s to state t under circumstances C_i*:

(D1) We say that s is preferred to t under circumstances C_i if and only if *every* C_i–world that is also a s–world but not a t–world is preferred to *every* C_i–world that is also a t–world but not a s–world.

(D2) We say that s is preferred to t under circumstances C_i if and only if *some* C_i–world that is also a s–world is preferred to *some* C_i–world that is also a t–world, but *no* C_i–world that is also a t–world is preferred to *any* C_i–world that is also a s–world.

However, von Wright does not choose any of those definitions for the intended one ([vW72, p.148]):

> The two definitions are two ways of explicating the notation of a holistic preference. Both ways seem to be reasonable. There is nothing to decide between them on grounds of logic.

He also does not want to address the problem of arguments of preference relation that are tautologies or contradictions. As for the axioms, von Wright drops axiom (W2) and instead studies the following one (which he calls 'Principle of Value Comparability')

$$\text{(W2}') \quad xPy \to xPz \vee zPy$$

that he uses to prove (W2). Contraposition of this axiom gives axiom (C&S2) of system Chisholm–Sosa. If we define indifference relation in this system as $\neg(xPy) \wedge \neg(yPx)$, we can readily see that this relation satisfies principles (E1) and (E2) (as well as the usual condition that it is an equivalence relation).

Indeed: $xPy \wedge xEz \to xPy \wedge \neg(xPz) \wedge \neg(zPx) \to (xPz \vee zPy) \wedge \neg(xPz) \wedge \neg(zPx) \to zPy \wedge \neg(xPz) \wedge \neg(zPx) \to zPy$. That proves (E1), analogously for (E2).

In that sense, the modified system is really a simplification of the original one, because there are no (and also no need for) two different indifference relations. We will mainly work with the original von Wright's system. We can also notice that Halldén's axiom (H8) (see section 1.3) is also derivable in this extended system.

Indeed: Let us suppose that xIy and zIu. That means that $\neg(xPy)$, $\neg(yPx)$, $\neg(zPu)$ and $\neg(uPz)$. Using (W2′), we have $xPz \to xPy \vee yPz$, and hence $xPz \to yPz$. Furthermore, $yPz \to yPu \vee uPz$, and hence $yPz \to yPu$, which together give $xPz \to yPu$. Similar in other direction. That proves the axiom (H8) of Halldén's system.

1.5 Question of semantics

Rescher addressed semantics of preferences in [Res67]. He mainly used 'utility–function' approach.

For a set of worlds $w_1, w_2, \ldots, w_n$ he introduced measure $\#$, which he calls 'index of merit', as a function from set of worlds to real numbers. Using this measure, he defined an interpretation $P^{\#}$ in the following way:

$$\alpha P^{\#} \beta \text{ if } \#(\alpha) > \#(\beta).$$

He also introduced another measure $\star(\alpha) \stackrel{\text{def}}{=} \#(\alpha) - \#(\neg\alpha)$ and, accordingly, another interpretation:

$$\alpha P^{\star} \beta \text{ if } \star(\alpha) > \star(\beta).$$

For von Wright's preference relation, he introduced *von Wright's semantics* in the following way: Let, as before, $w_1, \ldots w_n$ be a set of worlds (states of affairs in von Wright's terminology), and suppose that there exists a preferential order $>$ together with the indifference $\simeq$ between worlds. We say that $\alpha P^{w} \beta$ if

$$(\forall\gamma)(\forall w_1)(\forall w_2)(w_1 \models (\alpha \wedge \neg\beta \wedge \gamma) \wedge w_2 \models (\neg\alpha \wedge \beta \wedge \gamma) \to w_1 > w_2)$$

where γ is independent of α and β.

With the help of those semantics, Rescher investigated some of the more popular axiomatizations and principles of logics of preference. It is interesting to note that none of those semantics supports Chisholm's and Sosa's standpoint that principles (W3) and (M8) should not hold. Moreover, in semantics P^{w} all von Wright's axioms hold, except (W4) from right to left.

1.6 Other approaches to semantics

The question of semantics was also addressed by Sven Danielsson in [Dan68]. In this work, on pages 19–21, he gave one model for preference relation.

First, he introduced PC–valuation t, that is the usual valuation for propositional formulae, then M–valuation (modal logic valuation) as an ordered pair (t, m), where t is (above–defined) PC–valuation, and m is a mapping from the set of valuations t into its power set, so that $t'(\Diamond p) = 1$ if and only if $t''(p) = 1$ for some $t'' \in m(t')$.

Finally, he introduced i–valuation: i–valuation, relative to M–valuation (t, m), is an ordered triple (OM, om, R), where:

- OM is a non–empty set;
- om is a function that to every propositional formula ψ so that $t(\Diamond\psi) = 1$ assigns a non–empty subset of OM so that $om(q) = om(r)$ if $t(\Box(q \leftrightarrow r)) = 1$, and
- R is a relation on OM with the following properties:
  1. $(\forall x \in OM)(\forall y \in OM)(xRy \vee yRx)$
  2. $(\forall x \in OM)(\forall y \in OM)(\forall z \in OM)(xRy \wedge yRz \rightarrow xRz)$
  3. $x \in om(p \vee q) \rightarrow (\exists y)(\exists z)(y, z \in om(p) \cup om(q) \wedge yRx \wedge xRz)$

Of course, we write xPy for $xRy \wedge \neg yRx$. In this model, xPy is valid if x is 'intrinsically better' then y.

Except for this valuation, Danielsson also defined A–valuation which he used for investigating properties of his A_i and B_i operators ($i = 1, 2, \ldots, 6$).

1.7 Our intentions

In this work, we are trying to find some appropriate semantics for von Wright's logic system, but not in a way Rescher did, but using the following approach:

We assume that states of affairs (arguments of preference relation) behave according to the laws of classical propositional logic (we will call it *inner logic*), as von Wright remarked in [vW63], and for *outer logic*, we employ different logics: first, we use first degree entailment E_{fde} (relevance logic described in [AB75]), and after that various other relevance, paraconsistent and many–valued logics, Sugihara matrix etc. A criteria for choosing outer logic is that it has a matrix (tabular) interpretation, as we are looking for matrix interpretation for preference relation.

In all those logics, unless otherwise stated, we will look for the interpretation f of the following type:

$$f(xPy) = \begin{cases} a, & \text{if } x = 0 \text{ and } y = 0; \\ b, & \text{if } x = 0 \text{ and } y = 1; \\ c, & \text{if } x = 1 \text{ and } y = 0; \\ d, & \text{if } x = 1 \text{ and } y = 1. \end{cases}$$

and we try, using computer, to find values for a, b, c and d, so that this interpretation retains the most of the preference relation properties.

Except for this approach, we also tried to find connections between logics of preference and modal propositional logics, and we investigated properties of preference relations defined using modal operators.

Chapter 2

In search for semantics

2.1 System of preferences and E_{fde}

2.1.1 Von Wright's preference system

If we denote with P preference relation, we have the following system of axioms for von Wright's logic of preference (see [vW63] or [KM75]):

(W1) $xPy \to \neg(yPx)$

(W2) $xPy \wedge yPz \to xPz$

(W3) $xPy \leftrightarrow (x \wedge \neg y)P(y \wedge \neg x)$

(W4) $(x \vee y)P(z \vee u) \leftrightarrow (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u)$

(W5) $xPy \leftrightarrow (x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$, where α is a new variable.

2.1.2 Intended interpretation

The first interpretation we are going to consider is a *natural interpretation* i.e. a kind of interpretation that assures us that if x is true and y false, then xPy is true and yPx is false. So, one way to embed logic system (W1)–(W5) into the first degree entailment E_{fde} is the following:

$$f(xPy) = \begin{cases} 1, & \text{if } x = 1 \text{ and } y = 0; \\ 4, & \text{if } x = 0 \text{ and } y = 1; \\ 2, & \text{otherwise.} \end{cases}$$

where $\varphi \leftrightarrow \psi$ is assumed to mean $\varphi \rightleftarrows \psi$, i.e. $\varphi \to \psi$ and $\psi \to \varphi$ (see [AB75]). We are using characteristic matrices for E_{fde} given by T. J. Smiley ([AB75, pp.162–163]):

$\wedge$	1	2	3	4
1	1	2	3	4
2	2	2	4	4
3	3	4	3	4
4	4	4	4	4

$\vee$	1	2	3	4
1	1	1	1	1
2	1	2	1	2
3	1	1	3	3
4	1	2	3	4

$\to$	1	2	3	4
1	1	4	4	4
2	1	1	4	4
3	1	4	1	4
4	1	1	1	1

	$\neg$
1	4
2	2
3	3
4	1

where 1 is designated true, and 4 designated false value.

Testing above axioms in this interpretation, we obtained the following results:

- Axioms (W1), (W2) and (W3) are valid;
- Axioms (W4) and (W5) are only valid from right to left.

2.1.3 What is valid?

With the help of the computer program written in `LISP` and using above tables for E_{fde}, we obtained validness of the following formulae:

(1) $xPy \to \neg(yPx)$

(2) $xPy \wedge yPz \to xPz$

(3) $xPy \to (x \wedge \neg y)P(y \wedge \neg x)$

(4) $(x \wedge \neg y)P(y \wedge \neg x) \to xPy$

(5) $(x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \to (x \vee y)P(z \vee u)$

(6) $(x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha) \to xPy$, where α is a new variable.

(7) $xPy \to (x \wedge \alpha)P(y \wedge \alpha) \vee (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$, where α is a new variable.

(8) $xP(y \vee z) \to xPy \wedge xPz$

(9) $xPy \wedge xPz \to xP(y \vee z)$

(10) $(x \vee y)Pz \to xPz \vee yPz$

(11) $xPz \vee yPz \to (x \vee y)Pz$

(12) $xPz \wedge yPz \to (x \vee y)Pz$

(13) $(x \wedge y)Pz \to xPz \wedge yPz$

(14) $xPz \wedge yPz \to (x \wedge y)Pz$

(15) $(x \wedge y)Pz \to xPz \vee yPz$

(16) $xPy \to \neg yP\neg x$

(17) $\neg yP\neg x \to xPy$

(18) $xP(y \wedge z) \to xPy \vee xPz$

(19) $xPy \vee xPz \to xP(y \wedge z)$

(20) $xPy \wedge xPz \to xP(y \wedge z)$

(21) $\neg(yPx) \to xPy$

2.1.4 What is not valid?

The following formulae are invalid in the above–mentioned interpretation:

(o2) Converse of (2): $xPz \to xPy \wedge yPz$.

(o5) Converse of (5): $(x \vee y)P(z \vee u) \to (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u)$

(o6) Converse of (6): $xPy \to (x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$, where α is a new variable.

(o7) Converse of (7): $(x \wedge \alpha)P(y \wedge \alpha) \vee (x \wedge \neg\alpha)P(y \wedge \neg\alpha) \to xPy$, where α is a new variable.

(o12) Converse of (12): $(x \vee y)Pz \to xPz \wedge yPz$.

(o15) Converse of (15): $xPz \vee yPz \to (x \wedge y)Pz$.

(o20) Converse of (20): $xP(y \wedge z) \to xPy \wedge xPz$.

2.1.5 New system of axioms

Considering the following set of axioms:

(S1) $xPy \rightleftarrows \neg(yPx)$ (formulae (1), (21))
(S2) $xPy \wedge yPz \to xPz$ (formula (2))
(S3) $xPy \rightleftarrows \neg yP\neg x$ (formulae (16), (17))
(S4) $(x \vee y)Pz \rightleftarrows xPz \vee yPz$ (formulae (10), (11))

we are going to prove the following theorem:

Theorem 2.1.1 *From axioms* (S1)–(S4)*, using axioms and rules of* E_{fde}*, we can derive all formulae* (1)–(21)*.*

2.1.6 Axioms and rules of E_{fde}

To prove theorem 2.1.1, we need axioms and rules for a logical system we are working in. We will use axiomatization of E_{fde} from [AB75, p.158]. Axioms are:

(E1) $A \wedge B \to A$
(E2) $A \wedge B \to B$
(E3) $A \to A \vee B$
(E4) $B \to A \vee B$
(E5) $A \wedge (B \vee C) \to (A \wedge B) \vee C$
(E6) $A \to \neg\neg A$
(E7) $\neg\neg A \to A$

and inference rules are:

(R1) $\dfrac{\vdash A \to B \quad \vdash B \to C}{\vdash A \to C}$
(R2) $\dfrac{\vdash A \to B \quad \vdash A \to C}{\vdash A \to B \wedge C}$
(R3) $\dfrac{\vdash A \to C \quad \vdash B \to C}{\vdash A \vee B \to C}$
(R4) $\dfrac{\vdash A \to B}{\vdash \neg B \to \neg A}$

We will also use the following lemma from [AB75, p.159]:

Lemma 2.1.1 *The following equivalences are provable in* E_{fde}*:*

(a) $A \wedge B \rightleftarrows B \wedge A$ *(commutativity of* $\wedge$*)*
(b) $A \vee B \rightleftarrows B \vee A$ *(commutativity of* $\vee$*)*
(c) $A \wedge (B \wedge C) \rightleftarrows (A \wedge B) \wedge C$ *(associativity of* $\wedge$*)*
(d) $A \vee (B \vee C) \rightleftarrows (A \vee B) \vee C$ *(associativity of* $\vee$*)*
(e) $A \wedge (B \vee C) \rightleftarrows (A \wedge B) \vee (A \wedge C)$ *(distributivity of* $\wedge$ *over* $\vee$*)*
(f) $A \vee (B \wedge C) \rightleftarrows (A \vee B) \wedge (A \vee C)$ *(distributivity of* $\vee$ *over* $\wedge$*)*
(g) $\neg(A \wedge B) \rightleftarrows \neg A \vee \neg B$ *(de Morgan's law for* $\wedge$*)*
(h) $\neg(A \vee B) \rightleftarrows \neg A \wedge \neg B$ *(de Morgan's law for* $\vee$*)*
(i) *If* $\varphi_1 \rightleftarrows \varphi_2$ *then* $\psi(\ldots,\varphi_1,\ldots) \rightleftarrows \psi(\ldots,\varphi_2,\ldots)$ *(uniform substitution)*

2.1.7 Dependences between formulae

In sequel, we will write $(9), (13) \vdash (6)$ for the statement that formula (6) is derivable from formulae (9) and (13).

Lemma 2.1.2 *The following holds:*

- **(a)** $(12), (9), (4) \vdash (5)$*;*
- **(b)** $(6), (1), (21) \vdash (7)$*;*
- **(c)** $(13), (9) \vdash (6)$*;*
- **(d)** $(11) \vdash (12)$*;*
- **(e)** $(19) \vdash (20)$*;*
- **(f)** $(13) \vdash (15)$*;*
- **(g)** $(10), (13), (16), (17), (18) \vdash (4)$.

Proof:

(a)

$(x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \rightarrow (x \wedge \neg(z \vee u))P(z \wedge \neg(x \vee y)) \wedge (x \wedge \neg(z \vee u))P(u \wedge \neg(x \vee y)) \wedge (y \wedge \neg(z \vee u))P(z \wedge \neg(x \vee y)) \wedge (y \wedge \neg(z \vee u))P(u \wedge \neg(x \vee y))$ (de Morgan's laws)

$(x \wedge \neg(z \vee u))P(z \wedge \neg(x \vee y)) \wedge (x \wedge \neg(z \vee u))P(u \wedge \neg(x \vee y)) \wedge (y \wedge \neg(z \vee u))P(z \wedge \neg(x \vee y)) \wedge (y \wedge \neg(z \vee u))P(u \wedge \neg(x \vee y)) \rightarrow ((x \vee y) \wedge \neg(z \vee u))P(z \wedge \neg(x \vee y)) \wedge ((x \vee y) \wedge \neg(z \vee u))P(u \wedge \neg(x \vee y))$ (formula (12), distributivity in E_{fde})

$((x \vee y) \wedge \neg(z \vee u))P(z \wedge \neg(x \vee y)) \wedge ((x \vee y) \wedge \neg(z \vee u))P(u \wedge \neg(x \vee y)) \rightarrow ((x \vee y) \wedge \neg(z \vee u))P((z \vee u) \wedge \neg(x \vee y))$ (formula (9))

$((x \vee y) \wedge \neg(z \vee u))P((z \vee u) \wedge \neg(x \vee y)) \rightarrow (x \vee y)P(z \vee u)$ (formula (4))

$(x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \rightarrow (x \vee y)P(z \vee u)$ (rule (R1) several times)

(b)

$(y \wedge \alpha)P(x \wedge \alpha) \wedge (y \wedge \neg\alpha)P(x \wedge \neg\alpha) \rightarrow yPx$ (formula (6))

$\neg(yPx) \rightarrow \neg((y \wedge \alpha)P(x \wedge \alpha)) \vee \neg((y \wedge \neg\alpha)P(x \wedge \neg\alpha))$ (rule (R4) and de Morgan's laws)

$xPy \rightarrow \neg(yPx)$ (formula (1))

$\neg((y \wedge \alpha)P(x \wedge \alpha)) \vee \neg((y \wedge \neg\alpha)P(x \wedge \neg\alpha)) \rightarrow (x \wedge \alpha)P(y \wedge \alpha) \vee (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$ (formulae (1) and (21), substitution)

$xPy \rightarrow (x \wedge \alpha)P(y \wedge \alpha) \vee (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$ (rule (R1) on previous 2 formulae)

(c)

$(x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha) \rightarrow xP(y \wedge \alpha) \wedge \alpha P(y \wedge \alpha) \wedge xP(y \wedge \neg\alpha) \wedge \neg\alpha P(y \wedge \neg\alpha)$ (formula (13))

$xP(y \wedge \alpha) \wedge \alpha P(y \wedge \alpha) \wedge xP(y \wedge \neg\alpha) \wedge \neg\alpha P(y \wedge \neg\alpha) \rightarrow xP(y \wedge (\alpha \vee \neg\alpha)) \wedge \alpha P(y \wedge \alpha) \wedge \neg\alpha P(y \wedge \neg\alpha)$ (formula (9))

$xP(y \wedge (\alpha \vee \neg\alpha)) \wedge \alpha P(y \wedge \alpha) \wedge \neg\alpha P(y \wedge \neg\alpha) \rightarrow xPy \wedge \alpha P(y \wedge \alpha) \wedge \neg\alpha P(y \wedge \neg\alpha)$ ($\alpha \vee \neg\alpha$ is a tautology of propositional logic)

$xPy \wedge \alpha P(y \wedge \alpha) \wedge \neg\alpha P(y \wedge \neg\alpha) \rightarrow xPy$ (axiom (E1))

$(x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha) \rightarrow xPy$ (rule (R1) several times)

(d)

$xPz \wedge yPz \rightarrow xPz$ (axiom (E1))

$xPz \rightarrow xPz \vee yPz$ (axiom (E3))

$xPz \vee yPz \rightarrow (x \vee y)Pz$ (formula (11))

$xPz \wedge yPz \rightarrow (x \vee y)Pz$ (rule (R1) applied several times)

(e)

$xPy \wedge xPz \to xPy$ (axiom (E1))
$xPy \to xPy \vee xPz$ (axiom (E3))
$xPy \vee xPz \to xP(y \wedge z)$ (formula (19))
$xPy \wedge xPz \to xP(y \wedge z)$ (rule (R1) applied several times)

(f)

$(x \wedge y)Pz \to xPz \wedge yPz$ (formula (13))
$xPz \wedge yPz \to xPz$ (axiom (E1))
$xPz \to xPz \vee yPz$ (axiom (E3))
$(x \wedge y)Pz \to xPz \vee yPz$ (rule (R1) applied several times)

(g)

$(x \wedge \neg y)P(y \wedge \neg x) \to xP(y \wedge \neg x) \wedge \neg yP(y \wedge \neg x)$ (formula (13))
$xP(y \wedge \neg x) \wedge \neg yP(y \wedge \neg x) \to (xPy \vee xP\neg x) \wedge (\neg yPy \vee \neg yP\neg x)$ (formula (18))
$\neg yP\neg x \rightleftarrows xPy$ (formulae (16), (17))
$(xPy \vee xP\neg x) \wedge (\neg yPy \vee \neg yP\neg x) \to (xPy \vee xP\neg x) \wedge (\neg yPy \vee xPy)$ (substitution)
$(xPy \vee xP\neg x) \wedge (\neg yPy \vee xPy) \to xPy \wedge (xP\neg x \vee \neg yPy)$ (distributivity)
$xPy \wedge (xP\neg x \vee \neg yPy) \to xPy$ (axiom (E1))
$(x \wedge \neg y)P(y \wedge \neg x) \to xPy$ (rule (R1) several times)

□

So, we derived formulae (4), (5), (6), (7), (12), (15) and (20). Since formulae (1) and (21) are in fact axiom (S1), (2) is axiom (S2), (16) and (17) are axiom (S3), and (10) and (11) are axiom (S4), we only need to derive formulae (3), (8), (9), (13), (14), (18) and (19).

Lemma 2.1.3 *The following holds:*

(a) $(1), (18), (19), (21) \vdash (13)$*;*
(b) $(1), (18), (19), (21) \vdash (14)$.

Proof:

(a)

$(x \wedge y)Pz \to \neg(zP(x \wedge y))$ (formula (1))
$\neg(zP(x \wedge y)) \to \neg(zPx \vee zPy)$ (formula (18), (19), substitution)
$\neg(zPx \vee zPy) \to \neg(zPx) \wedge \neg(zPy)$ (de Morgan's laws)
$\neg(zPx) \wedge \neg(zPy) \to xPz \wedge yPz$ (formulae (18), (19))
$(x \wedge y)Pz \to xPz \wedge yPz$ (rule (R1) several times)

(b)

$xPz \wedge yPz \to \neg(zPx) \wedge \neg(zPy)$ (formula (1))
$\neg(zPx) \wedge \neg(zPy) \to \neg(zPx \vee zPy)$ (de Morgan's laws)
$\neg(zPx \vee zPy) \to \neg(zP(x \wedge y))$ (formulae (18), (19), substitution)
$\neg(zP(x \wedge y)) \to (x \wedge y)Pz$ (formula (21))
$xPz \wedge yPz \to (x \wedge y)Pz$ (rule (R1) several times)

□

Lemma 2.1.4 *The following holds:*

(a) $(1), (10), (11), (21) \vdash (9)$*;*
(b) $(1), (10), (21) \vdash (8)$*;*
(c) $(10), (16), (17) \vdash (18)$*;*
(d) $(11), (16), (17) \vdash (19)$*;*
(e) $(14), (16), (17), (19) \vdash (3)$.

Proof:

(a)

$xPy \wedge xPz \to \neg(yPx) \wedge \neg(zPx)$ (formulae (1), (21), substitution)
$\neg(yPx) \wedge \neg(zPx) \to \neg(yPx \vee zPx)$ (de Morgan's laws)
$\neg(yPx \vee zPx) \to \neg((y \vee z)Px)$ (formulae (10), (11), substitution)
$\neg((y \vee z)Px) \to xP(y \vee z)$ (formula (21))
$xPy \wedge xPz \to xP(y \vee z)$ (rule (R1) applied several times)

(b)

$xP(y \vee z) \to \neg((y \vee z)Px)$ (formula (1))
$\neg((y \vee z)Px) \to \neg(yPx \vee zPx)$ (formula (10) and rule (R4))
$\neg(yPx \vee zPx) \to \neg(yPx) \wedge \neg(zPx)$ (de Morgan's laws)
$\neg(yPx) \wedge \neg(zPx) \to xPy \wedge xPz$ (formulae (1), (21), substitution)
$xP(y \vee z) \to xPy \wedge xPz$ (rule (R1) several times)

(c)

$xP(y \wedge z) \to \neg(y \wedge z)P\neg x$ (formula (16))
$\neg(y \wedge z)P\neg x \to (\neg y \vee \neg z)P\neg x$ (de Morgan's laws)
$(\neg y \vee \neg z)P\neg x \to \neg yP\neg x \vee \neg zP\neg x$ (formula (10))
$\neg yP\neg x \vee \neg zP\neg x \to xPy \vee xPz$ (formula (17))
$xP(y \wedge z) \to xPy \vee xPz$ (rule (R1) several times)

(d)

$xPy \vee xPz \to \neg yP\neg x \vee \neg zP\neg x$ (formula (16)
$\neg yP\neg x \vee \neg zP\neg x \to (\neg y \vee \neg z)P\neg x$ (formula (11))
$(\neg y \vee \neg z)P\neg x \to \neg(y \wedge z)P\neg x$ (de Morgan's laws)
$\neg(y \wedge z)P\neg x \to xP(y \wedge z)$ (formula (17))
$xPy \vee xPz \to xP(y \wedge z)$ (rule (R1) several times)

(e)

$xPy \to xPy \vee (xP\neg x \wedge \neg yPy)$ (axiom (E3))
$xPy \vee (xP\neg x \wedge \neg yPy) \to (xPy \vee \neg xPx) \wedge (xPy \vee \neg yPy))$ (distributivity)
$(xPy \vee \neg xPx) \wedge (xPy \vee \neg yPy) \to (xPy \vee xP\neg x) \wedge (\neg yPy \vee \neg yP\neg x)$ (formulae (16), (17))
$(xPy \vee xP\neg x) \wedge (\neg yPy \vee \neg yP\neg x) \to xP(y \wedge \neg x) \wedge \neg yP(y \wedge \neg x)$ (formula (19))
$xP(y \wedge \neg x) \wedge \neg yP(y \wedge \neg x) \to (x \wedge \neg y)P(y \wedge \neg x)$ (formula (14))
$xPy \to (x \wedge \neg y)P(y \wedge \neg x)$ (rule (R1) several times)

□

Proof of theorem 2.1.1: Follows directly from lemmas 2.1.2–2.1.3, since all formulae are derived from (S1)–(S4) □

2.1.8 Slight modifications

The above–presented system is in a way too weak: for instance, the axiom (S3), i.e. Martin's axiom (M8), is problematic and prone to criticism (see [CS66a],[CS66b]). To avoid this, we will consider the following interpretation that still belongs to the above–mentioned class of natural interpretations:

$$f(xPy) = \begin{cases} 1, & \text{if } x = 1 \text{ and } y = 0; \\ 4, & \text{if } x = 0 \text{ and } y = 1; \\ 2, & \text{if } x = 1 \text{ and } y = 1; \\ 3, & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

Slightly modifying the already mentioned `LISP` program, we were able to show that the following formulae are not true in this interpretation:

(3) $xPy \to (x \wedge \neg y)P(y \wedge \neg x)$

(4) $(x \wedge \neg y)P(y \wedge \neg x) \to xPy$

(5) $(x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \to (x \vee y)P(z \vee u)$

(16) $xPy \to \neg yP\neg x$

(17) $\neg yP\neg x \to xPy$

and all others from the list (1)–(21) are valid.

2.1.9 Dependences in new system

Looking at the proofs of lemmas 2.1.2–2.1.3, we can readily see that the following theorem is true:

Theorem 2.1.2 *In the new system, the following dependences hold:*

(a) $(6), (1), (21) \vdash (7)$*;*

(b) $(13), (9) \vdash (6)$*;*

(c) $(1), (10), (11), (21) \vdash (9)$*;*

(d) $(11) \vdash (12)$*;*

(e) $(19) \vdash (20)$*;*

(f) $(13) \vdash (15)$*;*

(g) $(1), (10), (21) \vdash (8)$*;*

(h) $(1), (18), (19), (21) \vdash (13)$*;*

(i) $(1), (18), (19), (21) \vdash (14)$*.*

that is the list of axioms for this system could be:

(S^*1) $xPy \rightleftarrows \neg(yPx)$ (formulae (1), (21))

(S^*2) $xPy \wedge yPz \to xPz$ (formula (2))

(S^*3) $xP(y \wedge z) \rightleftarrows xPy \vee xPz$ (formulae (18), (19))

(S^*4) $(x \vee y)Pz \rightleftarrows xPz \vee yPz$ (formulae (10), (11))

because all other formulae can be derived from those.

2.2 Interpretation in Sugihara matrix

2.2.1 Definitions

Following the work of M. Tokarz ([Tok80]), we will define *Sugihara matrix* as an ordered 6–tuple $S_I = (I, -, \wedge, \vee, \to, D)$, where (carrier set) I is a chain with respect to some (fixed) ordering $\leq$, connective $-$ is a unary, bijective involution with the property that $x \leq y$ implies $-y \leq -x$, connectives $\wedge$ and $\vee$ are defined as a minimum and a maximum of two elements with respect to the ordering $\leq$, connective $\to$ is defined as

$$x \to y = \begin{cases} -x \vee y\ , & \text{if } x \leq y; \\ -x \wedge y\ , & \text{if } x > y. \end{cases}$$

and $D = \{a \in I \mid -a \leq a\}$ is a set of non–negative elements of I.

We usually choose $I = \{-n, \ldots, -1, 1, \ldots, n\}$, and in that case[1] we will denote Sugihara matrix with S_n. If it contains 0, we will denote it by S_n^0. For given matrix S_I, we define a S_I–interpretation $f: Var \mapsto I$ on propositional variables which extends in natural way to formulae in the following way:

[1] Sugihara matrix which does not contain 0 is called *normal* Sugihara matrix.

1^0 If B is a propositional variable, then $f(B) \in I$;
2^0 $f(C \wedge D) = \min(f(C), f(D))$;
3^0 $f(C \vee D) = \max(f(C), f(D))$;
4^0 $f(\neg C) = -f(C)$;
5^0 $f(C \to D) = f(C) \to f(D)$;

We will say that formula φ is *valid in S_I–interpretation f* if $f(\varphi) \in D$, *S_I–valid* if it is valid in all S_I–interpretations and, finally, *valid* if it is S_I–valid for all S_I (see [AB75], pages 400–401).

One of the well–known results concerning Sugihara matrices is the connection between relevance logic RM and Sugihara matrix ([AB75]).

2.2.2 Why using Sugihara matrix?

All previous methods of interpretation in E_{fde} have the same shortcoming, i.e. rather problematic formulae (20) and (21) are valid there. So we will try to circumvent this problem. First, let us consider the formula (21), that is

$$\neg(yPx) \to xPy$$

and let us try to find what interpretation we need to dispose of its validity.

Theorem 2.2.1 *The condition for invalidness of the formula* (21) *and for validness of the formula* (1) *at the same time in some Sugihara matrix is*

$$f(xPy) + f(yPx) < 0.$$

Proof: Let us denote $f(xPy) = p$ and $f(yPx) = q$, for some $p, q \in I$. Then we have:

$$f(xPy \to \neg(yPx)) = \begin{cases} \max(-p, -q), & \text{if } p \le -q; \\ \min(-p, -q), & \text{if } p > -q. \end{cases} = \begin{cases} -\min(p, q), & p \le -q; \\ -\max(p, q), & \text{otherwise.} \end{cases}$$

and

$$f(\neg(yPx) \to xPy) = \begin{cases} \max(p, q), & \text{if } p \ge -q; \\ \min(p, q), & \text{if } p < -q. \end{cases}$$

so (1) will be true and (21) false if for all x and y: $f(xPy \to \neg(yPx)) \ge 0$ and $f(\neg(yPx) \to xPy) < 0$. We have the following cases:

(i) $p = -q$. In that case, we must have $\min(p, q) \le 0$ and $\max(p, q) < 0$, and that is impossible since p and q have opposite signs.

(ii) $p > -q$. In that case, we must have $\max(p, q) \le 0$ and $\max(p, q) < 0$, and that implies that both p and q are negative, which is impossible since $p + q > 0$.

(iii) $p < -q$. In that case, we must have $\min(p, q) \le 0$ and $\min(p, q) < 0$ — no contradiction here. This condition is equivalent to $p + q < 0$. □

So, with respect to this theorem, we made another slight modification of our `LISP` program, and for interpretation of the preference relation P, we have chosen the following (fairly arbitrary) one:

$$f(xPy) = \begin{cases} -\max(|x|, y)\ , & \text{if } x \le y; \\ \min(x, |y|)\ , & \text{if } x > y. \end{cases}$$

which satisfies the condition of the theorem.

We obtained the following results for Sugihara matrix S_2 (we will use notation (1)–(21) from page 10):

- **(a)** Valid formulae: (1), (2), (4), (5), (9), (10), (11), (12), (13), (14), (15), (18) and (20);
- **(b)** Invalid formulae: (3), (6), (7), (8), (16), (17), (19), (21), as well as all formulae invalid in the previous interpretation.

2.2.3 Negative results

As we could see, the formula (20) is still valid although it is intuitively false[2]. So we tried to find an interpretation in Sugihara matrix in which (20) is invalid. Unfortunately, it is not possible because of the following theorem:

Theorem 2.2.2 *In all Sugihara matrices S_I, no matter what interpretation f is, formula*

$$xPy \wedge xPz \rightarrow xP(y \wedge z) \qquad (*)$$

is valid.

Proof: Let as before, $f(xPy) = p$, $f(xPz) = q$, where $p, q \in I$. As I is a chain and $x \wedge y$ is defined as $\min(x, y)$, we have that $\min(y, z) \in \{y, z\}$ and, hence, $f(xP(y \wedge z)) \in \{p, q\}$. That means that $f(xPy \wedge xPz) = \min(p, q)$ and $f(xP(y \wedge z)) \in \{p, q\}$ and, since $\min(p, q) \leq p$ and $\min(p, q) \leq q$, we have:

$$f((*)) = \begin{cases} \max(-\min(p, q), p) \ , & \text{if } \min(y, z) = y; \\ \max(-\min(p, q), q) \ , & \text{otherwise.} \end{cases}$$

Suppose that $p \geq q$ (otherwise we have symmetrical reasoning). In that case $\min(p, q) = q$ and hence

$$f((*)) = \begin{cases} \max(-q, p) \ , & \text{if } y \leq z; \\ \max(-q, q) \ , & \text{if } y > z. \end{cases}$$

If $y > z$, then $f((*)) \geq 0$, since at least one from $-q$, q must be ≥ 0.
If $y \leq z$, we have the following two cases:

(a) $p \geq 0$. In that case $f((*)) = \max(-q, p) \geq p \geq 0$.

(b) $p < 0$. In that case also $q < 0$, so $f((*)) = \max(-q, p) \geq -q > 0$.

So, $f((*))$ is always non–negative, and hence formula $(*)$ is valid. □

Corollary 2.2.1 *The algebraic structure in which* (20) *is invalid is either not a chain, or a conjunction is not defined as a minimum of two elements.*

2.3 Return to E_{fde}

So, we have seen what can be done using Sugihara matrices, and what can be done if we are looking for "natural interpretations" for preference relation[3] in E_{fde}. But, the problem with Sugihara matrices is that we do not know what its background logic is[4], so we can not prove lemmas and theorems of dependences as we did in E_{fde} in previous sections. So, we returned to E_{fde} and wrote another `LISP` program which, using 'brute force' method, searches for 'the best' interpretation for preference relation P in E_{fde}. We will understand the term 'the best' as the one in which as many acceptable formulae as possible are valid and as few non–acceptable formulae (i.e. of (16), (17), (20) and (21)) as possible are valid.

[2] One example is: if x is interpreted as "to win \$100", y as "to win \$70", and z as "to win \$80", we (usually) do have xPy and xPz but not $xP(y \wedge z)$, since $y \wedge z$ means "to win \$150".

[3] Natural in the sense that if $x = 1$ (i.e. true) and $y = 0$ (i.e. false) then xPy should be true.

[4] Not strictly true, because we can apply a result from [AB75] and use relevance logic RM.

2.3.1 Results

We suppose that our interpretation for P is as follows (for $a, b, c, d \in \{1, 2, 3, 4\}$):

$$f(xPy) = \begin{cases} a, & \text{if } x = 0 \text{ and } y = 0; \\ b, & \text{if } x = 0 \text{ and } y = 1; \\ c, & \text{if } x = 1 \text{ and } y = 0; \\ d, & \text{if } x = 1 \text{ and } y = 1. \end{cases}$$

The program checked out all von Wright's axioms, all formulae (1)–(21), as well as all hopefully invalid formulae in *all* interpretations (there are $4^4 = 256$), and found out the following results:

(a) If $(a, b, c, d) \in \{(2, 4, 4, 4), (3, 4, 3, 4), (3, 4, 4, 4)\}$, then formulae (16), (17) and (21) are invalid.

(b) In all under **(a)** mentioned interpretations, formulae (4) and (5) are invalid, and in the first and the third one, formulae (11) and (13) are also not valid.

(c) Formula (20) is valid in all interpretations!

(d) In all under **(a)** mentioned interpretations, valid von Wright's axioms are (W1), (W2) and (W5), whereas axioms (W3) and (W4) are only valid from left to right (that is, opposite from what we had before!).

(e) In the second under **(a)** mentioned interpretation, as a consequence we have $xPz \vee yPz \rightleftarrows xPz \wedge yPz$, (formulae (12) and (o12) give $xPz \wedge yPz \rightleftarrows (x \vee y)Pz$, and formulae (10) i (11) give $(x \vee y)Pz \rightleftarrows xPz \vee yPz$), so we can say that this interpretation is also unacceptable.

By the way, the fact mentioned under **(c)** can also be proved:

Theorem 2.3.1 *In every matrix interpretation of preference relation P of the above–mentioned type where logical operations between preferences are defined as in E_{fde}, and operations in preference relation are defined as in classical logic, formula* (20) *is valid.*

Proof: In the same way the theorem 2.2.2 from page 17 was proved since inner connectives are defined in a classical way, and so $y \wedge z \in \{y, z\}$. □

2.4 Paraconsistent and many–valued logics

A step to paraconsistent and many–valued logics is a rather natural one for further exploring of preference relations. Many–valued logics have some properties that are not true in two–valued logics. One of these was demonstrated in article *Social choice and Lukasiewicz logic* ([Ovc91]) by Sergei Ovchinnikov, where the author, using many–valued Łukasiewicz logic, constructed a model for collective choice rule in group preferences that satisfies the property of unrestricted domain, nondictatorship, independence of irrelevant alternatives and Pareto principle and obtained a kind of result that Arrow proved that are impossible if we are working in classical logic (see [Kel78] for more details about Arrow's theorems).

2.4.1 Why paraconsistent logics?

In paraconsistent logics ([PRN89]), deriving contradiction in a logical system does not mean that any formula is derivable. Namely, we distinguish two concepts: we say that logic is *trivial* if it contains all formulae, and that it is *contradictory* if

there exists a formula φ so that both φ and $\neg\varphi$ are contained in that logic. In the classical case, those two notions have identical meaning, but in the paraconsistent one they are different.

So, why paraconsistent logics? It happens very often that our list of wishes (preferences) that we put into preference logic is contradictory, and we try to keep those contradictions local, that is, not to destroy the whole system.

2.4.2 Definitions for many–valued logics

In order to work with matrices for many–valued logics, we need a method to define a *consequence relation* formally. One method is given in [Urq86] on pages 76–77.

Let us assume that we are dealing with a fixed language L for propositional many–valued logic, and that connectives are contained in $\{\wedge, \vee, \neg, \rightarrow, \leftrightarrow\}$.

A *matrix* for the language L consists of

- an abstract algebra $\mathcal{A}$ of the appropriate type;
- a non–empty subset $D \subseteq A$ of designated elements;

If $v\colon A \mapsto 2^{Var(L)}$ is an assignment[5] of elements of the matrix to propositional variables, then v can, in the usual way, be extended to formulae.

Let Γ and Δ be finite (possibly empty) sets of formulae of L. We say that Δ is a *consequence of* Γ *with respect to matrix* M, denoted with $\Gamma \models_M \Delta$, if the following holds: for every assignment v of elements of M to variables of L, if $v(\varphi) \in D$ for all $\varphi \in \Gamma$, then $v(\psi) \in D$ for some $\psi \in \Delta$. Formula φ is a *tautology* with respect to M if $\emptyset \models_M \varphi$, which we abbreviate as $\models_M \varphi$.

2.4.3 Testing method

All paraconsistent and/or many–valued logic systems presented bellow have matrix representation for logical connectives, and for the preference relation P, we will use the following interpretation:

$$v(xPy) = \begin{cases} a, & \text{if } x = 0 \text{ and } y = 0; \\ b, & \text{if } x = 0 \text{ and } y = 1; \\ c, & \text{if } x = 1 \text{ and } y = 0; \\ d, & \text{if } x = 1 \text{ and } y = 1. \end{cases}$$

where a, b, c and d are constants that belong to the language of given logic. We again modified our `LISP` program to search for interpretations according to the given matrix.

2.4.4 System 1

This is a paraconsistent logic defined in [AT75] in 1975. Its matrix is:

$\wedge$	0	1	2	$\vee$	0	1	2	$\rightarrow$	0	1	2		$\neg$
0	0	1	2	0	0	0	0	0	0	1	2	0	1
1	1	1	1	1	0	1	2	1	0	0	0	1	0
2	2	1	2	2	0	2	2	2	0	1	2	2	2

where 0 and 2 are designated true and 1 a designated false value.

[5] We can also use a function $\sigma\colon Var(L) \mapsto A$, that is more usual.

2.4.5 System 2

This system is from [KdC89] and [BR89]. The implication in this system is the strong one in the sense that $\varphi \to \psi$ is a theorem only if φ and ψ share variables. The matrix below is also a matrix for relevance logic RM3 ([AB75, p.470]).

System 2 is defined in the following way: $\mathcal{M} = (\{1,2,3\}, \to, \vee, \wedge, \neg\}$, where $x \wedge y = \min(x,y)$ and $x \vee y = \max(x,y)$ for $x, y \in \{1,2,3\}$, and with the following matrix for $\to$ and $\neg$:

$\to$	1	2	3		$\neg$
1	3	3	3	1	3
2	1	2	3	2	2
3	1	1	3	3	1

with 3 and 2 as designated true, and 1 as a designated false value.

Note that if we substitute $1 \mapsto -1$, $2 \mapsto 0$ and $3 \mapsto 1$, we obtain Sugihara matrix S_1^0 with carrier set $I = \{-1, 0, 1\}$ and with connectives defined as in section 2.2.1, page 15.

2.4.6 System 3

These are actually two systems from [Mor89, pp.299 and 301] and from [Car85, p.367], but, applied to von Wright's preference logic, they give the same result. Those systems are in paraconsistent logics discussed in order to establish a relationship between classical logic $\mathbf{C}_0$ and paraconsistent logics $\mathbf{C}_1$, $\mathbf{C}_2$, ..., $\mathbf{C}_\omega$ investigated by da Costa ([dC74],[dCA77]).

The first system, denoted by $\mathbf{C}_{0.1}$ (or by P^1 in [Car85]) has the following matrix:

$\wedge$	1	2	3	$\vee$	1	2	3	$\to$	1	2	3		$\neg$
1	1	1	3	1	1	1	1	1	1	1	3	1	3
2	1	1	3	2	1	1	1	2	1	1	3	2	1
3	3	3	3	3	1	1	3	3	1	1	1	3	1

with 1 and 2 as designated true, and 3 as a designated false value.

The second system, denoted by $\mathbf{C}_{0.2}$, has the following matrix:

$\wedge$	1	2	3	$\vee$	1	2	3	$\to$	1	2	3		$\neg$
1	1	1	3	1	1	1	1	1	1	1	3	1	3
2	1	1	3	2	1	1	1	2	1	1	3	2	2
3	3	3	3	3	1	1	3	3	1	1	1	3	1

with 1 as a designated true, and 2 and 3 as designated false values.

2.4.7 System 4

This system is discussed in [Por84], and it represents Łukasiewicz's Ł–modal system. Here, we have two kinds of negation: a *strong* (or a classical) one ($\neg$) and a *weak* one ($\sim$). Its matrix is ([Por84], pages 87 and 88):

$\wedge$	1	2	3	4	$\vee$	1	2	3	4	$\to$	1	2	3	4		$\neg$	$\sim$
1	1	2	3	4	1	1	1	1	1	1	1	2	3	4	1	4	2
2	2	2	4	4	2	1	2	1	2	2	1	1	3	3	2	3	1
3	3	4	3	4	3	1	1	3	3	3	1	2	1	2	3	2	2
4	4	4	4	4	4	1	2	3	4	4	1	1	1	1	4	1	1

with 1 as a designated true value. We will name **4.1** the system with the strong negation $\neg$, and **4.2** the system with the weak negation $\sim$.

Note that we can obtain system 4 as a direct product of two tables of classical logic with operations defined componentwise, and substituting $(1,1) \mapsto 1$, $(1,0) \mapsto 2$, $(0,1) \mapsto 3$ and $(0,0) \mapsto 4$ (see [Res69, pp.96–97]).

2.4.8 System 5

This system is discussed in [Mor71] where it was used for checking some axioms of deontic logic. Some 5 different systems were used there for each binary connective, and we used all of them for our testing, but here we will only present the system that is mathematically 'the most correct[6] one', because, for our purpose, those differences do not show. Domain is the set $\{1, 2, \ldots, 6\}$, and the matrix for negation and conjunction is:

$\wedge$	1	2	3	4	5	6		$\neg$
1	1	2	3	4	5	6	1	6
2	2	2	3	5	5	6	2	5
3	3	3	3	6	6	6	3	4
4	4	5	6	4	5	6	4	3
5	5	5	6	5	5	6	5	2
6	6	6	6	6	6	6	6	1

with disjunction and implication defined in the standard way: $x \vee y \stackrel{\text{def}}{=} \neg(\neg x \wedge \neg y)$ and $x \to y \stackrel{\text{def}}{=} \neg x \vee y$. A formula is said to be true if its value is ≤ 3.

Other systems can be obtained modifying the table for conjunction in the following way:

variant 2: $2 \wedge 5 = 5 \wedge 2 = 6$ (instead of 5);

variant 3: $1 \wedge 4 = 4 \wedge 1 = 5$ (instead of 4);

variant 4: $1 \wedge 3 = 3 \wedge 1 = 2$ (instead of 3), $1 \wedge 6 = 6 \wedge 1 = 5$ (instead of 6), $3 \wedge 4 = 4 \wedge 3 = 5$ (instead of 6) and $4 \wedge 6 = 6 \wedge 4 = 5$ (instead of 6);

variant 5: $1 \wedge 2 = 2 \wedge 1 = 1$ (instead of 2), $1 \wedge 5 = 5 \wedge 1 = 4$ (instead of 5), $2 \wedge 4 = 4 \wedge 2 = 4$ (instead of 5) and $4 \wedge 5 = 5 \wedge 4 = 4$ (instead of 5).

2.5 Results

All those matrices are implemented on computer in order to find the best interpretations in the following sense:

- As many axioms as possible should hold;
- The axioms (W1) and (W2) *must* be valid;
- Formulae (16), (17) and (21) should not be valid;
- As many formulae as possible from the set (1)–(21) except from those mentioned above should be valid;
- The opposite of (20) should not be valid;

Here is a table with results based on presented criteria:

[6] In some of them even the associativity of conjunction does not hold.

System	a	b	c	d	Invalid axioms	Invalid formulae	Valid converses
E_{fde}	2	4	1	2	w5,w7	—	—
	2	4	4	4	w4,w6	4,5,11,13,16,17,21	o5,o6,o12,o15
	3	4	1	2	w3,w4,w5,w6,w7	3,4,5,16,17	—
	3	4	3	4	w4,w6	4,5,16,17,21	o5,o6,o12,o15
	3	4	4	4	w4,w6	4,5,11,13,16,17,21	o5,o6,o12,o15
sugihara	–	–	–	–	w3,w5,w7	3,6,7,8,16,17,19,21	—
system 1	2	1	0	1	w4,w6	4,5,16,17,21	o5,o6,o12,o15
	2	1	1	1	w4,w6	4,5,11,13,15,17,21	o5,o6,o12,o15
	2	1	2	1	w4,w6	4,5,16,17,21	o5,o6,o12,o15
system 2	1	1	3	2	w3,w5,w7	3,16,17,21	o7
	2	1	1	1	w4,w6	4,5,11,13,16,17,21	o5,o6,o12,o15
	2	1	2	1	w4,w6	4,5,16,17,21	o5,o6,o12,o15
	2	1	3	1	w4,w5,w6,w7	4,5,16,17,21	—
system 3	2	3	1	3	w4,w6	4,5,16,17,21	o5,o6,o12,o15
	2	3	2	3	w4,w6	4,5,16,17,21	o5,o6,o12,o15
	2	3	3	3	w4,w6	4,5,11,13,16,17,21	o5,o6,o12,o15
system 4.1	–	–	–	–	–	–	–
system 4.2	2	2	3	4	w4,w5,w6,w7	4,5,11,13,16,17,21	–
	2	3	2	4	w4,w5,w6,w7	4,5,8,11,13,16,17,19,21	o12,o15
	2	3	4	4	w4,w5,w6,w7	4,5,8,11,13,16,17,19,21	o12,o15
	2	4	1	4	w4,w5,w6,w7	4,5,16,17,21	–
	2	4	2	4	w4,w6	4,5,16,17,21	o5,o6,o12
	2	4	3	4	w4,w5,w6,w7	4,5,11,13,16,17,21	–
	2	4	4	4	w4,w6	4,5,11,13,16,17,21	o5,o6,o12,o15
	4	2	3	2	w3,w5,w7	3,8,16,17,19,21	o7
	4	4	1	2	w3,w5,w7	3,16,17,21	o7
	4	4	3	2	w3,w5,w7	3,8,16,17,19,21	o7

Results for the system 5 are not presented in our table, but here they are:

- There is no interpretation (out of $6^4 = 1296$ different) that satisfies all conditions;
- In all interpretations the formula (20) is valid;
- If we weaken our conditions so that (16) and (17) are allowed to be valid, then we obtain the following results: if $a \in \{4, 5, 6\}$ and $b \in \{4, 5, 6\}$ and $c \in \{1, 2, 3\}$ and $d \in \{4, 5, 6\}$, then[7]:
  - In all interpretations mentioned above, from the set of axioms only (w5) and (w7) are invalid;
  - In all interpretations above, all formulae (1)–(20) are valid, that is, only (21) is invalid;
  - In all above–mentioned interpretations, from converses, only (o7) is valid.

Note also that in the system 5 all axioms and rules of E_{fde} are valid.

2.5.1 Remarks about the table of results

- Labels for formulae in the table have the following meaning:

[7]Note that this interpretation is the "natural one" in the sense described at the beginning of section 2.3.

(w1) is the axiom (W1);
(w2) is the axiom (W2);
(w3) is the axiom (W3) from left to right;
(w4) is the axiom (W3) from right to left;
(w5) is the axiom (W4) from left to right;
(w6) is the axiom (W4) from right to left;
(w7) is the axiom (W5) from left to right;
(w8) is the axiom (W5) from right to left;

– Label (for instance) (o7) denotes converse of the formula (7), i.e. other direction of implication in that formula (see section 2.1.4 on page 10).
– The first and the third interpretation in E_{fde} are not of the above–mentioned type, but they are included in the table for the sake of completeness.
– The interpretation in Sugihara matrix is also not of the above–mentioned type, but it is also presented for the sake of completeness.

2.6 Opposite problem

In previous sections, we have seen how axioms and formulae 'behave' in various logic systems: relevance, paraconsistent and many–valued. We can also bring out the opposite question: is it possible to define logical connectives in such a way that all von Wright's axioms are valid but none of the formulae (16), (17), (20) and (21) is? Of course, all that under the restriction that inner logic is the classical one. This restriction is seemingly a logical one, as we should expect states of affairs to 'behave' in a natural way.

2.6.1 Definition of interpretations

In the line with those remarks, we will distinguish two kinds of logical connectives: classical connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$ and $\leftrightarrow$ for operations *in* preferences, and connectives $\wedge_P$, $\vee_P$, $\neg_P$, $\rightarrow_P$ and $\leftrightarrow_P$ for operations *on* preferences. With P^f we will denote an interpretation of relation P and will use lower–case Greek letters for propositional formulae ($\varphi, \psi \ldots$) and upper–case Greek letters for formulae of the logic of preference ($\Phi, \Psi, \ldots$). Furthermore, we will write 'preference formula' for the term 'formula of the logic of preference'.

Definition of preference formulae: The set of *preference formulae* is the smallest set closed under the following two rules:

(1^0) If φ and ψ are propositional formulae, then $\varphi P \psi$ is a preference formula[8].

(2^0) If Φ and Ψ are preference formulae then $\neg\Phi$, $\Phi \wedge \Psi$, $\Phi \vee \Psi$, $\Phi \rightarrow \Psi$ and $\Phi \leftrightarrow \Psi$ are preference formulae, too.

Definition of interpretation: For propositional formulae, interpretation f is the usual one, and for the preference formulae, we have the following cases:

(1^0) If φ and ψ are propositional formulae, then $f(\varphi P \psi) = f(\varphi) P^f f(\psi)$.

(2^0) If Φ and Ψ are preference formulae, then:

(a) $f(\neg\Phi) = \neg_P f(\Phi)$

[8]Note that, because of their nature, we usually restrict φ and ψ to formulae only consisting of $\wedge$, $\vee$, $\neg$ and propositional variables.

(b) $f(\Phi \wedge \Psi) = f(\Phi) \wedge_P f(\Psi)$
(c) $f(\Phi \vee \Psi) = f(\Phi) \vee_P f(\Psi)$
(d) $f(\Phi \to \Psi) = f(\Phi) \to_P f(\Psi)$
(e) $f(\Phi \leftrightarrow \Psi) = f(\Phi) \leftrightarrow_P f(\Psi)$

In the rest of this section, we will assume that the interpretation f for preference relation P is the usual one:

$$f(xPy) = \begin{cases} a, & \text{if } x = 0 \text{ and } y = 0; \\ b, & \text{if } x = 0 \text{ and } y = 1; \\ c, & \text{if } x = 1 \text{ and } y = 0; \\ d, & \text{if } x = 1 \text{ and } y = 1. \end{cases}$$

2.6.2 Results

We examined tables for all formulae, and here are the results:

- Examination of the axiom (W1) gives the following constraints:
  $a \to \neg a$, $b \to \neg c$, $c \to \neg b$ and $d \to \neg d$ (U–W1)
- Examination of the axiom (W2) gives the following constraints:
  $a \wedge b \to b$, $b \wedge c \to a$, $b \wedge d \to b$, $c \wedge a \to c$, $c \wedge b \to d$, $d \wedge c \to c$ (U–W2)
- Examination of the axiom (W3) gives the following constraint:
  $a = d$ (U–W3)

It is also a corollary to the following theorem:

Theorem 2.6.1 *For* (W3) *i.e.* $xPy \leftrightarrow (x \wedge \neg y)P(y \wedge \neg x)$ *to be valid,* $f(xPx) = f(yPy)$ *must hold for all* x *and* y *and for all interpretations* f*.*

Proof: Proof is case–based. If x and y are equivalent, then both $(x \wedge \neg y)$ and $(y \wedge \neg x)$ are false, so the left hand side reduces to $f(0P0)$, and the right hand side could be either $f(0P0)$ or $f(1P1)$, so we must have $f(xPx) = f(yPy)$ for (W3) to hold. If x and y are not equivalent, then (easy check) $x \leftrightarrow (x \wedge \neg y)$, and $y \leftrightarrow (y \wedge \neg x)$, so the formula is always true. □

However, the following theorem is also true:

Theorem 2.6.2 *If for all* x *and* y *we have* $f(xPx) = f(yPy)$*, then formulae* (16) *and* (17) *are also valid, that is*

$$xPy \leftrightarrow \neg yP\neg x.$$

Proof: Similar to the proof of theorem 2.11.1 on page 38, except there are not that many cases (the fourth case is not needed). □

So, we have to choose: either we do not have the axiom (W3), or we have both this axiom and the Martin's axiom (formulae (16), (17)).

This is hardly a surprising result, as we also have the following lemma:

Lemma 2.6.1 *The following holds:*

(a) (W3) $\vdash$ (16)
(b) (W3) $\vdash$ (17)

Proof: Using (W3), we obtain $xPy \rightleftarrows (x \wedge \neg y)P(y \wedge \neg x)$, and $\neg yP\neg x \rightleftarrows (\neg y \wedge \neg\neg x)P(\neg x \wedge \neg\neg y)$, that is $\neg yP\neg x \rightleftarrows (x \wedge \neg y)P(y \wedge \neg x)$, and, because implication is transitive, we finally get $xPy \rightleftarrows \neg yP\neg x$. □

Corollary 2.6.1 *For* (W3) *to be valid, and* (16) *and* (17) *to be invalid, we need at least one of the conditions:*
1^0 *In outer logic, implication should not be transitive;*
2^0 *In inner logic, formula* $x \rightleftarrows \neg\neg x$ *should not be valid.*

The first condition in corollary 2.6.1 is rather an unusual one, and it is not clear what a logic is without transitive implication, and the second condition is not acceptable for the von Wright's concept of preference relation.

• Examination of the axiom (W4) gives the following constraints (we present only nontrivial constraints):
$c \leftrightarrow a \wedge c$, $b \leftrightarrow a \wedge b$, $d \leftrightarrow a \wedge a$ (U–W4)

• Examination of the axiom (W5) gives the following constraints:
$b \leftrightarrow a \wedge b$, $c \leftrightarrow a \wedge c$, $d \leftrightarrow a \wedge d$ (U–W5)
but the third constraint is not necessary because of (U–W3), and the first two are already contained in (U–W4).

• The formula (8) gives the following constraints:
$b \rightarrow a \wedge b$, $d \rightarrow c \wedge d$ (U–8)
so we can expect that there are models for (W1)–(W5) where (8) is invalid. Because of (U–W4), the first constraint in (U–8) must hold, so, only the second one could be a candidate for invalidness of the formula (8). Since (U–W3) implies $a = d$, the formula (8) will not be valid if $a \rightarrow c \wedge a$ is invalid.

• The formula (9) gives the following constraints:
$a \wedge b \rightarrow b$, $c \wedge d \rightarrow d$. (U–9)
that is, since the first constraint, because of (U–W4), must be valid, this formula will not be valid if $c \wedge d \rightarrow d$ is invalid, i.e. (because of (U–W3)) if $c \wedge a \rightarrow a$ is invalid.

So, we proved the following theorem:

Theorem 2.6.3 *For the formula* (9) *to be invalid, we need the formula* $x \wedge y \rightarrow x$ *to be invalid.*

• The formula (10) gives the following constraints:
$c \rightarrow a \vee c$, $d \rightarrow b \vee d$ (U–10)
that is, because of (U–W3), $c \rightarrow a \vee c$ and $a \rightarrow b \vee a$, and so we immediately have the following theorem:

Theorem 2.6.4 *For invalidness of the formula* (10)*, we need the invalidness of the formula* $x \rightarrow x \vee y$.

• The formula (11) gives the following constraints:
$a \vee c \rightarrow c$, $d \vee b \rightarrow d$ (U–11)
that is, because of (U–W3), $a \vee c \rightarrow c$ and $a \vee b \rightarrow a$, so we can safely expect that there is an interpretation in which (W1)–(W5) are valid and (11) is not.

• The formula (12) gives the following constraints:
$a \wedge c \rightarrow c$, $b \wedge d \rightarrow d$ (U–12)
so, since $a \wedge c \rightarrow c$, because of (U–W4), must be valid, the constraints for the invalidness of the formula (12) are given in the theorem 2.6.4.

• The formula (13) gives the following constraints:
$a \rightarrow a \wedge c$, $b \rightarrow b \wedge d$ (U–13)
that is, since (U–W3) reduces the second constraint to $b \rightarrow b \wedge a$, which is true according to (U–W4), the invalidness of the formula (13) is reduced to the invalidness of the formula $a \rightarrow a \wedge c$, and that means that we should expect that there are models for von Wright's logic where (13) is non valid.

• The formula (14) gives the following constraints:
$a \wedge c \to a$, $b \wedge d \to b$ (U–14)
that is, since (U–W3) reduces the second constraint to $b \wedge a \to b$, which is true according to (U–W4), the invalidness of the formula (14) is reduced to the invalidness of the formula $a \wedge c \to a$, and conditions for the invalidness of the formula (14) are given in the theorem 2.6.3.

• The formula (15) gives the following constraints:
$a \to a \vee c$, $b \to b \vee d$ (U–15)
that is, since (U–W3) reduces the second constraint to $b \to b \vee a$, the constraints are $a \to a \vee c$ and $b \to a \vee b$, and the conditions for the invalidness of the formula (15) are given in the theorem 2.6.4.

• For the formulae (16) and (17) theorem 2.6.2 shows that they must be valid.

• The formula (18) gives the following constraints:
$a \to a \vee b$, $c \to c \vee d$ (U–18)
that is, since (U–W3) reduces the second constraint to $c \to c \vee a$, the constraints are $a \to a \vee b$ and $c \to a \vee c$, and the conditions for the invalidness of the formula (18) are given in the theorem 2.6.4.

• The formula (19) gives the following constraints:
$a \vee b \to a$, $c \vee d \to c$ (U–19)
that is, since (U–W3) reduces the second constraint to $c \vee a \to c$, the formula (19) is invalid if both $a \vee b \to a$ and $a \vee c \to c$ are invalid, and that means that we should expect that there are models for von Wright's logic where (19) is invalid.

• The formula (20) gives the following constraints:
$a \wedge b \to a$, $c \wedge d \to c$ (U–20)
that is, because of (U–W3), $a \wedge b \to a$ and $a \wedge c \to c$. The second constraint is necessarily true (because of (U–W5)), and that means that the invalidness of (20) is reduced to the the invalidness of $a \wedge b \to a$, and the conditions for the invalidness of (20) are given in the theorem 2.6.3.

• Finally, the formula (21) gives the following constraints:
$\neg a \to a$, $\neg c \to b$, $\neg b \to c$, $\neg d \to d$ (U–21)
and that means, because of (U–W1), that (21) is invalid in interpretations where $b \neq \neg c$ or $a \neq \neg a$.

To conclude, there are interpretations in all logics where the formulae (8), (11), (13), (19) and (21) are invalid, and for the other formulae, we need conditions given in the theorems 2.6.3 and 2.6.4, i.e.:

- Formula (9) is invalid if not $\vdash a \wedge c \to a$ in outer logic;
- The formula (10) is invalid if not $\vdash c \to a \vee c$ or if not $\vdash a \to a \vee b$ in outer logic;
- The formula (12) is invalid if not $\vdash a \wedge b \to a$ in outer logic;
- The formula (14) is invalid if not $\vdash a \wedge c \to a$ in outer logic, so (9) and (14) are either both valid or both invalid;
- The formula (15) is invalid if not $\vdash a \to a \vee c$ or if not $\vdash b \to a \vee b$ in outer logic;
- The formula (18) is invalid if not $\vdash a \to a \vee b$ or if not $\vdash c \to a \vee c$ in outer logic, so the formulae (10) and (18) are either both valid or both invalid;
- The formula (20) is invalid if not $\vdash a \wedge b \to a$ in outer logic, so the formulae (12) and (20) are either both valid or both invalid;

Of course, the question is if we should go that far, and investigate so weak logics where the theorems 2.6.3 and 2.6.4 hold. One example of such a logic is Kleene's 3–valued logic $\mathbf{K}_3$ ([Res69]) if we choose only $\top$ for a designated true value. Another example is Linear logic ([Gir87]) if we choose its operator $\otimes$ ('tensor') for a conjunction, $\parr$ ('par') for a disjunction, and if we interpret our implication $\rightarrow$ as a linear implication $\multimap$. The big question is if those systems can validate all von Wright's axioms, so we will not investigate this question any further.

2.7 New interpretations

So, the previous section showed that it is impossible (at least in this way) to obtain an interpretation that fulfills all our wishes from the beginning of section 2.5. Because of that, we established a new goal — to find interpretations in above–presented logical systems where *all* von Wright's axioms are valid. The table in section 2.7.2 below shows the results of that goal. The notation is as above, but the table is slightly different because there is no column of valid axioms – they are all valid. Furthermore, concerning converses, we do not consider the formula (o5), since it is one direction of the axiom (W4) and (o6), since it is one direction of the axiom (W5). We give only those interpretations where (21) does not hold[9] and where converse of (20) does not hold[10]. In our investigations, we also included Kleene's 3–valued logic $\mathbf{K}_3$ because of the reasons given at the end of section 2.6.2.

2.7.1 Kleene's 3–valued logic

Matrix for the logic $\mathbf{K}_3$ is ([Res69, p.34]):

$\wedge$	$\top$	I	$\bot$
$\top$	$\top$	I	$\bot$
I	I	I	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$

$\vee$	$\top$	I	$\bot$
$\top$	$\top$	$\top$	$\top$
I	$\top$	I	I
$\bot$	$\top$	I	$\bot$

$\rightarrow$	$\top$	I	$\bot$
$\top$	$\top$	I	$\bot$
I	$\top$	I	I
$\bot$	$\top$	$\top$	$\top$

	$\neg$
$\top$	$\bot$
I	I
$\bot$	$\top$

We are going to consider two systems: kleene 1, with a designated true value $\top$, and kleene 2, with designated true values $\top$ and I.

Actually, except for all those systems, we also considered two further variants of Kleene's logic described in [MD90]. Both are defined on the same domain as the original system, conjunction, disjunction and negation are as in $\mathbf{K}_3$, the difference is only in implication. The first system is a system of Przymusinski, and the second one is their own (i.e. Matheieu–Delahaye). Tables for implication are as follows ([MD90, p.388]):

$\rightarrow$	$\top$	$\bot$	I
$\top$	$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\top$
I	$\top$	$\bot$	$\top$

$\rightarrow$	$\top$	$\bot$	I
$\top$	$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\top$
I	$\top$	$\top$	$\top$

But those systems do not make any difference for our purposes, so we did not want to include them in our table (the table is already big enough!). Also, 3–valued logics of Słupecki and Sobociński ([BB92, pp.76–77]) did not change the general picture.

[9] The reason why we try to avoid (21) is that if it is valid, then we are not able to define an indifference relation as $xIy \leftrightarrow \neg(xPy) \wedge \neg(yPx)$.

[10] Having to admit (20) is bad enough, but having both implications is even worse.

2.7.2 Results

So, the new results are:

System	a	b	c	d	Invalid formulae	Valid converses
E_{fde}	2	4	2	2	21	–
	2	4	4	2	8,11,13,19,21	–
	3	4	3	3	21	–
	3	4	4	3	8,11,13,19,21	–
system 1	2	1	1	2	8,11,13,19,21	–
	2	1	2	2	21	–
system 2	2	1	1	2	8,11,13,19,21	–
	2	1	2	2	21	–
system 3	2	3	2	2	21	–
	2	3	3	2	8,11,13,19,21	–
system 4.1	–	–	–	–	–	–
system 4.2	2	4	2	2	21	–
	2	4	4	2	8,11,13,19,21	–
system 5	–	–	–	–	–	–
kleene 1	–	–	–	–	–	–
kleene 2	I	I	$\top$	$\bot$	21	o7,o12,o15
	I	$\bot$	$\top$	$\bot$	21	o7,o12,o15

2.7.3 Von Wright's extended system

We will call *von Wright's extended system* one that is given in [vW72] with axiom

$$(\mathrm{W2}') \quad xPy \to xPz \vee zPy$$

instead of the axiom (W2). As already mentioned, from this axiom we can derive (W2), and in the extended system an indifference relation satisfies all conditions for the strong indifference relation in [vW63].

The natural question is: what is the behavior of extended system in the above–mentioned interpretations? Checking it out, we get some very interesting results: Namely, the formula (W2′) is only valid in those models where all formulae except (21) are valid. In other models, it is not valid.

So, we can conjecture that there is a connection between the formula (W2′) and (at least one of) the formulae (8), (11), (13) and (19).

2.8 Indifference relation

2.8.1 Definitions

We are going to include indifference relation into our investigations. Von Wright ([vW63]) distinguishes two different indifference relations:

- the *weak* one, I, defined by $xIy \leftrightarrow \neg(xPy) \wedge \neg(yPx)$
- the *strong* one E. We say that xEy if and only if under no circumstances is state $x \wedge \neg y$ preferred to $y \wedge \neg x$ nor vice versa. According to Huber ([Hub72]), the relation E is defined by:

  $$xEy \leftrightarrow \neg((x \wedge \neg y)P(y \wedge \neg x)) \wedge \neg((y \wedge \neg x)P(x \wedge \neg y))$$

  but it seems that it is not what von Wright had in mind.

2.8.2 Weak indifference relation

The main properties of this relation are (see [vW63],[vW72] or [Hub72]):

(i1) xIx

(i2) $xIy \rightarrow yIx$

(i3) $xIy \wedge yIz \rightarrow xIz$[11]

Checking out the formulae (i1)–(i3) in our models of von Wright's logic (i.e. in E_{fde}, system 1–3, system 4.2 and kleene 2), we get the following results:

System	a	b	c	d	Valid formulae
E_{fde}	2	4	2	2	i1,i2,i3
	2	4	4	2	i1,i2
	3	4	3	3	i1,i2,i3
	3	4	4	3	i1,i2
system 1	2	1	1	2	i1,i2,i3
	2	1	2	2	i1,i2,i3
system 2	2	1	1	2	i1,i2
	2	1	2	2	i1,i2,i3
system 3	2	3	2	2	i1,i2,i3
	2	3	3	2	i1,i2,i3
system 4.2	2	4	2	2	i1,i2,i3
	2	4	4	2	i1,i2,i3
kleene 2	I	I	$\top$	$\bot$	i1,i2,i3
	I	$\bot$	$\top$	$\bot$	i1,i2,i3

2.8.3 Strong indifference relation

Contrary to the weak indifference relation, which was defined by means of the preference relation P, the strong indifference relation has to be considered as an independent relation. Of course, because of principles (e1) and (e2) below, it is not completely independent, at least not in interpretations we are considering. According to von Wright, this relation should be an equivalence relation, should satisfy the axioms (W2)–(W5) (with E instead of P) as well as the following two principles:

(e1) $xPy \wedge xEz \rightarrow zPy$

(e2) $xPy \wedge yEz \rightarrow xPz$

As already said, if we drop the axiom (W2) and add $xPy \rightarrow xPz \vee zPy$ instead, as von Wright did in his extended system ([vW72]), then the weak indifference relation also satisfies (e1) and (e2) what otherwise is not the case.

The table of results for strong indifference relation is as follows (we only present those interpretations where all above–presented constraints hold):

[11] Formally, the transitivity is not necessary, but we will consider it to restrict the class of models we are exploring.

System	Preference	a	b	c	d
E_{fde}	2 4 2 2 or 2 4 4 2	1	3	3	1
		1	4	4	1
		2	4	4	2
		3	3	3	3
		3	4	4	3
	3 4 3 3 or 3 4 4 3	1	2	2	1
		1	4	4	1
		2	2	2	2
		2	4	4	2
		3	4	4	3
system 1	2 1 1 2 or 2 1 2 2	0	1	1	0
		0	1	1	2
		2	1	1	0
		2	1	1	2
system 2	2 1 1 2 or 2 1 2 2	2	1	1	2
		3	1	1	3
system 3	2 3 2 2 or 2 3 3 2	1	3	3	1
		1	3	3	2
		2	3	3	1
		3	3	3	2
system 4.2	2 4 2 2 or 2 4 4 2	1	3	3	1
		1	4	4	1

The system kleene 2 is not included in the table because there are too many interpretations for the relation E (exactly $2 \cdot 2 \cdot 3 \cdot 2 = 24$), but, basically, the results are as follows: for both interpretations for P, we have the same results: $a \in \{\top, I\}$, $b \in \{I, \bot\}$, $c \in \{\top, I, \bot\}$ and $d \in \{\top, I\}$.

For the above–presented table we only considered the interpretations where all von Wright's axioms were valid, and for the relation E, we were considering interpretations of the following type:

$$f(xEy) = \begin{cases} a, & \text{if } x = 0 \text{ and } y = 0; \\ b, & \text{if } x = 0 \text{ and } y = 1; \\ c, & \text{if } x = 1 \text{ and } y = 0; \\ d, & \text{if } x = 1 \text{ and } y = 1. \end{cases}$$

2.9 Further investigations

Since all logical systems give, more or less, the same results, in the sequel we will explore only two (or three) different systems. One of them is the system 2 (defined in section 2.4.5 on page 20) with interpretation:

$$xPy = \begin{cases} 2, & \text{if } x = 0 \text{ and } y = 0; \\ 1, & \text{if } x = 0 \text{ and } y = 1; \\ 2, & \text{if } x = 1 \text{ and } y = 0; \\ 2, & \text{if } x = 1 \text{ and } y = 1. \end{cases} \quad \text{and } xEy = \begin{cases} 3, & \text{if } x = 0 \text{ and } y = 0; \\ 1, & \text{if } x = 0 \text{ and } y = 1; \\ 1, & \text{if } x = 1 \text{ and } y = 0; \\ 3, & \text{if } x = 1 \text{ and } y = 1. \end{cases}$$

There are several reasons for that, but the most important are:

- This system has a well–known and (in [AB75]) well–developed logic;
- Interpretation of the preference relation is the 'natural' one;

- The axioms of E_{fde} are valid in this logic (actually, E_{fde} is its subsystem), so we can use (with slight modifications) all proofs from section 2.1.7 on page 12.

Note that, although the formula (W2′) is valid in this interpretation, we still have two different indifference relations. The reason for that is that for proving (e1) and (e2), except for the formula (W2′) and definition of the indifference relation, we also need a tautology $(x \vee y) \wedge \neg y \leftrightarrow x \wedge \neg y$, which is not true in this logic. That means that I and E are different in this system.

A logic 'behind' the system 2 is the relevance logic RM3.

2.9.1 Definition of relevance logic RM3

The relevance logic RM3 is described in [AB75], and its axioms are ([AB75, pp.231–232,470]):

(A1) $((A \to A) \to B) \to B$

(A2) $(A \to B) \to ((B \to C) \to (A \to C))$

(A3) $(A \to (A \to B)) \to (A \to B)$

(A4) $A \wedge B \to A$

(A5) $A \wedge B \to B$

(A6) $(A \to B) \wedge (A \to C) \to (A \to B \wedge C)$

(A7) $\Box A \wedge \Box B \to \Box(A \wedge B)$

(A8) $A \to A \vee B$

(A9) $B \to A \vee B$

(A10) $(A \to C) \wedge (B \to C) \to (A \vee B \to C)$

(A11) $A \wedge (B \vee C) \to (A \wedge B) \vee C$

(A12) $(A \to \neg A) \to \neg A$

(A13) $(A \to \neg B) \to (B \to \neg A)$

(A14) $\neg\neg A \to A$

(A15) $A \rightleftarrows (\neg A \to A)$

(A16) $A \to ((A \to B) \to B)$

(A17) $A \vee (A \to B)$

and inference rules are:

$$\text{(r1)}\ \frac{\vdash A \quad \vdash B}{\vdash A \wedge B}$$

$$\text{(MP)}\ \frac{\vdash A \quad \vdash A \to B}{\vdash B}$$

In the list of axioms, the axioms (A1)–(A14) are the axioms of a relevance logic E, and the modal operator of necessity $\Box$ in the axiom (A7) is defined as $\Box A \stackrel{\text{def}}{=} (A \to A) \to A$.

Because of the axiom (A2) and of the modus ponens (i.e. the rule (MP)), the rule (R1) of logic E_{fde} (transitivity of implication) is valid here, because of (A6) the rule (R2) of E_{fde} is also valid, because of (A10) the rule (R3) of E_{fde} is also valid and because of the axiom (A13) the rule (R4) of E_{fde} is also valid in RM3. So, we can safely use proofs of the lemmas 2.1.2–2.1.3 on pages 12–13 in this system.

2.9.2 Dependences between formulae in RM3

With respect to the above–mentioned remarks we can immediately state the following lemmas. We will use the same notation as before.

Lemma 2.9.1 *The following is true:*

- **(a)** $(12), (9), (4) \vdash (5)$*;*
- **(b)** $(13), (9) \vdash (6)$*;*
- **(c)** $(11) \vdash (12)$*;*
- **(d)** $(19) \vdash (20)$*;*
- **(e)** $(13) \vdash (15)$*;*
- **(f)** $(10), (13), (16), (17), (18) \vdash (4)$.

Proof:
(a) The same as the proof of lemma 2.1.2 **(a)** on page 12.
(b) The same as the proof of lemma 2.1.2 **(c)** on page 12.
(c) The same as the proof of lemma 2.1.2 **(d)** on page 12.
(d) The same as the proof of lemma 2.1.2 **(e)** on page 12.
(e) The same as the proof of lemma 2.1.2 **(f)** on page 13.
(f) The same as the proof of lemma 2.1.2 **(g)** on page 13.

Lemma 2.9.2 *The following is true:*

- **(a)** $(10), (16), (17) \vdash (18)$*;*
- **(b)** $(11), (16), (17) \vdash (19)$*;*
- **(c)** $(14), (16), (17), (19) \vdash (3)$.

Proof:
(a) The same as the proof of lemma 2.1.4 **(c)** on page 14.
(b) The same as the proof of lemma 2.1.4 **(d)** on page 14.
(c) The same as the proof of lemma 2.1.4 **(e)** on page 14.

Lemma 2.9.3 *The following is true:*

- **(a)** $(w7) \vdash (7)$
- **(b)** $(8), (16), (17) \vdash (13)$
- **(c)** $(9), (16), (17) \vdash (14)$

Proof:
(a)

$xPy \rightarrow (x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$	(formula (w7))
$(x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha) \rightarrow (x \wedge \alpha)P(y \wedge \alpha)$	(axiom (A4))
$(x \wedge \alpha)P(y \wedge \alpha) \rightarrow (x \wedge \alpha)P(y \wedge \alpha) \vee (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$	(axiom (A8))
$xPy \rightarrow (x \wedge \alpha)P(y \wedge \alpha) \vee (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$	(transitivity of implication)

(b)

$(x \wedge y)Pz \rightarrow \neg zP\neg(x \wedge y)$	(formula (16))
$\neg zP\neg(x \wedge y) \rightarrow \neg zP(\neg x \vee \neg y)$	(de Morgan's laws)
$\neg zP(\neg x \vee \neg y) \rightarrow \neg zP\neg x \wedge \neg zP\neg y$	(formula (8))
$\neg zP\neg x \wedge \neg zP\neg y \rightarrow xPz \wedge yPz$	(formula (17))
$(x \wedge y)Pz \rightarrow xPz \wedge yPz$	(transitivity of implication)

(c)

$xPz \wedge yPz \to \neg zP\neg x \wedge \neg zP\neg y$ (formula (16))
$\neg zP\neg x \wedge \neg zP\neg y \to \neg zP(\neg x \vee \neg y)$ (formula (9))
$\neg zP(\neg x \vee \neg y) \to \neg zP\neg(x \wedge y)$ (de Morgan's laws)
$\neg zP\neg(x \wedge y) \to (x \wedge y)Pz$ (formula (17))
$xPz \wedge yPz \to (x \wedge y)Pz$ (transitivity of implication)
□

So, the list of formulae we are left with is:
$xPy \to \neg(yPx)$ (w1)
$xPy \wedge yPz \to xPz$ (w2)
$(x \vee y)P(z \vee u) \to (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u)$ (w5)
$xPy \to (x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$, where α is a new variable (w7)
$xP(y \vee z) \rightleftarrows xPy \wedge xPz$ (8), (9)
$(x \vee y)Pz \rightleftarrows xPz \vee yPz$ (10), (11)
$xPy \rightleftarrows \neg yP\neg x$ (16), (17)

Alternatively, we can make an axiomatization which contains all von Wright's original axioms. For this case, we will need the following lemma:

Lemma 2.9.4 *The following is true:*

(a) $(W3) \vdash (16), (17)$*;*

(b) $(W4) \vdash (W3)$.

Proof:

(a) The same as the proof of lemma 2.6.1 on page 24.

(b) If we substitute $x = y$ and $z = u$ in (W4), we obtain our result using laws of propositional logic (see also [KM75]). □

Hence, in that case the axiomatization will be (not necessarily the minimal one):

(W^*1) $xPy \to \neg(yPx)$

(W^*2) $xPy \wedge yPz \to xPz$

(W^*3) $(x \vee y)P(z \vee u) \rightleftarrows (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u)$

(W^*4) $xPy \rightleftarrows (x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$, where α is a new variable

(W^*5) $xP(y \vee z) \rightleftarrows xPy \wedge xPz$

(W^*6) $(x \vee y)Pz \rightleftarrows xPz \vee yPz$

We can also note that the formulae (8) and (9) (i.e. (W^*5)) are in a way derivations of (W3) and (W4). Indeed, using (W3) we get:

$$xP(y \vee z) \rightleftarrows (x \wedge (\neg y \wedge \neg z))P((y \wedge \neg x) \vee (z \wedge \neg x))$$

whereas (W4) gives:

$$xP(y \vee z) \rightleftarrows (x \wedge (\neg y \wedge \neg z))P(y \wedge \neg x) \wedge (x \wedge (\neg y \wedge \neg z))P(z \wedge \neg x)$$

and using transitivity we finally get:

$$(x \wedge (\neg y \wedge \neg z))P((y \wedge \neg x) \vee (z \wedge \neg x)) \rightleftarrows (x \wedge (\neg y \wedge \neg z))P(y \wedge \neg x) \wedge (x \wedge (\neg y \wedge \neg z))P(z \wedge \neg x).$$

2.10 Alternative system

During our search for admissible interpretations, we obtained two different classes of models: one where from the formulae (1)–(21) only (21) is invalid and which is represented by the relevance logic RM3, and the other one where the formulae (8), (11), (13) and (19) are invalid, too. This class of models cannot be obtained in the

logic RM3 because the axiom that the weak indifference relation is transitive is not fulfilled there. One system that gives a model that is in that class is the system 1 (described in section 2.4.4 on page 19) with interpretation:

$$xPy = \begin{cases} 2, & \text{if } x = 0 \text{ and } y = 0; \\ 1, & \text{if } x = 0 \text{ and } y = 1; \\ 1, & \text{if } x = 1 \text{ and } y = 0; \\ 2, & \text{if } x = 1 \text{ and } y = 1. \end{cases} \text{ and } \quad xEy = \begin{cases} 0, & \text{if } x = 0 \text{ and } y = 0; \\ 1, & \text{if } x = 0 \text{ and } y = 1; \\ 1, & \text{if } x = 1 \text{ and } y = 0; \\ 0, & \text{if } x = 1 \text{ and } y = 1. \end{cases}$$

This logic is described in [AT75].

2.10.1 Definition of the system 1

A logic behind the system 1 is a paraconsistent logic **Ł**. In this logic, we distinguish two kinds of variables: A–variables $(A_1, A_2, \ldots)$ for variables that take values 0 and 1, and B–variables $(B_1, B_2, \ldots)$ for variables that take only the value 2. We will write $\mathcal{A}_1, \mathcal{A}_2, \ldots$ for formulae that take only values 0 and 1 and $\mathcal{B}_1, \mathcal{B}_2, \ldots$ for formulae that take all 3 values.

We define $\mathcal{B}$–formulae in the following way:

1^0 All variables are $\mathcal{B}$–formulae;

2^0 If $\mathcal{B}_1$ and $\mathcal{B}_2$ are $\mathcal{B}$–formulae, then $\mathcal{B}_1 \to \mathcal{B}_2$, $\mathcal{B}_1 \wedge \mathcal{B}_2$, $\mathcal{B}_1 \vee \mathcal{B}_2$ and $\neg\mathcal{B}_1$ are also $\mathcal{B}$–formulae.

$\mathcal{A}$–formulae are defined in the following way:

1^0 All A–variables are $\mathcal{A}$–formulae;

2^0 Formulae $A_1 \to A_2$, $A_1 \wedge A_2$, $A_1 \vee A_2$, $\neg A_1$, $B_1 \to A_1$ and $A_1 \to (A_1 \vee B_1)$ are $\mathcal{A}$–formulae;

3^0 If $\mathcal{A}_1$ and $\mathcal{A}_2$ are $\mathcal{A}$–formulae and $\mathcal{B}_1$ arbitrary formula, then $\mathcal{A}_1 \to \mathcal{A}_2$, $\mathcal{A}_1 \wedge \mathcal{A}_2$, $\mathcal{A}_1 \vee \mathcal{A}_2$, $\neg\mathcal{A}_1$, $\mathcal{B}_1 \to \mathcal{A}_1$ and $\mathcal{A}_1 \to \mathcal{A}_1 \vee \mathcal{B}_1$ are also $\mathcal{A}$–formulae;

4^0 If $\neg\mathcal{B}_1$ is an $\mathcal{A}$–formula, then $\mathcal{B}_1$ is an $\mathcal{A}$–formula;

5^0 Formulae $\neg\mathcal{A}_1 \to (\mathcal{A}_1 \to \mathcal{B}_1)$ i $\neg\mathcal{A}_1 \to \neg(\mathcal{A}_1 \wedge \mathcal{B}_1)$ are $\mathcal{A}$–formulae;

6^0 Theorems obtained from $\mathcal{A}$–formulae using only modus ponens are also $\mathcal{A}$–formulae.

Axioms of the logic **Ł** are listed below ([AT75, p.21]):

(L1) $\mathcal{B}_1 \to (\mathcal{B}_2 \to \mathcal{B}_1)$

(L2) $(\mathcal{B}_1 \to (\mathcal{B}_2 \to \mathcal{B}_3)) \to ((\mathcal{B}_1 \to \mathcal{B}_2) \to (\mathcal{B}_1 \to \mathcal{B}_3))$

(L3) $\neg\mathcal{A}_1 \to (\mathcal{A}_1 \to \mathcal{B}_1)$

(L4) $\neg\mathcal{A}_1 \to \neg(\mathcal{A}_1 \wedge \mathcal{B}_1)$

(L5) $(\mathcal{B}_1 \to \mathcal{B}_2) \to ((\neg\mathcal{B}_1 \to \mathcal{B}_2) \to \mathcal{B}_2)$

(L6) $\mathcal{B}_1 \to (\neg\mathcal{B}_2 \to \neg(\mathcal{B}_1 \to \mathcal{B}_2))$

(L7) $\mathcal{B}_1 \to (\mathcal{B}_2 \to \mathcal{B}_1 \wedge \mathcal{B}_2)$

(L8) $\mathcal{B}_1 \to \neg\neg\mathcal{B}_1$

(L9) $\neg\neg\mathcal{B}_1 \to \mathcal{B}_1$

(L10) $\mathcal{B}_1 \wedge \mathcal{B}_2 \to \mathcal{B}_1$

(L11) $\mathcal{B}_1 \wedge \mathcal{B}_2 \to \mathcal{B}_2$

(L12) $\mathcal{B}_1 \to \mathcal{B}_1 \vee \mathcal{B}_2$

(L13) $\mathcal{B}_2 \to \mathcal{B}_1 \vee \mathcal{B}_2$
(L14) $\neg\mathcal{B}_1 \vee \neg\mathcal{B}_2 \to \neg(\mathcal{B}_1 \wedge \mathcal{B}_2)$
(L15) $\neg\mathcal{B}_1 \wedge \neg\mathcal{B}_2 \to \neg(\mathcal{B}_1 \vee \mathcal{B}_2)$
(L16) $\neg(\mathcal{B}_1 \to \mathcal{B}_2) \to \mathcal{B}_1 \wedge \neg\mathcal{B}_2$
(L17) $\neg B_i \wedge B_i$

and a inference rule is

$$\text{(MP)}\ \frac{\vdash \mathcal{B}_1 \quad \vdash \mathcal{B}_1 \to \mathcal{B}_2}{\vdash \mathcal{B}_2}.$$

To prove some dependence results, we will need the following lemma:

Lemma 2.10.1 *In the logic* $\mathbf{L}$ *implication is transitive, that is:*

$$\varphi_1 \to \varphi_2, \varphi_2 \to \varphi_3 \vdash \varphi_1 \to \varphi_3$$

Proof:

(1) $(\varphi_2 \to \varphi_3) \to (\varphi_1 \to (\varphi_2 \to \varphi_3))$ (axiom (L1))
(2) $\varphi_2 \to \varphi_3$ (hypothesis)
(3) $\varphi_1 \to (\varphi_2 \to \varphi_3)$ ((MP) on (1) and (2))
(4) $(\varphi_1 \to (\varphi_2 \to \varphi_3)) \to ((\varphi_1 \to \varphi_2) \to (\varphi_1 \to \varphi_3))$ (axiom (L2))
(5) $(\varphi_1 \to \varphi_2) \to (\varphi_1 \to \varphi_3)$ ((MP) on (3) and (4))
(6) $\varphi_1 \to \varphi_2$ (hypothesis)
(7) $\varphi_1 \to \varphi_3$ ((MP) on (5) and (6))

□

2.10.2 Another system

Another system with desired properties is the system 3 presented in section 2.4.6 on page 20 and described in [Mor89]. In that system, the interpretation is given by:

$$xPy = \begin{cases} 2, & \text{if } x = 0 \text{ and } y = 0; \\ 3, & \text{if } x = 0 \text{ and } y = 1; \\ 3, & \text{if } x = 1 \text{ and } y = 0; \\ 2, & \text{if } x = 1 \text{ and } y = 1. \end{cases} \text{ and } xEy = \begin{cases} 1, & \text{if } x = 0 \text{ and } y = 0; \\ 3, & \text{if } x = 0 \text{ and } y = 1; \\ 3, & \text{if } x = 1 \text{ and } y = 0; \\ 1, & \text{if } x = 1 \text{ and } y = 1. \end{cases}$$

An axiomatization of the logic $\mathbf{C}_{0.1}$ of this system (the logic $\mathbf{C}_{0.2}$ is not suitable since the formula $\neg(xPx)$ is invalid there) contains axioms (C1)–(C12) of a paraconsistent logic $\mathbf{C}_1$, together with axioms (C13)–(C16). We will use a shorthand A^0 for $\neg(A \wedge \neg A)$. Axioms are:

(C1) $A \to (B \to A)$
(C2) $(A \to B) \to ((A \to (B \to C)) \to (A \to C))$
(C3) $A \wedge B \to A$
(C4) $A \wedge B \to B$
(C5) $A \to (B \to A \wedge B)$
(C6) $A \to A \vee B$
(C7) $B \to A \vee B$
(C8) $(A \to C) \to ((B \to C) \to (A \vee B \to C))$

(C9) $A \vee \neg A$

(C10) $\neg\neg A \to A$

(C11) $B^0 \to ((A \to B) \to ((A \to \neg B) \to \neg A))$

(C12) $A^0 \wedge B^0 \to ((A \wedge B)^0 \wedge (A \to B)^0 \wedge (A \vee B)^0)$

(C13) $(\neg A)^0$

(C14) $(A \wedge B)^0$

(C15) $(A \vee B)^0$

(C16) $(A \to B)^0$

and inference rules are (MP) and uniform substitution.

If we note that the axioms (C1) and (L1) are the same and that the axioms (C2) and (L2) are very similar, and because it is all we needed in the previous proof, we have the following lemma:

Lemma 2.10.2 *In the logic* $\mathbf{C}_{0.1}$ *implication is transitive, i.e.:*

$$\varphi_1 \to \varphi_2, \varphi_2 \to \varphi_3 \vdash \varphi_1 \to \varphi_3$$

Proof:

(1) $(\varphi_2 \to \varphi_3) \to (\varphi_1 \to (\varphi_2 \to \varphi_3))$	(axiom (C1))
(2) $\varphi_2 \to \varphi_3$	(hypothesis)
(3) $\varphi_1 \to (\varphi_2 \to \varphi_3)$	((MP) on (1) and (2))
(4) $(\varphi_1 \to \varphi_2) \to ((\varphi_1 \to (\varphi_2 \to \varphi_3)) \to (\varphi_1 \to \varphi_3))$	(axiom (C2))
(5) $\varphi_1 \to \varphi_2$	(hypothesis)
(6) $(\varphi_1 \to (\varphi_2 \to \varphi_3)) \to (\varphi_1 \to \varphi_3)$	((MP) on (4) and (5))
(7) $\varphi_1 \to \varphi_3$	((MP) on (3) and (6))

□

2.10.3 Dependences between formulae in those systems

In those systems we have the validness of all the axioms (W1)–(W5) as well as the validness of the formulae (7), (9), (10), (12), (14)–(18) and (20).

If we analyze proofs of lemmas of dependences and if we use the above–proved fact that the implication is transitive, we can readily state the following lemma:

Lemma 2.10.3 *In the systems* $\mathbf{L}$ *and* $\mathbf{C}_{0.1}$ *the following is true:*

(a) $(W4) \vdash (W3)$*;*

(b) $(W5) \vdash (7)$*;*

(c) $(9), (16), (17) \vdash (14)$*;*

(d) $(W3) \vdash (16), (17)$*;*

(e) $(10), (16), (17) \vdash (18)$*;*

(f) $(12), (16), (17) \vdash (20)$.

Proof:
(a) Same as the proof of lemma 2.9.4 **(b)** on page 33.
(b) Same as the proof of lemma 2.9.3 **(a)** on page 32.
(c) Same as the proof of lemma 2.9.3 **(c)** on page 32.
(d) Same as the proof of lemma 2.9.4 **(a)** on page 33.
(e) Same as the proof of lemma 2.1.4 **(c)** on page 14.
(f)

$xPy \wedge xPz \to \neg yP\neg x \wedge \neg zP\neg x$ (formula (16))
$\neg yP\neg x \wedge \neg zP\neg x \to (\neg y \vee \neg z)P\neg x$ (formula (12))
$(\neg y \vee \neg z)P\neg x \to \neg(y \wedge z)P\neg x$ (de Morgan's laws)
$\neg(y \wedge z)P\neg x \to xP(y \wedge z)$ (formula (17))
$xPy \wedge xPz \to xP(y \wedge z)$ (transitivity of implication)

□

After that, we are left with the following formulae:

($W^{**}1$) $xPy \to \neg(yPx)$

($W^{**}2$) $xPy \wedge yPz \to xPz$

($W^{**}3$) $(x \vee y)P(z \vee u) \rightleftarrows (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P(\neg x \wedge \neg y \wedge u)$

($W^{**}4$) $xPy \rightleftarrows (x \wedge \alpha)P(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P(y \wedge \neg\alpha)$, where α is a new variable.

($W^{**}5$) $xPy \wedge xPz \to xP(y \vee z)$

($W^{**}6$) $(x \vee y)Pz \to xPz \vee yPz$

($W^{**}7$) $xPz \wedge yPz \to (x \vee y)Pz$

($W^{**}8$) $(x \wedge y)Pz \to xPz \vee yPz$

2.11 Slightly misbehaving

All above–presented interpretation methods were based on the fact that the inner logic is a classical, binary one. We can also look at the problem from a different point of view, as Huber did in his dissertation ([Hub72]). Namely, for a fixed preference formula Φ, let m be the number of variables in Φ, and define a mapping $k(m)\colon \mathbf{N} \mapsto \mathbf{N}$ as

$$k(m) = \begin{cases} 3, & \text{if } m = 1; \\ m, & \text{if } m > 1 \text{ and odd}; \\ m+1, & \text{otherwise.} \end{cases}$$

He assigns positive values to variables using a function $v\colon Var \mapsto \{1, 2, \ldots, k(m)\}$ and expands it to propositional formulae: $v(x \wedge y) = \min(x, y)$, $v(x \vee y) = \max(x, y)$, and $v(\neg x) = k(m) + 1 - x$. Finally, he defines a function τ that maps preference and indifference relations to a set $\{\top, \bot\}$ in the following way:

$$\tau(xPy) = \begin{cases} \top, & \text{if } v(x) > v(y); \\ \bot, & \text{if } v(x) \leq v(y) \end{cases}$$

and

$$\tau(xIy) = \begin{cases} \top, & \text{if } v(x) = v(y); \\ \bot, & \text{if } v(x) \neq v(y) \end{cases}$$

The outer logic in his case is the classical one.

In this interpretation, as Huber showed, all von Wright's axioms, except (w5) and (w7), are valid (of course, Huber used this interpretation to investigate his own preference logic and this result is just a corollary).

We will try the similar approach with the difference that the outer logic is not the classical one, but one of the above–presented. We also have to change slightly his definition of the interpretation τ. We will use the following one:

$$\tau(xPy) = \begin{cases} a, & \text{if } v(x) = v(y); \\ b, & \text{if } v(x) > v(y); \\ c, & \text{if } v(x) < v(y). \end{cases}$$

where a, b and c are constants that are to be found. We will only consider models where all von Wright's axioms are valid.

2.11.1 Results

Unfortunately, results are not very spectacular. The table for all systems, except the system 5 is as follows:

System	a	b	c	Invalid formulae	Invalid converses
E_{fde}	2	2	2	21	–
	3	3	3	21	–
	4	4	4	21	–
system 1	1	1	1	21	–
	2	2	2	21	–
system 2	1	1	1	21	–
	2	2	2	21	–
system 3	2	2	2	21	–
	3	3	3	21	–
system 4.1	4	4	4	21	–
system 4.2	2	2	2	21	–
	4	4	4	21	–

and for the system 5 we have: if $a, b, c \in \{4, 5, 6\}$, then all axioms, all formulae (except (21)) and all converses are valid.

In other words, the interpretation becomes a trivial one and we do not get all[12] we are looking for.

2.11.2 Another E_{fde}–model

Of course, another question is what will happen if we choose E_{fde} for our inner logic as well. For that case we ran our program with interpretation[13]

$$xPy = \begin{cases} a, & \text{if } x = y; \\ b, & \text{if } x < y \text{ (in } \mathrm{E}_{fde} \text{ sense)}; \\ c, & \text{if } x > y \text{ (in } \mathrm{E}_{fde} \text{ sense)}; \\ d, & \text{otherwise.} \end{cases}$$

and found out that in all these interpretations the formulae (16), (17) and (20) are valid. This can also be viewed as a consequence of the following theorem:

Theorem 2.11.1 *The formulae* (16) *and* (17) *are valid in all interpretations of above–mentioned type, no matter what values a, b, c and d take.*

Proof: The proof is case–based:

[12] Actually, we get more, and that is the problem.

[13] We will say that $x < y$ in E_{fde} sense if implication $x \to y$ is true in E_{fde}.

(i) $(x = y)$ In that case $f(xPy) = f(\neg yP\neg x) = a$, so the formulae are valid.

(ii) $(x < y$ in E_{fde} sense) Using E_{fde} tables, we can easily see that the formula $x < y \rightleftarrows \neg y < \neg x$ is true, and so we have that $f(xPy) = f(\neg yP\neg x) = b$, which implies that our formulae are valid.

(iii) $(y < x$ in E_{fde} sense) The same as in previous case.

(iv) (x and y are incompatible in E_{fde} sense) It is only possible if $x = 2$, $y = 3$ or if $x = 3$, $y = 2$. But then we also have that $x = \neg x$ and $y = \neg y$, so the formulae are again valid. □

As we already know, this result is not very surprising. If we search for models for all von Wright's axioms, we get the following results:

System	a	b	c	d	Invalid formulae	Invalid converses
E_{fde}	2	2	2	2	–	–
	3	3	3	3	–	–
	4	4	4	4	21	–

In general case, exploring all interpretations will be very complicated and long–time job even with the help of computer as there are $4^{16} = 4294967296 \sim 4.3 \cdot 10^9$ different models.

Chapter 3

Logic of preference and modal logics

If we take a closer look at the definitions (D1) and (D2) from page 6 which von Wright gave in [vW72], we can hope that modal logics can be useful for describing the preference relation. Namely, the definition (D2) says:

> We say that s is preferred to t under circumstances C_i if and only if *some* C_i–world that is also a s–world is preferred to *some* C_i–world that is also a t–world, but *no* C_i–world that is also a t–world is preferred to *any* C_i–world that is also a s–world.

Furthermore, Halldén's system from section 1.3 on page 5 has an axiom $\Box(x \leftrightarrow y) \rightarrow xEy$, and since formula $\neg(xEy) \rightarrow xPy \vee yPx$ is valid, a simple manipulation of Halldén's axiom shows us that the following definition does make sense.

3.1 Definition of interpretations

Definition: The *modal interpretation of preference relation*, denoted by $P^{\Box}$, is defined as:

$$xP^{\Box}y \leftrightarrow \Diamond(x \wedge \neg y) \wedge \neg\Diamond(y \wedge \neg x)$$

The immediate result of this definition is a Kripke satisfiability relation $\Vdash$. We can define a Kripke model as (X, R, v), where $X \neq \emptyset$ is a set of possible worlds, $R \subseteq X^2$ is an accessibility relation and $v : X \times Var \mapsto \{\top, \bot\}$ is an assignment of truth values to atomic formulae at possible worlds. Following the definition of $P^{\Box}$, we immediately have:

$$w \Vdash xP^{\Box}y \text{ iff } (\exists w_1)(wRw_1 \wedge w_1 \Vdash x \wedge \neg y) \wedge (\forall w_2)(wRw_2 \rightarrow w_2 \nVdash y \wedge \neg x)$$

where for formulae that does not contain preference relation, the satisfiability is defined in a usual way. Since we do not want to specify modal logic, we will say nothing more about relation R.

3.1.1 Properties

From the definition of $P^{\Box}$, we readily have the following theorem:

Theorem 3.1.1 *For $P^{\Box}$, we have:*

(a) $\Vdash xP^{\Box}y \rightarrow \neg(yP^{\Box}x)$*;*

(b) $\not\Vdash \neg(yP^{\Box}x) \rightarrow xP^{\Box}y$ (formula (21));

(c) $\Vdash xP^{\Box}y \wedge yP^{\Box}z \rightarrow xP^{\Box}z$;

(d) $\not\Vdash xP^{\Box}z \rightarrow xP^{\Box}y \wedge yP^{\Box}z$ (formula (o2));

(e) $\Vdash xP^{\Box}y \leftrightarrow (x \wedge \neg y)P^{\Box}(y \wedge \neg x)$;

(f) $\Vdash xP^{\Box}y \leftrightarrow \neg yP^{\Box}\neg x$.

Proof:

(a) As we have $\neg(yP^{\Box}x) \leftrightarrow \neg\Diamond(y \wedge \neg x) \vee \Diamond(x \wedge \neg y)$, the formula is reduced to

$$\Diamond(x \wedge \neg y) \wedge \neg\Diamond(y \wedge \neg x) \rightarrow \neg\Diamond(y \wedge \neg x) \vee \Diamond(x \wedge \neg y)$$

which is obviously valid, and that proves **(a)**.

(b) Follows directly from the proof under **(a)**.

(c) In that case, the formula is reduced to:

$$\Diamond(x \wedge \neg y) \wedge \neg\Diamond(y \wedge \neg x) \wedge \Diamond(y \wedge \neg z) \wedge \neg\Diamond(z \wedge \neg y) \rightarrow \Diamond(x \wedge \neg z) \wedge \neg\Diamond(z \wedge \neg x)$$

This implication is false if antecedent is true and consequent false. That means that $\Diamond(x \wedge \neg y)$ and $\Diamond(y \wedge \neg z)$ are both true and that both $\Diamond(y \wedge \neg x)$ and $\Diamond(z \wedge \neg y)$ are false. Furthermore, $\Diamond(x \wedge \neg z)$ could not be false, since in that case we will have a contradiction as at least one of $x \wedge \neg y$, $y \wedge \neg z$ must be true in some world. Indeed, if for a world w_1 we have $v(w_1, x) = \top$, $v(w_1, y) = \bot$, then we must have $v(w_1, z) = \bot$ and that means that $v(w_1, x \wedge \neg z) = \top$, what is a contradiction, and if for a world w_2 we have $v(w_2, y) = \top$, $v(w_2, z) = \bot$, then we must have $v(w_2, x) = \top$ and that implies again that $v(w_2, x \wedge \neg z) = \top$.

So, $\Diamond(x \wedge \neg z)$ is true, and for consequent to be false we must force $\Diamond(z \wedge \neg x)$ to be true, and that means that in a world w_3 we have $v(w_3, z) = \top$, $v(w_3, x) = \bot$ and that is again a contradiction since both $\Diamond(y \wedge \neg x)$ and $\Diamond(z \wedge \neg y)$ are false.

That proves that our formula is valid and thus **(c)** is proved.

(d) In this case we have to prove that the converse of the formula of the proof of **(c)** is false:

$$\Diamond(x \wedge \neg z) \wedge \neg\Diamond(z \wedge \neg x) \rightarrow \Diamond(x \wedge \neg y) \wedge \neg\Diamond(y \wedge \neg x) \wedge \Diamond(y \wedge \neg z) \wedge \neg\Diamond(z \wedge \neg y)$$

One counter–model for this formula could be: $(\{w_1, w_2\}, ((w_1, w_2)), v)$, where $v(w_1, x) = v(w_2, x) = \top$, $v(w_1, y) = v(w_2, y) = v(w_1, z) = v(w_2, z) = \bot$.

(e) The proof is straightforward and follows directly from the definition of $P^{\Box}$.

(f) Follows directly from the definition of $P^{\Box}$ and the fact that $x \leftrightarrow \neg\neg x$.

□

In a similar way, we prove the following theorem:

Theorem 3.1.2 *For $P^{\Box}$, we have:*

(a) $\Vdash (x \wedge \alpha)P^{\Box}(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P^{\Box}(y \wedge \neg\alpha) \rightarrow xP^{\Box}y$, *where α is a new variable;*

(b) $\not\Vdash xP^{\Box}y \rightarrow (x \wedge \alpha)P^{\Box}(y \wedge \alpha) \wedge (x \wedge \neg\alpha)P^{\Box}(y \wedge \neg\alpha)$, *where α is a new variable* (formula (o6));

(c) $\Vdash (x \wedge \neg z \wedge \neg u)P^{\Box}(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P^{\Box}(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P^{\Box}(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P^{\Box}(\neg x \wedge \neg y \wedge u) \rightarrow (x \vee y)P^{\Box}(z \vee u)$;

(d) $\not\Vdash (x \vee y)P^{\Box}(z \vee u) \rightarrow (x \wedge \neg z \wedge \neg u)P^{\Box}(\neg x \wedge \neg y \wedge z) \wedge (x \wedge \neg z \wedge \neg u)P^{\Box}(\neg x \wedge \neg y \wedge u) \wedge (y \wedge \neg z \wedge \neg u)P^{\Box}(\neg x \wedge \neg y \wedge z) \wedge (y \wedge \neg z \wedge \neg u)P^{\Box}(\neg x \wedge \neg y \wedge u)$ (formula (o5));

Proof:

(a) After applying the definition, this formula is reduced to:

$\Diamond(x\wedge\alpha\wedge\neg y)\wedge\neg\Diamond(y\wedge\alpha\wedge\neg x)\wedge\Diamond(x\wedge\neg\alpha\wedge\neg y)\wedge\neg\Diamond(y\wedge\neg\alpha\wedge\neg x) \rightarrow \Diamond(x\wedge\neg y)\wedge\neg\Diamond(y\wedge\neg x)$

Let us prove this implication. The method is similar to the method used in the previous theorem. If we suppose that the implication is not true, we get that formula $\Diamond(y \wedge \neg x)$ must be false (because both $\Diamond(y \wedge \alpha \wedge \neg x)$ and $\Diamond(y \wedge \neg\alpha \wedge \neg x)$ are false, so formula $\Diamond(x \wedge \neg y)$ must also be false). But in that case formulae $\Diamond(x \wedge \alpha \wedge \neg y)$ and $\Diamond(x \wedge \neg\alpha \wedge \neg y)$ could not both be true, what is needed for the implication to be false. So, we obtained a contradiction, and that means that the implication is true.

(b) Follows directly from the proof in **(a)**. It is easy to construct a counter–model for this formula.

(c) Reducing the formula to a modal formula and using $\Diamond(x \vee y) \leftrightarrow \Diamond x \vee \Diamond y$ as well as a distributivity property, we only need to prove the following formula:

$\Diamond(x \wedge \neg z \wedge \neg u) \wedge \Diamond(y \wedge \neg z \wedge \neg u) \wedge \neg\Diamond(z \wedge \neg x \wedge \neg y) \wedge \neg\Diamond(u \wedge \neg x \wedge \neg y) \rightarrow (\Diamond(x \wedge \neg z \wedge \neg u) \wedge \neg\Diamond(z \wedge \neg x \wedge \neg y) \wedge \neg\Diamond(u \wedge \neg x \wedge \neg y)) \vee (\Diamond(y \wedge \neg z \wedge \neg u) \wedge \neg\Diamond(z \wedge \neg x \wedge \neg y) \wedge \neg\Diamond(u \wedge \neg x \wedge \neg y))$

and this is further reduced to the proof of the following formula of a propositional logic:

$$a \wedge b \wedge c \wedge d \rightarrow (a \wedge c \wedge d) \vee (b \wedge c \wedge d)$$

which is obviously true. That proves the implication.

(d) Follows directly from the proof under **(c)**, if we look at the modal translation of this formula. □

Checking other properties of relation $P^{\Box}$, we immediately have the following theorem:

Theorem 3.1.3 *For $P^{\Box}$, we have:*

- **(a)** $\Vdash xP^{\Box}y \rightarrow (x \wedge \alpha)P^{\Box}(y \wedge \alpha) \vee (x \wedge \neg\alpha)P^{\Box}(y \wedge \neg\alpha)$, *where α is a new variable* (formula (7));
- **(b)** $\Vdash xP^{\Box}(y \vee z) \rightarrow xP^{\Box}y \wedge xP^{\Box}z$ (formula (8));
- **(c)** $\nVdash xP^{\Box}y \wedge xP^{\Box}z \rightarrow xP^{\Box}(y \vee z)$ (formula (9));
- **(d)** $\Vdash xP^{\Box}z \vee yP^{\Box}z \rightarrow (x \vee y)P^{\Box}z$ (formula (11));
- **(e)** $\nVdash (x \vee y)P^{\Box}z \rightarrow xP^{\Box}z \vee yP^{\Box}z$ (formula (10));
- **(f)** $\Vdash xP^{\Box}z \wedge yP^{\Box}z \rightarrow (x \vee y)P^{\Box}z$ (formula (12));
- **(g)** $\nVdash (x \vee y)P^{\Box}z \rightarrow xP^{\Box}z \wedge yP^{\Box}z$ (formula (o12));

Proof:

(a) The modal translation of this formula gives the following formula:

$$\Diamond(x \wedge \neg y) \wedge \neg\Diamond(y \wedge \neg x) \rightarrow (\Diamond(x \wedge \alpha \wedge \neg y) \wedge \neg\Diamond(y \wedge \alpha \wedge \neg x)) \vee (\Diamond(x \wedge \neg\alpha \wedge \neg y) \wedge \neg\Diamond(y \wedge \neg\alpha \wedge \neg x))$$

Let us prove this formula. Suppose it is not true. That means that antecedent is true and that consequent is false. That means furthermore that formulae $\Diamond(x \wedge \alpha \wedge \neg y)$, $\Diamond(x \wedge \neg\alpha \wedge \neg y)$, $\Diamond(y \wedge \alpha \wedge \neg x)$, $\Diamond(y \wedge \neg\alpha \wedge \neg x)$ and $\Diamond(y \wedge \neg x)$ are all false, and that $\Diamond(x \wedge \neg y)$ is true. This is impossible, because it means that there is a world where $x \wedge \neg y$ is true and that in the same world both $x \wedge \alpha \wedge \neg y$ and $x \wedge \neg\alpha \wedge \neg y$ are false.

This proves our implication.

(b) The modal translation of this formula, using modal tautology $\Diamond(x \vee y) \leftrightarrow \Diamond x \vee \Diamond y$, gives us a formula

$$\Diamond(x \wedge \neg y \wedge \neg z) \wedge \neg\Diamond(y \wedge \neg x) \wedge \neg\Diamond(z \wedge \neg x) \rightarrow$$
$$\Diamond(x \wedge \neg y) \wedge \Diamond(x \wedge \neg z) \wedge \neg\Diamond(y \wedge \neg x) \wedge \neg\Diamond(z \wedge \neg x)$$

An analysis of this formula shows us that one of $\Diamond(x \wedge \neg y)$ and $\Diamond(x \wedge \neg z)$ must be false, which, together with the condition that $\Diamond(x \wedge \neg y \wedge \neg z)$ must be true leads to a contradiction. This proves our formula.

(c) For the other implication direction we need that modal formula $\Diamond(x \wedge \neg y) \wedge \Diamond(x \wedge \neg z) \rightarrow \Diamond(x \wedge \neg y \wedge \neg z)$ is true, what generally is not the case.

(d) The modal translation of this formula, together with properties of operator $\Diamond$, gives us a formula

$$(\Diamond(x \wedge \neg z) \wedge \neg\Diamond(z \wedge \neg x)) \vee (\Diamond(y \wedge \neg z) \wedge \neg\Diamond(z \wedge \neg y)) \rightarrow$$
$$(\Diamond(x \wedge \neg z) \vee \Diamond(y \wedge \neg z)) \wedge \neg\Diamond(z \wedge \neg x \wedge \neg y)$$

For this formula to be false, disjunction $\Diamond(x \wedge \neg z) \vee \Diamond(y \wedge \neg z)$ must be true, and so formula $\Diamond(z \wedge \neg x \wedge \neg y)$ must also be true. But this is impossible, since at least one of $\Diamond(z \wedge \neg x)$ and $\Diamond(z \wedge \neg y)$, according to hypothesis, must be false. That leads to a contradiction, and that proves our formula.

(e) In order to prove this, we need to prove that there is a model where the converse of the implication from proof under **(d)**) is not true. Such a model could be $(\{w_1, w_2, w_3\}, ((w_1, w_2), (w_1, w_3)), v)$ where v is given by: $v(w_1, x) = \top$, $v(w_1, y) = v(w_1, z) = \bot$, $v(w_2, x) = \top$, $v(w_2, y) = v(w_2, z) = \bot$, $v(w_3, x) = \bot$, $v(w_3, y) = v(w_3, z) = \top$.

That means that the formula (10) is not true,

(f) The modal translation of the formula (12) is

$$\Diamond(x \wedge \neg z) \wedge \Diamond(y \wedge \neg z) \wedge \neg\Diamond(z \wedge \neg x) \wedge \neg\Diamond(z \wedge \neg y) \rightarrow$$
$$(\Diamond(x \wedge \neg z) \vee \Diamond(y \wedge \neg z)) \wedge \neg\Diamond(z \wedge \neg x \wedge \neg y)$$

Suppose that this formula is false. That means that $\Diamond(x \wedge \neg z)$ and $\Diamond(y \wedge \neg z)$ are both true and that both $\Diamond(z \wedge \neg x)$ and $\Diamond(z \wedge \neg y)$ are false, what implies that $\Diamond(z \wedge \neg x \wedge \neg y)$ must be true (for implication to be false) which is impossible. That proves the implication and, at the same time, validness of the formula (12).

(g) The formula to be falsified is the converse of the implication from previous proof. One model where this formula is not true is $(\{w_1, w_2\}, ((w_1, w_2)), v)$ where v is given by: $v(w_1, x) = \bot$, $v(w_1, y) = \bot$, $v(w_1, z) = \bot$, $v(w_2, x) = \bot$, $v(w_2, y) = \top$, $v(w_2, z) = \bot$.

So, this implication is not true, and that means that the converse of the formula (12), i.e. the formula (o12), is also not true. □

We also have the following theorem:

Theorem 3.1.4 *For $P^{\Box}$, we have:*

(a) $\Vdash (x \wedge y)P^{\Box}z \rightarrow xP^{\Box}z \wedge yP^{\Box}z$ (formula (13));

(b) $\nVdash xP^{\Box}z \wedge yP^{\Box}z \rightarrow (x \wedge y)P^{\Box}z$ (formula (14));

(c) $\Vdash (x \wedge y)P^{\Box}z \rightarrow xP^{\Box}z \vee yP^{\Box}z$ (formula (15));

(d) $\nVdash xP^{\Box}z \vee yP^{\Box}z \rightarrow (x \wedge y)P^{\Box}z$ (formula (o15));

(e) $\Vdash xP^{\Box}(y \wedge z) \rightarrow xP^{\Box}y \vee xP^{\Box}z$ (formula (18));

(f) $\nVdash xP^{\Box}y \vee xP^{\Box}z \rightarrow xP^{\Box}(y \wedge z)$ (formula (19));

(g) $\Vdash xP^{\Box}y \wedge xP^{\Box}z \rightarrow xP^{\Box}(y \wedge z)$ (formula (20));

(h) $\nVdash xP^{\Box}(y \wedge z) \rightarrow xP^{\Box}y \wedge xP^{\Box}z$ (formula (o20));

Proof:

(a) The modal translation of this formula is

$$\Diamond(x\wedge y\wedge\neg z)\wedge\neg\Diamond(z\wedge\neg x)\wedge\neg\Diamond(z\wedge\neg y)\rightarrow\Diamond(x\wedge\neg z)\wedge\Diamond(y\wedge\neg z)\wedge\neg\Diamond(z\wedge\neg x)\wedge\neg\Diamond(z\wedge\neg y)$$

This implication is false if formula $\Diamond(x\wedge y\wedge\neg z)$ is true and at least one of $\Diamond(x\wedge\neg z)$ and $\Diamond(y\wedge\neg z)$ is false, and this is not possible.

(b) The proof is reduced to the proof that formula $\Diamond(x\wedge\neg z)\wedge\Diamond(y\wedge\neg z)\rightarrow\Diamond(x\wedge y\wedge\neg z)$ is not a theorem of modal logic, and a counter–model for that is easy to find.

(c) The proof is similar to the proof under **(a)**. In this case, we need to prove implication

$$\Diamond(x\wedge y\wedge\neg z)\wedge\neg\Diamond(z\wedge\neg x)\wedge\neg\Diamond(z\wedge\neg y)\rightarrow$$
$$(\Diamond(x\wedge\neg z)\wedge\neg\Diamond(z\wedge\neg x))\vee(\Diamond(y\wedge\neg z)\wedge\neg\Diamond(z\wedge\neg y))$$

This implication is false if $\Diamond(x\wedge y\wedge\neg z)$ is true and both $\Diamond(x\wedge\neg z)$ and $\Diamond(y\wedge\neg z)$ are false, what is impossible.

(d) Modal formula here is the converse of the formula in the proof under **(c)**, and it is easy to see that one counter–model for this is: $(\{w_1,w_2\},((w_1,w_2)),v)$ with v given by: $v(w_1,x)=\bot$, $v(w_1,y)=\bot$, $v(w_1,z)=\bot$, $v(w_2,x)=\top$, $v(w_2,y)=\bot$, $v(w_2,z)=\bot$.

(e) Modal translation gives us a formula

$$(\Diamond(x\wedge\neg y)\vee\Diamond(x\wedge\neg z))\wedge\neg\Diamond(y\wedge\neg x)\wedge\neg\Diamond(z\wedge\neg x)\rightarrow$$
$$(\Diamond(x\wedge\neg y)\wedge\neg\Diamond(y\wedge\neg x))\vee(\Diamond(x\wedge\neg z)\wedge\neg\Diamond(z\wedge\neg x))$$

and that reduces our proof to the proof that propositional formula $(a\vee b)\wedge c\wedge d\rightarrow(a\wedge c)\vee(b\wedge d)$ is a tautology, and that is evidently true.

So, the formula (18) is true.

(f) The proof follows directly from the previous proof as the converse of the above–presented propositional formula, i.e. formula $(a\wedge c)\vee(b\wedge d)\rightarrow(a\vee b)\wedge c\wedge d$ is not a tautology. Of course, we can easily make a Kripke counter–model for the respective modal formula.

(g) Modal translation of this formula is:

$$\Diamond(x\wedge\neg y)\wedge\neg\Diamond(y\wedge\neg x)\wedge\Diamond(x\wedge\neg z)\wedge\neg\Diamond(z\wedge\neg x)\rightarrow$$
$$(\Diamond(x\wedge\neg y)\vee\Diamond(x\wedge\neg z))\wedge\neg\Diamond(y\wedge\neg x)\wedge\neg\Diamond(z\wedge\neg x)$$

which is further reduced to propositional formula $a\wedge b\wedge c\wedge d\rightarrow(a\vee b)\wedge c\wedge d$, which is a tautology of propositional logic. That proves our implication.

(h) For converse of the formula (20) to be true, we need validness of the converse of the implication from the proof under **(g)** which is not the case since $a\vee b\rightarrow a\wedge b$ is (obviously) not a tautology of propositional logic. □

So, in our modal interpretation, from the list of formulae (1)–(21), the following are valid: (1)–(8), (11)–(13), (15)–(18) and (20).

3.2 Modal logic and indifference relation

If we introduce the indifference relation $I^{\Box}$ in the usual way, i.e. as $xI^{\Box}y\leftrightarrow\neg(xP^{\Box}y)\wedge\neg(yP^{\Box}x)$, using modal logic principles, we obtain:

$$xI^{\Box}y\leftrightarrow(\Diamond(x\wedge\neg y)\leftrightarrow\Diamond(y\wedge\neg x))$$

and its immediate consequence is the following theorem:

Theorem 3.2.1 *The following is true:*

- **(a)** $I^{\Box}$ *is reflexive;*
- **(b)** $I^{\Box}$ *is symmetric.*

But $I^{\Box}$ is not necessarily an equivalence relation as the following theorem shows:

Theorem 3.2.2 *$I^{\Box}$ is not transitive.*

Proof: The modal translation of the transitivity condition gives the following formula:

$$(\Diamond(x \wedge \neg y) \leftrightarrow \Diamond(y \wedge \neg x)) \wedge (\Diamond(y \wedge \neg z) \leftrightarrow \Diamond(z \wedge \neg y)) \rightarrow (\Diamond(x \wedge \neg z) \leftrightarrow \Diamond(z \wedge \neg x))$$

One counter–model for this formula is:

$$(\{w_0, w_1, w_2, w_3, w_4, w_5\}, ((w_0, w_1), (w_0, w_2), (w_0, w_3), (w_0, w_4), (w_0, w_5)), v)$$

where v is given by $v(w_0, x \wedge \neg z) = \bot$, $v(w_1, x) = v(w_1, z) = \top$, $v(w_1, y) = \bot$, $v(w_2, y) = \top$, $v(w_2, x) = \bot$, $v(w_3, x) = v(w_3, z) = \bot$, $v(w_3, y) = \top$, $v(w_4, y) = \bot$, $v(w_4, z) = \top$, $v(w_5, x) = \bot$ and $v(w_5, z) = \top$. Other (unspecified) variable assignments in v are arbitrary.

So, transitivity of relation $I^{\Box}$ need not hold. $\Box$

Chapter 4

Decision supporting systems

4.1 Simple example

The main application of preference relations should be in decision supporting systems. The idea is that a user gives his set of preferences, and the system decides between what alternatives the user should choose. The following simple example is from [KM75, pp.190–191]:

A bachelor has possibilities to spend the evening with Jane, to spend the evening with Helen or to watch television. He gave the following set of preferences:

(1^0) I prefer to spend the evening with Jane than to watch television;

(2^0) I prefer to spend the evening with Helen than to watch television;

(3^0) I prefer to be alone than to be with both Jane and Helen;

Formally, his list of preferences is (t is television, j is Jane and h is Helen):

$$jPt, \quad hPt, \quad \neg(j \wedge h)P(j \wedge h).$$

Applying von Wright's system, we get the following list of states:

$$(j \wedge \neg h \wedge \neg t), \quad (\neg j \wedge h \wedge \neg t), \quad (\neg j \wedge \neg h \wedge \neg t)$$

that is not to watch television in any case!

Of course, it is very hard (maybe even impossible) to design a decision supporting procedure that will give only one (i.e. the best) alternative — this procedure here is useful for *reducing* the set of alternatives (unfortunately sometimes to an empty set) the user faces (there were 8 alternatives in the example and the program eliminated 5 of them.).

4.2 The making of decision procedure

The main goal of this section is to make a procedure for deciding in the von Wright's original systems (as described in [vW63] and [vW72]). The input to the procedure is a list (set) of preferences (as in the above–presented example) and the output should be a ranked set of states of affairs, as explained in [vW63] and [vW72]. Contrary to the discussion in previous chapters, this procedure is for the case when outer logic is a classical logic as well.

4.2.1 Theoretic foundations

In order to construct a procedure as described above, we will need the following lemmas:

Lemma 4.2.1 *Let* $\{x_1, \ldots, x_n\}$ *be a list of* all *variables that appear in preference, and let* $\Phi_1 = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$, $\Phi_2 = x_1^{\beta_1} x_2^{\beta_2} \ldots x_n^{\beta_n}$ *and* $\Phi_3 = x_1^{\gamma_1} x_2^{\gamma_2} \ldots x_n^{\gamma_n}$, *where* $\alpha_i, \beta_i, \gamma_i \in \{0, 1\}$. *Then*

$$(\Phi_1 \vee \Phi_2) P \Phi_3 \leftrightarrow \Phi_1 P \Phi_3 \wedge \Phi_2 P \Phi_3.$$

Proof: In the proof we will need the following tautologies that are easy to prove:

$(*)$ $\quad (x \wedge (x \vee y)) \leftrightarrow x$

$(**)$ $\quad (x \wedge (\neg x \vee y)) \leftrightarrow (x \wedge y)$

Applying the axiom (W4) we have

$$(\Phi_1 \vee \Phi_2) P \Phi_3 \leftrightarrow (\Phi_1 \wedge \neg\Phi_3) P (\Phi_3 \wedge \neg\Phi_1 \wedge \neg\Phi_2) \wedge (\Phi_2 \wedge \neg\Phi_3) P (\Phi_3 \wedge \neg\Phi_1 \wedge \neg\Phi_2).$$

Further, we have:

$$\begin{aligned}
\Phi_1 \wedge \neg\Phi_3 &\leftrightarrow (x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}) \wedge (x_1^{1-\gamma_1} \vee x_2^{1-\gamma_2} \vee \ldots \vee x_n^{1-\gamma_n}) \\
&\leftrightarrow \begin{cases} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}, & \text{if } \alpha_1 = 1 - \gamma_1 \text{ by } (*) \\ (x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}) \wedge (x_2^{1-\gamma_2} \vee \ldots \vee x_n^{1-\gamma_n}), & \text{otherwise by } (**) \end{cases} \\
&\leftrightarrow \cdots \\
&\leftrightarrow \Phi_1
\end{aligned}$$

In a similar way we prove that $\Phi_2 \wedge \neg\Phi_3 \leftrightarrow \Phi_2$ as well as that $(\Phi_3 \wedge \neg\Phi_1 \wedge \neg\Phi_2) \leftrightarrow \Phi_3$. □

Lemma 4.2.2 *Let* $\{x_1, \ldots, x_n\}$ *be a list of* all *variables that appear in preference, and let* $\Phi_1 = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$, $\Phi_2 = x_1^{\beta_1} x_2^{\beta_2} \ldots x_n^{\beta_n}$ *and* $\Phi_3 = x_1^{\gamma_1} x_2^{\gamma_2} \ldots x_n^{\gamma_n}$, *where* $\alpha_i, \beta_i, \gamma_i \in \{0, 1\}$. *Then*

$$\Phi_1 P (\Phi_2 \vee \Phi_3) \leftrightarrow \Phi_1 P \Phi_2 \wedge \Phi_1 P \Phi_3.$$

Proof: Similar to the proof of lemma 4.2.1. □

The immediate consequence of those two lemmas is:

Corollary 4.2.1 *Let* $\{x_1, \ldots, x_n\}$ *be a list of* all *variables that appear in preference, and let* $\Phi_1 = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$, $\Phi_2 = x_1^{\beta_1} x_2^{\beta_2} \ldots x_n^{\beta_n}$, $\Phi_3 = x_1^{\gamma_1} x_2^{\gamma_2} \ldots x_n^{\gamma_n}$ *and* $\Phi_4 = x_1^{\delta_1} x_2^{\delta_2} \ldots x_n^{\delta_n}$, *where* $\alpha_i, \beta_i, \gamma_i, \delta_i \in \{0, 1\}$. *Then*

$$(\Phi_1 \vee \Phi_2) P (\Phi_3 \vee \Phi_4) \leftrightarrow \Phi_1 P \Phi_3 \wedge \Phi_1 P \Phi_4 \wedge \Phi_2 P \Phi_3 \wedge \Phi_2 P \Phi_4.$$

We also need the following theorem from [vW63] (see also [KM75]):

Theorem 4.2.1 *Suppose that* $\phi P \psi$, $\phi \leftrightarrow \phi_1$ *and* $\psi \leftrightarrow \psi_1$, *where* ϕ_1 *and* ψ_1 *do not contain any variable not already in* ϕ *and* ψ. *Then* $\phi_1 P \psi_1$.

4.2.2 Decision procedure

According to the lemmas and theorems from the previous section, the main job in constructing a decision–supporting procedure is to reduce propositional formulae, that are arguments of the preference relation, into perfect disjunctive normal form (i.e. the disjunction of conjunctions of *all* variables, or their negations, that appear in preferences). After that, we only need to apply the axiom (W2) and the lemmas and theorems above. The problem with this decision procedure is that we do not

want to prove if a system is consistent or not (i.e. to give an answer of the type Yes/No), what can be easily done — we want to generate all acceptable states of affairs for the given preference list.

The algorithm is as follows:

S1. Make a list V of *all* variables that appear in our preferences;

S2. Apply the axiom (W3) to every preference where a set of variables on the left side is disjoint from the set of variables on the right side;

S3. For every $x \in V$ that appears only on the one side of preference relation, put it also on the other side using tautology $\phi \leftrightarrow \phi \wedge (x \vee \neg x)$, and according to theorem 4.2.1. Repeat this step until all variables of this preference relation appear on both sides;

S4. If an argument of preference relation is not already in disjunctive normal form, transform it into disjunctive normal form. After that, apply lemmas 4.2.1 and 4.2.2 and corollary 4.2.1 to obtain conjunction of preferences whose arguments are only conjunctions of literals (i.e. of variables and their negations);

S5. If some variable from V is missing from some preference, apply axiom (W5) to add it to that preference;

S6. Apply axiom (W2) to the obtained set of preferences.

The output from this procedure are relations between worlds (i.e. states of affairs), and we can safely eliminate all worlds (out of 2^n for n variables) that are preferred by some other world.

Example: If we apply this procedure step by step to the 'Jane–or–Helen–or–TV' dilemma above, we get the following (we will write xy for $x \wedge y$ and $\bar{x}$ for $\neg x$, and will use the following shorthand: $1 \equiv jht$, $2 \equiv jh\bar{t}$, $3 \equiv j\bar{h}t$, $4 \equiv j\bar{h}\bar{t}$, $5 \equiv \bar{j}ht$, $6 \equiv \bar{j}h\bar{t}$, $7 \equiv \bar{j}\bar{h}t$ and $8 \equiv \bar{j}\bar{h}\bar{t}$):

- after step (S1), we have $V = \{j, h, t\}$;
- step (S2) transforms our set of preferences into $(j\bar{t})P(\bar{j}t)$, $(h\bar{t})P(\bar{h}t)$, $(\bar{j} \vee \bar{h})P(jh)$;
- step (S3) is not applicable;
- step (S4) first transforms formula $(\bar{j} \vee \bar{h})P(jh)$ into $(\bar{j}h \vee j\bar{h} \vee \bar{j}\bar{h})P(jh)$, and after this step our set of preferences is $(j\bar{t})P(\bar{j}t)$, $(h\bar{t})P(\bar{h}t)$, $(\bar{j}h)P(jh)$, $(j\bar{h})P(jh)$, $(\bar{j}\bar{h})P(jh)$;
- after step (S5), our set of preferences is: $2P5$, $4P7$, $2P3$, $6P7$, $5P1$, $6P2$, $3P1$, $4P2$, $7P1$, $8P2$.
- step (S6) adds the following preferences: $2P1$, $4P1$, $6P1$, $6P5$, $6P3$, $4P5$, $4P3$, $8P5$, $8P3$, $8P1$;

As states 1, 2, 3, 5 and 7 are preferred by some other states, we eliminate them, and the set of alternatives that remains is $\{4, 6, 8\}$ i.e. $\{j\bar{h}\bar{t}, \bar{j}h\bar{t}, \bar{j}\bar{h}\bar{t}\}$.

4.2.3 Remarks about the procedure

- Step S2 should be applied before other steps, otherwise we can obtain a contradiction as shown in the following example: if our preference is xPy and $V = \{x, y\}$ applying S2, we obtain $x\bar{y}P\bar{x}y$ and that is all we get, but if we apply other steps before S2, we will also get a preference $xyPxy$ that is a contradictory one.
- For the same reason, we should apply step S4 before step S5.
- Applying transitivity only at the very end of the procedure does not change the set of worlds we obtain. For instance, if our original preference set is xPy, yPz, we get the same result no matter when we apply transitivity. Indeed, rule S2 gives $x\bar{y}P\bar{x}y$, $y\bar{z}P\bar{y}z$, steps S3 and S4 are not applicable, step S5 adds $x\bar{y}zP\bar{x}yz$, $x\bar{y}\bar{z}P\bar{x}y\bar{z}$, $xy\bar{z}Px\bar{y}z$ and $\bar{x}y\bar{z}P\bar{x}\bar{y}z$, and transitivity (step S6) finally adds $xy\bar{z}P\bar{x}yz$ and $x\bar{y}\bar{z}P\bar{x}\bar{y}z$. We obtain the same result if we also apply transitivity at the very beginning, that is if we start from the set of preferences xPy, yPz and xPz.

4.2.4 Modifications for the extended system

As mentioned several times, von Wright's extended system was obtained from the original one by adding axiom $xPy \to xPz \vee zPy$. As dealing with disjunction of preferences is generally hard, we are going to apply this axiom in another form i.e. as $xPy \wedge \neg(xPz) \to zPy$ and as $xPy \wedge \neg(zPy) \to xPz$.

According to this remark, we modify our decision procedure by adding the following steps:

S3.1 If the left side of preference relation is equal to the left side of some negated preference, add the preference between the right side of the negated preference and the right hand side of the one considered.

S3.2 If the right side of preference relation is equal to the right side of some negated preference, add the preference between the left side of the considered preference and the left side of the negated one.

4.2.5 Modifications for indifference relation

According to von Wright, we have two different indifference relations, the weak one and the strong one. We can do nothing with the weak one, except to replace eventually every appearance of $\varphi I\psi$ by $\neg(\varphi P\psi) \wedge \neg(\psi P\varphi)$.

As for the strong one, we have a different situation: because of principles (e1) and (e2) from page 29, we can safely exchange in preferences states that are in relation E.

So, for those relations, we add another two steps to our procedure:

S1.1 Replace every appearance of formula $\varphi I\psi$ with $\neg(\varphi P\psi) \wedge \neg(\psi P\varphi)$;

S1.2 Apply the rule of symmetry to the relation E i.e. for every $\varphi E\psi$, add $\psi E\varphi$ to the set of preferences;

S1.3 Apply the same rules to the relation E as to the relation P;

as well as the following two steps:

S5.1 If the left side of relation E is equal to the left side of some preference, add to the set of preferences a preference whose left side is the right side of relation E, and the right side is the right side of the considered preference;

S5.2 If the left side of relation E is equal to the right side of some preference, add to the set of preferences a preference whose right side is the right side of relation E, and the left side is the left side of the considered preference;

4.3 Implementation

The above–described decision supporting procedure (prover) is implemented in `Common LISP` ([Fra88],[Ste90]) in a version `GC Lisp` for PC computers[1], but the version presented in appendixA also runs under Unix. Input to this prover is the name of the file where the set of preferences is given. Preferences are in `LISP`–form, i.e. they are of the form (P<left> <right>). Expressions <left> and <right> are also `LISP`–expressions, i.e. expressions in prefix notation. Logical connectives are `AND`, `OR` and `NOT`, where `AND` and `OR` could be of any arity. We also assume that every preference is of the above–specified type, i.e. that there are no operations between preferences. We will understand the set of preferences as the conjunction of them.

Another input parameter is the file that contains the list of all variables[2]. Variables are arbitrary words made from letters, but, as `LISP` does not make difference between upper– and lower–case letters except in strings, it means that variables `Jane`, `jane` and `JANE` (as well as the other 13 combinations) are all the same. This file also contains a filter–formula, i.e. a formula that is used to reduce further (to 'filtrate') the set of alternatives, as explained in [MKRT75]. Namely, for every alternative that remains, we check if the filter is also true in this world (i.e. valuation), and if not, we do not include this alternative into our final list of alternatives. If the user does not want to mess up with the filter, he can just leave that line empty and that will mean that the filter is always true.

During its work, the prover creates several temporary files and the final output is also written into a file.

An example: For the example presented in section 4.1 on page 46, input to our prover is
File `wishlist.prf` contains the following lines:

```
(P Jane television)
(P Helen television)
(P (NOT (AND Jane Helen)) (AND Jane Helen))
```

whereas file `vars.prf` contains the single line:

```
(Jane Helen television)
```

If we run our prover on this input, we will obtain the following set of worlds in the file `out.prf`[3]:

```
(AND JANE (NOT HELEN) (NOT TELEVISION))
(AND (NOT JANE) HELEN (NOT TELEVISION))
(AND (NOT JANE) (NOT HELEN) (NOT TELEVISION))
```

However, if we want to have at least one of the options above (i.e. Jane, Helen or TV), we add a filter (in notation from section 4.1) $(j \vee h \vee t)$, i.e. file `vars.prf` is:

[1] As the most of this thesis, the program was written when author was at the Computer Linguistic department at the University of Saarland, Saarbrücken, Germany, in period from March–September 1992, thanks to European Community stipend.

[2] In fact, this was not necessary, since it is possible to modify the prover to find this list itself, but in this way it makes life easier both to the program and to the programmer.

[3] Or in any file that user specified.

```
(Jane Helen television)
(OR Jane Helen television)
```

and the program output will be:

```
(AND JANE (NOT HELEN) (NOT TELEVISION))
(AND (NOT JANE) HELEN (NOT TELEVISION))
```

4.3.1 Program description

The program is split into several modules, each of them corresponding to one step in our procedure described in section 4.2.2 on page 47 as well as in the subsequent sections. Here we will only describe the most important of them, and the complete listing of the program is given in appendix A. The communication between modules is over files: every module reads in (the internal representation of) the result of the work of previous module, performs its function, and stores the result of its work in another file. The master program decides upon the names of intermediate files and takes care that all intermediate files are to be erased at the end of the final step. The version of the program described here only covers the basic version of the decision procedure, but, as the program is modularly written, it is relatively easy to add modules for the extended system and for the indifference relation.

`do-vars` Reads in a list of variables, makes pairs of variables and their internal representations and at the end replaces every variable with its internal representation. The input to this function is the name of the file that contains the list of variables.

`do-formulas` It performs the following task on all formulae from the input file: it splits them into the left hand part and the right hand part, replaces all variables with their internal representation and writes resulting formulae into the output file. The arguments to this function are the names of the input and the output file.

`step2` Applies the axiom (W3) to all formulae from the input file and stores the result into the output file. The arguments to this function are the names of the input and the output file.

`step3` Performs step 3 to all formulae of the input file, and stores the resulting formulae into the output file. The arguments to this function are the names of the input and the output file.

`step4` Performs step 4 to all formulae of the input file and stores the resulting formulae into the output file. This step is split into two subparts:

(1.) Transform formulae into a perfect disjunctive normal form (for the given list of variables). This task is performed by function `pdnf`, whose arguments are the input formula and the list of variables.

(2.) Apply lemmas and theorems from section 4.2.1 on page 47.

The arguments to this function are the names of the input and the output file.

`step5` Application of the axiom (W5). The arguments to this function are the names of the input and the output file.

`trans-clos` This function, for a given relation, finds its transitive closure ([Ben91]). Its input is a relation (as the list of ordered pairs of elements) whose closure is to be found, and the output is this closure.

`step6` This function reads from the input file ordered pairs of worlds (world = one conjunct in a perfect disjunctive normal form) that are related one with another, finds the transitive closure of this relation, and stores its result, as the list of ordered pairs, into the output file. The arguments to this function are the names of the input and the output file.

`postprocessing` This function eliminates all worlds that are on the right side of preference relation from the list of worlds and translates the given result from internal into original representation. The arguments to this function are the names of 3 files: the input one, the intermediate one and the output one.

`prover` This function is the master function of the whole prover: it communicates with the user, calls all of the above described modules and generates unique names for all intermediate files.

The function `prover` is as follows:

```
; procedure that performs the main job: it reads the file
; names, calls all modules and gives the results
(DEFUN prover (&AUX input-prf input-var res-prf file-t1
                    file-t2 file-t3 file-t4 file-t5 file-t6 file-t7)
  (SETQ input-prf (read-and-check "input"))
  (SETQ input-var (read-and-check "variables"))
  (TERPRI)
  (PRINC "The name of the output file:")
  (SETQ res-prf (READ-LINE))
  (TERPRI)
  (do-vars input-var)
  (MAKE-RANDOM-STATE)
  (SETQ file-t1 (unique-f-name))
  (do-formulas input-prf file-t1)
  (SETQ file-t2 (unique-f-name))
  (step2 file-t1 file-t2)
  (SETQ file-t3 (unique-f-name))
  (step3 file-t2 file-t3)
  (SETQ file-t4 (unique-f-name))
  (step4 file-t3 file-t4)
  (SETQ file-t5 (unique-f-name))
  (step5 file-t4 file-t5)
  (SETQ file-t6 (unique-f-name))
  (step6 file-t5 file-t6)
  (SETQ file-t7 (unique-f-name))
  (postprocessing file-t6 file-t7 res-prf)
  (delete-files *all-f-names*))
```

and we start it by `(prover)`.

Appendix A

Source program listing

A.1 File `variable.lsp`

```
;;; file variables.lsp

;; contains declarations for all global variables

(in-package "WRIGHT")
(DEFVAR *v-names* NIL "List of variable names.")
(DEFVAR *all-f-names* NIL "List of temporary file names.")
(DEFVAR *filter* NIL "Filter formula.")
(DEFVAR *prefvars* NIL "List of pairs (ord-number variable).")
(DEFVAR *all-worlds* NIL "List of all states of affairs.")
(DEFVAR *checked-contrad* NIL "Checked for a contradiction.")
(DEFVAR *contrad* NIL "Contains all contradictions found.")
(DEFVAR *dont-ask* NIL "If to ask about contradictions.")
(DEFVAR *left-side* NIL "Worlds on the left side of preferences.")
(DEFVAR *right-side* NIL "Worlds on the right side of preferences.")
(DEFVAR *allowed-worlds* NIL "Worlds that remained.")
```

A.2 File `include.lsp`

```
;;; file include.lsp

;; contains functions needed in all modules

(in-package "WRIGHT")

;; function my-error takes care of errors, give diagnostic
; message and exits the program
; code 1: file does not exist
; code 2: variable from formula is not in the list
; code 3: something happened to file from previous step
; code 4: some variable was forgotten before step 5
; code 5: there is a contradiction in preferences
; code 6: list of variables was empty
; code 7: could not create unique file name in 40 tries
(DEFUN my-error (code text)
  (TERPRI)
```

```
  (CASE code
      (1 (PRIN1 text) (PRINC "-- file does not exist"))
      (2 (PRIN1 text) (PRINC "-- var does not exist"))
      (3 (PRIN1 text) (PRINC "-- file vanished"))
      (4 (PRIN1 text) (PRINC "-- some vars are missing"))
      (5 (PRINC "There is a contradiction:")
         (DOLIST (item text)
            (PRINT (CONS 'P (LIST (in2var (CAR item))
                                  (in2var (CADR item)))))))
      (6 (PRIN1 text) (PRINC "--variable list is empty"))
      (7 (PRINC "Could not create file name")))
  (TERPRI)
  (PRINC "Prover error. Erasing files and aborting program!")
  (TERPRI)
  (delete-files *all-f-names*)
  (QUIT))

; function deletes all temporary files, their names are
; in the list that is suplied as argument
(DEFUN delete-files (llist)
  (DOLIST (i (REVERSE llist))
          (IF (PROBE-FILE i) (DELETE-FILE i))))

; function that prints a line and newline after that.
; built-in PRINT first print a newline, then text and
; after that one space
(DEFUN my-print (text file)
  (PRIN1 text file)
  (TERPRI file))

; this function looks if the file exists and if not
; it returns error message back
(DEFUN not-exists? (file-name code)
  (IF (NULL (PROBE-FILE file-name)) (my-error code file-name)))

; this function reads file name and check if it exists.
; if not it tries again
(DEFUN read-and-check(text &AUX res)
  (LOOP
   (FORMAT t "~%The name of the ~A file:" text)
   (SETQ res (READ-LINE))
   (IF (AND (> (LENGTH res) 0) (PROBE-FILE res))
       (RETURN res)
       (FORMAT t " The file ~A does not exist! Try again." res))))

;; general operations on the lists
;; set operations

; narrowing of the list
(DEFUN narrow (llist)
  (COND
   ((ATOM llist) llist)
   ((NULL llist) NIL)
   ((ATOM (CAR llist)) (CONS (CAR llist) (narrow (CDR llist))))
```

```
   (T (APPEND (narrow (CAR llist)) (narrow (CDR llist))))))

; removes elements listed in l1 from llist
; it first narrows the llist
(DEFUN rm-el-l (llist l1 &AUX res)
  (SETQ res (narrow llist))
  (DOLIST (i l1 res)
          (SETQ res (REMOVE i res))))

; remove duplicates from the list
(DEFUN rm-dp(llist)
  (REMOVE-DUPLICATES llist :test #'EQUAL))

; makes the set-union of two lists
(DEFUN my-union (list1 list2)
  (REMOVE-DUPLICATES (APPEND list1 list2) :test #'EQUAL))

; set difference of two lists
(DEFUN difference (list1 list2)
  (SET-DIFFERENCE list1 list2 :test #'EQUAL))

; set intersection of two lists, it also works on atoms
(DEFUN my-intersec (l1 l2)
  (COND
   ((OR (NULL l1) (NULL l2)) NIL)
   ((AND (ATOM l1) (ATOM l2)) (my-intersec (LIST l1) (LIST l2)))
   ((ATOM l1) (IF (MEMBER l1 l2 :test #'EQUAL) (LIST l1) NIL))
   ((ATOM l2) (IF (MEMBER l2 l1 :test #'EQUAL) (LIST l2) NIL))
   (T (INTERSECTION l1 l2 :test #'EQUAL))))

; Makes direct product of lists
(DEFUN mul (llist &AUX res)
  (SETQ res (CAR llist))
  (DOLIST (item (CDR llist))
          (SETQ res (REVERSE (product item res))))
  (REVERSE (MAPCAR #'REVERSE res)))

; makes direct product of two lists
(DEFUN product (list1 list2)
  (COND
   ((NULL list1) NIL)
   (T (APPEND (merge-el (CAR list1) list2)
              (product (CDR list1) list2)))))

(DEFUN merge-el (x llist)
  (COND
   ((NULL llist) NIL)
   (T (CONS
       (IF (ATOM (CAR llist)) (LIST x (CAR llist))
           (CONS x (CAR llist)))
       (merge-el x (CDR llist))))))

; assigns an ordered number to every member
(DEFUN ord-numb (llist)
```

```
  (rdbr 1 llist))

(DEFUN rdbr (n llist)
  (COND
   ((NULL llist) NIL)
   (T (CONS (LIST n (CAR llist)) (rdbr (+ 1 n) (CDR llist))))))

(DEFUN literal? (boolean) ; if literal
  (OR (poslit? boolean) (neglit? boolean)))

(DEFUN poslit? (boolean) ; if positive literal
  (ATOM boolean))

(DEFUN neglit? (boolean) ; if negative literal
  (AND (LISTP boolean)
       (EQ (CAR boolean) 'not)
       (poslit? (CADR boolean))))

; Sorts list of numbers in increasing order for abs.
; As SORT destroys its argument, we have to copy its contest
(DEFUN abs-sort (llist)
  (SORT (COPY-LIST llist) #'< :key #'abs))

; includes x in list so that list is sorted, if
; x is already in a list, does nothing
(DEFUN include-by-size (x llist)
  (COND
   ((NULL llist) (LIST x))
   ((ATOM llist) (include-by-size x (LIST llist)))
   (T (abs-sort (ADJOIN x llist :key #'ABS)))))

;;; pseudo-random-number generator, generates letters and returns
;;; a string of len such letters
(DEFUN random-string (len &AUX res)
  (SETQ res (LIST (+ 65 (RANDOM 26))))
  (DOTIMES (i (- len 1) res)
           (SETQ res (CONS (+ 65 (RANDOM 26)) res)))
  (FORMAT nil "~{~C~}" (MAPCAR #'CHARACTER res)))

;; find unique name for the file
; file is of (DOS) length 8+3, all random
; it adds its name into the list *all-f-names*
(DEFUN unique-f-name()
  (PROG (file-name i)
        (SETQ i 0)
        again
        (SETQ i (+ 1 i))
        (IF (> i 40) (my-error 7 NIL))
        (SETQ file-name
              (FORMAT nil "~A.~A"
                      (random-string 8) (random-string 3)))
        (IF (PROBE-FILE file-name) (GO again))
        (SETQ *all-f-names* (CONS file-name *all-f-names*))
        (RETURN file-name)))
```

```
; for a given world, translate it into a number
; from 0 to 2^number-of-variables -1
(DEFUN l-to-n (llist &AUX res j)
  (SETQ res 0
        j 1)
  (DOLIST (i (REVERSE llist) res)
          (IF (> i 0)
              (SETQ res (+ res j)))
          (SETQ j (* 2 j))))

; translates given number into a list of length len
(DEFUN n-to-l (n len &AUX res i j)
  (SETQ j len
        res NIL
        i n)
  (DOTIMES (i1 len res)
           (SETQ res (CONS (- 0 i1 1) res)))
  (LOOP
   (IF (ZEROP i) (RETURN (REVERSE res)))
   (IF (> (MOD i 2) 0)
       (SETQ res (SUBST  j (- j) res)))
   (SETQ i (FLOOR i 2)
         j (- j 1))))
```

A.3 File `step1.lsp`

```
;;; file step1.lsp

(in-package "WRIGHT")

; reads a line from the file, reads a list of variables,
; makes ordered pairs, and then replaces every variable with
; its internal representation
(DEFUN do-vars (file-name)
  (not-exists? file-name 1)
  (WITH-OPEN-FILE
   (si file-name :direction :input)
   (SETQ *v-names* (READ si NIL NIL))
   (IF (NULL *v-names*)
       (my-error 6 file-name))
   (SETQ *prefvars* (ord-numb *v-names*)
         *filter* (READ si NIL NIL))
   (IF (NULL *filter*) (SETQ *filter* T))))

; It finds index of variable in a list that is a list of
; ordered pairs where second element is a variable. The
; first element is 0, and function returns NIL if this
; variable is not in the list.
(DEFUN find-index (var vlist)
  (POSITION var vlist :key #'CADR))

; replace every variable in a formula with its number in the
```

```
; list. The list is a list of pairs (number variable)
(DEFUN repl-var (formula llist &AUX fff varlist ind replment)
  (SETQ fff (narrow formula)
        varlist (MAPCAR #'CADR llist))
  (COND
   ((ATOM formula)
    (CAR (NTH (find-index formula llist) llist)))
   (T (DOLIST (var1 fff)
              (COND
               ((MEMBER var1 varlist :test #'EQUAL)
                (SETQ ind (find-index var1 llist))
                (COND
                 ((NOT (NULL ind))
                  (SETQ replment (CAR (NTH ind llist))
                        formula (SUBST replment var1 formula)))))
               (T (COND
                   ((NOT (MEMBER var1 '(and or -) :test #'EQUAL))
                    (my-error 2 var1))))))
      formula)))

;; It works on all formulas from the given file: it splits
;; it into left and right part, it substitutes variables and
;; stores result into another file.
(DEFUN do-formulas (fname-1 fname-2)
  (not-exists? fname-1 1)
  (WITH-OPEN-FILE
   (so fname-2 :direction :output)
   (WITH-OPEN-FILE
    (si fname-1 :direction :input)
   (DO* ((formula (READ si NIL NIL) (READ si NIL NIL)))
        ((NULL formula) NIL)
        (my-print
         (LIST
          (repl-var (SUBST '- 'not (CADR formula)) *prefvars*)
          (repl-var (SUBST '- 'not (CADDR formula)) *prefvars*))
         so)))))
```

A.4 File `step2.lsp`

```
;;; file step2.lsp

(in-package "WRIGHT")

; formula W3 is applied if left and right side of the
; relation have no variables in common, otherwise don't apply
; it (no need for). It returns T if lists of variables are
; disjoint, NIL otherwise
(DEFUN crit-W3 (left right)
   (NULL (my-intersec (narrow left) (narrow right))))

; as literals are numbers, complementing it just changes a
; sign of it
(DEFUN complement (literal)
```

```
  (- literal))

; de Morgan's laws, negates formula
(DEFUN de-morgan (boolean)
  (COND
   ((literal? boolean) (complement boolean))
   ((CASE (CAR boolean)
          (AND (CONS 'or (MAPCAR #'de-morgan (CDR boolean))))
          (OR (CONS 'and (MAPCAR #'de-morgan (CDR boolean))))
          (- (CADR boolean))))))

; application of axiom W3, list = ((left) (right)) and that
; corresponds to left P right
(DEFUN apply-W3 (left right)
  (LIST (CONS 'and (LIST left (de-morgan right)))
        (CONS 'and (LIST right (de-morgan left)))))

; perform step 2 (axiom (W3) on input file
(DEFUN step2 (fname1 fname2 &AUX left right a-new)
  (not-exists? fname1 3)
  (WITH-OPEN-FILE
   (so fname2 :direction :output)
   (WITH-OPEN-FILE
    (si fname1 :direction :input)
    (DO* ((formula (READ si NIL NIL) (READ si NIL NIL)))
         ((NULL formula) NIL)
         (SETQ left  (CAR formula)
               right (CADR formula))
         (IF (crit-w3 left right)
             (SETQ a-new (apply-W3 left right))
             (SETQ a-new (LIST left right)))
         (my-print a-new so)))))
```

A.5 File step3.lsp

```
;;; file step3.lsp

(in-package "WRIGHT")

;; step 3 is applied if the list of variables on one side
;; is different from the list of variables on the other side

; finds all variables that are on the left hand side and not
; on the right hand side.
(DEFUN only-on-one (lft rht &AUX l d)
  (COND
   ((ATOM lft) (SETQ l (LIST lft))
    (IF (ATOM rht)
        (SETQ d (LIST rht))
        (SETQ d (rm-dp (MAPCAR #'ABS (rm-el-l rht '(- and or)))))))
   ((ATOM rht) (SETQ d (LIST rht))
    (IF (ATOM lft)
        (SETQ l (LIST lft))
```

```
        (SETQ l (rm-dp (MAPCAR #'ABS (rm-el-l lft '(- and or)))))))
   (T (SETQ l (rm-dp (MAPCAR #'ABS (rm-el-l lft '(- and or)))))
      (SETQ d (rm-dp (MAPCAR #'ABS (rm-el-l rht '(- and or)))))))
  (difference l d))

; adds a variable: formula goes to (and formula (or x (not x)))
; where x is that new variable
(DEFUN add-var (formula v1)
  (CONS 'and (LIST formula (CONS 'or (LIST v1 (complement v1))))))

; adds to formula all variables from the list
(DEFUN completev (formula llist)
  (COND
   ((NULL llist) formula)
   (T (DOLIST (var1 llist formula)
              (SETQ formula (add-var formula var1))))))

(DEFUN step3 (fname1 fname2 &AUX left right o-left o-right)
  (not-exists? fname1 3)
  (WITH-OPEN-FILE
   (so fname2 :direction :output)
   (WITH-OPEN-FILE
    (si fname1 :direction :input)
    (DO* ((formula (READ si NIL NIL) (READ si NIL NIL)))
         ((NULL formula) NIL)
         (SETQ left (CAR formula)
               right (CADR formula)
               o-left (only-on-one left right)
               o-right (only-on-one right left))
         (my-print (LIST (completev left o-right)
                         (completev right o-left)) so)))))
```

A.6 File `step4.lsp`

```
;;; file step4.lsp

(in-package "WRIGHT")

;; we have to do two different things: 1. translate formulas
;; into perfect disjunctive normal form (for given list of
;; variables, and 2. apply lemmas and theorems of
;; distributivity

; formulas of the kind (or f1 (or f2 f3) ...)  translates
; into (or f1 f2 f3 ...)
(DEFUN or-up (fla)
  (COND
   ((EQUAL (CAR fla) 'or) (move-or fla))
   (T fla)))

(DEFUN move-or (fla &AUX res)
  (SETQ res (LIST 'or))
  (DOLIST (item (CDR fla) res)
```

```
          (COND
           ((ATOM item) (SETQ res (APPEND res (LIST item))))
           (T (COND
               ((EQUAL (CAR item) 'or)
                (SETQ res (APPEND res (CDR (or-up item)))))
               (T (SETQ res (APPEND res (LIST item))))))))))

; function transforms list (((x1 x2 ...)) ... ((y1 y2 ...)))
; into ((x1 x2 ...) ... (y1 y2 ...))
(DEFUN level-up (llist)
  (MAPCAR #'narrow llist))

; transforms formula into dnf formula. Arguments of the
; formula are numbers and negation is already replaced with
; -, here we will replace positive numbers with negative
(DEFUN dnf (fla)
  (IF (literal? fla) (LIST fla)
      (CASE (CAR fla)
            (- (dnf (de-morgan (CADR fla))))
            (or (level-up (MAPCAR #'dnf (CDR (or-up fla)))))
            (and (MAPCAR #'rm-dp
                         (MAPCAR #'narrow
                                 (mul (MAPCAR #'dnf (CDR fla))))))))))

; for formula in dnf (i.e without operators) finds the list
; of all variables
(DEFUN l-of-vars (fla)
  (rm-dp (MAPCAR #'ABS (rm-el-l fla '(and or)))))

; adds all combinations of new variables into formula
; (completion to perfect DNF)
(DEFUN pdnf-compl (fla vlist &AUX var1 left1 left2)
  (COND
   ((NULL vlist) fla)
   (T (SETQ var1 (CAR vlist)
            left1 (include-by-size var1 fla)
            left2 (include-by-size (- var1) fla))
      (LIST (pdnf-compl left1 (CDR vlist))
            (pdnf-compl left2 (CDR vlist))))))

; The list of the form (((x y) (a b)) ((p q) (r s)))
; transforms into ((x y) (a b) (p q) (r s)). More general:
; binary tree transforms into linear list. More general:
; arbitrary tree transforms into linear list
(DEFUN tree-to-list (llist &AUX res)
  (COND
   ((NULL llist) NIL)
   ((ATOM (CAR llist)) (LIST llist))
   (T (SETQ res NIL)
      (DOLIST (item llist res)
              (SETQ res (APPEND res (tree-to-list item)))))))

; function transforms formula into PDNF with respect to the
; list of added variables that is the second argument to the
```

```
; function.  formula is already in DNF, we only  add
; new variables.  The principle is the same as in step 5
; (with small modifications).
(DEFUN pdnf (fla list-var1 &AUX res f1)
  (SETQ res NIL)
  (IF (ATOM fla)
      (SETQ f1 (LIST fla))
      (SETQ f1 fla))
  (DOLIST (item f1 (tree-to-list res))
          (SETQ res (APPEND res (pdnf-compl item list-var1)))))

; it takes formulas and transform them into PDNF, and
; finally, step 4 and lemmas and theorems of distributivity
(DEFUN step4 (inp-file out-file &AUX left right var1)
  (not-exists? inp-file 3)
  (WITH-OPEN-FILE
   (so out-file :direction :output)
   (WITH-OPEN-FILE
    (si inp-file :direction :input)
    (DO* ((fla (READ si NIL NIL) (READ si NIL NIL)))
         ((NULL fla) NIL)
         (SETQ left (dnf (CAR fla))
               right (dnf (CADR fla))
               var1 (my-union (l-of-vars left) (l-of-vars right))
               left (rm-dp (pdnf left var1))
               right (rm-dp (pdnf right var1)))
         (DOLIST (a-new (product left right))
                 (my-print (LIST (CAR a-new) (CDR a-new)) so))))))
```

A.7 File `step5.lsp`

```
;;; file step5.lsp

(in-package "WRIGHT")

;; Application of axiom W5 i.e. adding variables.  In this
;; stage, formulas are already simple conjunctions
;; represented as simple list of numbers with sign - if
;; variable is negated.  Also, the list of variables on the
;; left hand side and the right hand side is the same,
;; thanks to previous steps.

; returns the list of variables not in this formula
(DEFUN list-to-add (formula)
  (difference
   (MAPCAR #'CAR *prefvars*)
   (rm-dp (MAPCAR #'ABS (narrow formula)))))

; adds all possible combinations of variables in a formula
; and stores the resulting formula into output file
(DEFUN completev-w5 (fla vlist out-file &AUX left right
                                 var1 left1 left2 right1 right2)
  (COND
```

```
   ((NULL vlist) (my-print (MAPCAR #'l-to-n (MAPCAR #'abs-sort fla))
                           out-file))
   (T (SETQ left (CAR fla)
            right (CADR fla)
            var1 (CAR vlist)
            left1 (include-by-size var1 left)
            right1 (include-by-size var1 right)
            left2 (include-by-size (- var1) left)
            right2 (include-by-size (- var1) right))
      (completev-w5 (LIST left1 right1) (CDR vlist) out-file)
      (completev-w5 (LIST left2 right2) (CDR vlist) out-file))))

; step 5
(DEFUN step5 (inp-file out-file &AUX addsl addsr)
  (not-exists? inp-file 3)
  (WITH-OPEN-FILE
   (so out-file :direction :output)
   (WITH-OPEN-FILE
    (si inp-file :direction :input)
    (DO* ((fla (READ si NIL NIL) (READ si NIL NIL)))
         ((NULL fla) NIL)
         (SETQ addsl (list-to-add (CAR fla))
               addsr (list-to-add (CADR fla)))
         (IF (NOT (EQUAL addsl addsr)) (my-error 4 fla))
         (completev-w5 fla addsl so)))))
```

A.8 File step6.lsp

```
;;; file step6.lsp

(in-package "WRIGHT")

;; stuff related to step 6, i.e. transitive closure of a
;; given relation

; makes an array out of relation
(DEFUN rel-to-arr (relation nmr-vars &AUX a m)
  (SETQ m (EXPT 2 nmr-vars)
        a (MAKE-ARRAY (LIST m m) :element-type '(MOD 2)
                      :initial-element 0))
  (DOLIST (it relation a)
          (SETF (APPLY #'BIT a it) 1)))

; makes a relation out of an array
(DEFUN arr-to-rel (my-arr &AUX res d2)
  (SETQ d2 (CADR (ARRAY-DIMENSIONS my-arr))
        res NIL)
  (DOTIMES (i (CAR (ARRAY-DIMENSIONS my-arr)))
           (DOTIMES (j d2)
                    (IF (= 1 (BIT my-arr i j))
                        (SETQ res (CONS (LIST i j) res)))))
  res)
```

```
; defines closure of given relation, where relation is
; represented as a list of ordered pairs of numbers.
; Warshall's algorithm.
(DEFUN tr-rel (relation &AUX arr2 n tmpb-ik ai ak ta m)
  (SETQ arr2 (rel-to-arr relation (LENGTH *v-names*))
        n (CAR (ARRAY-DIMENSIONS arr2)))
  (DOTIMES (k n)
           (SETQ ak (MAKE-ARRAY n :displaced-to arr2
                                  :displaced-index-offset (* k n)
                                  :element-type '(MOD 2)))
           (DOTIMES (i n)

                    (SETQ tmpb-ik (BIT arr2 i k))
                    (SETQ m (* i n)
                          ai (MAKE-ARRAY n :displaced-to arr2
                                           :displaced-index-offset m
                                           :element-type '(MOD 2))
                          ta (MAKE-ARRAY n :element-type '(MOD 2)
                                           :initial-element tmpb-ik))
                    (BIT-IOR ai (BIT-AND ta ak NIL) T)))
  (arr-to-rel arr2))

; This function reads in the pairs of worlds from the file
; inp-file, finds transitive closure and stores the result, as
; the list of pairs in the file out-file
(DEFUN step6 (inp-file out-file &AUX relation)
  (not-exists? inp-file 3)
  (SETQ relation NIL)
  (WITH-OPEN-FILE
   (si inp-file :direction :input)
   (DO* ((formula (READ si NIL NIL) (READ si NIL NIL)))
        ((NULL formula) NIL)
        (SETQ relation (ADJOIN formula relation :test #'EQUAL))))
  (SETQ relation (tr-rel relation))
  (WITH-OPEN-FILE
   (si out-file :direction :output)
   (DOLIST (item relation)
           (my-print item si))))
```

A.9 File `postproc.lsp`

```
;;; file postproc.lsp

(in-package "WRIGHT")

; makes the list of all variables from one (left or right)
; side of relation.
(DEFUN which-side (inp-file side &AUX res operation tmpv ip tf)
  (SETQ res NIL)
  (IF (EQUAL side 'left)
      (SETQ operation 'car)
      (SETQ operation 'cadr))
  (WITH-OPEN-FILE
```

```
   (si inp-file :direction :input)
   (DO* ((formula (READ si NIL NIL) (READ si NIL NIL)))
        ((NULL formula) NIL)
        (SETQ tf (EQUAL (CAR formula) (CADR formula)))
        (IF (AND (NULL *checked-contrad*)
                 (NULL *dont-ask*) tf)
            (PROGN
             (SETQ *dont-ask* T)
             (FORMAT t "~%Contradiction found! ~
                       Do you want to ignore it? (Y/N) ")
             (SETQ ip (READ))
             (IF (EQUAL ip 'Y)
                 (SETQ *checked-contrad* T))))
        (IF (AND (NULL *checked-contrad*) tf)
            (SETQ *contrad* (APPEND *contrad* (LIST formula))))
        (SETQ tmpv (APPLY operation (LIST formula)))
        (IF (NOT (MEMBER tmpv res :test #'EQUAL))
            (SETQ res (APPEND res (LIST tmpv))))))
  (IF (NULL *checked-contrad*) (SETQ *checked-contrad* T))
  res)

; all worlds on the left hand side
(DEFUN left-side (inp-file)
  (which-side inp-file 'left))

; all worlds on the right hand side
(DEFUN right-side (inp-file)
  (which-side inp-file 'right))

; check if there is a contradiction
(DEFUN contradiction? ()
  (NOT (NULL *contrad*)))

; function generates all possible worlds
(DEFUN all-worlds (&AUX res)
  (SETQ res NIL)
  (DOTIMES (i (EXPT 2 (LENGTH *v-names*)) res)
           (SETQ res (CONS i res))))

; worlds that are allowed, i.e. all except those on the right
; hand side of any preference
(DEFUN allowed-worlds()
  (difference *all-worlds* *right-side*))

; function translates internal representation into
; the original one.
(DEFUN in2var (formula &AUX r)
  (SETQ r (n-to-l formula (LENGTH *v-names*)))
  (DOLIST (itm r r) ; the first r is evaluated before loop!
          (IF (< itm 0)
              (SETQ r (SUBST (CONS 'not (LIST (- itm))) itm r))))
  (DOLIST (itm *prefvars* r)
          (SETQ r (SUBST (CADR itm) (CAR itm) r)))
  (CONS 'and r))
```

```
; we want to evaluate formula given a valuation of its
; variables. We need it to implement filter possibility.
; we assume that valuation is a list of ordered pairs
; (value var-name), and that formula is a wff
; (with and, or, not))
(DEFUN valuate (formula valuation &AUX res)
  (SETQ res formula)
  (DOLIST (item valuation (eval res))
          (SETQ res (SUBST (CAR item) (CADR item) res))))

; from *allowed-worlds* makes a list of valuations of above type
; argument world is one state of affairs
(DEFUN make-value (world &AUX r w)
  (SETQ r NIL)
  (DOLIST(vv world r)
         (IF (< vv 0) (SETQ w NIL) (SETQ w T))
         (SETQ r (CONS (LIST w (NTH (- (abs vv) 1) *v-names*)) r))))

; This function applies above procedure to all formulas in
; the file
(DEFUN postprocessing (inp-file inter out-file &AUX len)
  (not-exists? inp-file 3)
  (SETQ *checked-contrad* NIL
        *contrad* NIL
        *left-side* (left-side inp-file)
        *right-side* (right-side inp-file))
  (IF (contradiction?) (my-error 5 (rm-dp *contrad*)))
  (SETQ *all-worlds* (all-worlds)
        *allowed-worlds* (allowed-worlds)
        len (LENGTH *v-names*))
  (WITH-OPEN-FILE
   (so inter :direction :output)
   (DOLIST (item *allowed-worlds*)
           (IF (valuate *filter* (make-value (n-to-l item len)))
               (my-print item so))))
  (WITH-OPEN-FILE
   (so out-file :direction :output)
   (WITH-OPEN-FILE
    (si inter :direction :input)
   (DO* ((formula (READ si NIL NIL) (READ si NIL NIL)))
        ((NULL formula) NIL)
        (my-print (in2var formula) so)))))
```

A.10 File `main.lsp`

```
;;; file main.lsp
(in-package "WRIGHT")

; This functions sets to nil all global variables that were
; used in program. This is necessary for memory savings if
; we run the program again
(DEFUN erase-global ()
```

```
  (SETQ *v-names* NIL
        *all-f-names* NIL
        *filter* NIL
        *prefvars* NIL
        *all-worlds* NIL
        *checked-contrad* NIL
        *contrad* NIL
        *dont-ask* NIL
        *left-side* NIL
        *right-side* NIL
        *allowed-worlds* NIL))

; reads all program files necessary for the program
(DEFUN load-all ()
  (LOAD "variable.lsp" :verbose NIL)
  (LOAD "include.lsp" :verbose NIL)
  (LOAD "step1.lsp" :verbose NIL)
  (LOAD "step2.lsp" :verbose NIL)
  (LOAD "step3.lsp" :verbose NIL)
  (LOAD "step4.lsp" :verbose NIL)
  (LOAD "step5.lsp" :verbose NIL)
  (LOAD "step6.lsp" :verbose NIL)
  (LOAD "postproc.lsp" :verbose NIL))

; procedure that performs the main job: it reads the file
; names, calls all modules and gives the results
(DEFUN prover (&AUX input-prf input-var res-prf file-t1
                    file-t2 file-t3 file-t4 file-t5 file-t6 file-t7)
  (SETQ input-prf (read-and-check "input")
        input-var (read-and-check "variables"))
  (TERPRI)
  (PRINC "The name of the output file:")
  (SETQ res-prf (READ-LINE))
  (TERPRI)
  (do-vars input-var)
  (MAKE-RANDOM-STATE)
  (SETQ file-t1 (unique-f-name))
  (do-formulas input-prf file-t1)
  (SETQ file-t2 (unique-f-name))
  (step2 file-t1 file-t2)
  (SETQ file-t3 (unique-f-name))
  (step3 file-t2 file-t3)
  (SETQ file-t4 (unique-f-name))
  (step4 file-t3 file-t4)
  (SETQ file-t5 (unique-f-name))
  (step5 file-t4 file-t5)
  (SETQ file-t6 (unique-f-name))
  (step6 file-t5 file-t6)
  (SETQ file-t7 (unique-f-name))
  (postprocessing file-t6 file-t7 res-prf)
  (delete-files *all-f-names*))

(DEFUN main ()
  (PROG (again)
```

```
        (load-all)
        label-main
        (erase-global)
        (prover)
        (PRINC "Again? (Y/N) ")
        (SETQ again (READ))
        (IF (EQUAL again 'Y)
            (PROGN
             (erase-global)
             (GC)
             (GO label-main)))))

; do the execution
(in-package "WRIGHT")
(main)
(quit)
```

Bibliography

[AB75] A. R. Anderson and N. D. Belnap, Jr. *Etailment. The Logic of Relevance and Necessity.* Vol I. Princeton University Press, 1975.

[AT75] F. G. Asenjo and J. Tamburino. Logic of antinomies. *Notre Dame Journal of Formal Logic*, 16:17–44, 1975.

[BB92] L. Bolc and P. Borowik. *Many–Valued Logics 1. Theoretical Foundations.* Springer Verlag, 1992.

[Ben91] Stefan Benzschawel. Transitive closure: New aspects of old theme. Technical Report IBWS Report 203, Wissenschaftliches Zentrum, IWBS, IBM Deutschland, 1991.

[BR89] R. T. Brady and R. Routley. The non–triviality of extensional dialectical set theory. In Priest et al. [PRN89], pages 415–436.

[Car85] Walter A. Carnielli. An algorithm for axiomatizing and theorem proving in finite many–valued propositional logics. *Logique et Analyse*, 28:363–368, 1985.

[Chi64] Roderik M. Chisholm. The descriptive element in the concept of action. *The Journal of Philosophy*, 61:613–625, 1964.

[CS66a] R. M. Chisholm and E. Sosa. Intrinsic preferability and the problem of supererogation. *Synthese*, 16:321–331, 1966.

[CS66b] R. M. Chisholm and E. Sosa. On the logic of "intrinsically better". *American Philosophical Quarterly*, 3:244–249, 1966.

[Dan68] Sven Danielsson. *Preference and Obligation. Studies in the Logic of Ethics.* Filosofiska Föringen, Uppsala, 1968.

[dC74] Newton C. A. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15:497–510, 1974.

[dCA77] N. C. A. da Costa and E. H. Alves. A semantical analysis of the calculi $\mathbf{C}_n$. *Notre Dame Journal of Formal Logic*, 18:621–630, 1977.

[Dun86] J. Michael Dunn. Relevance logic and entailment. In Gabbay and Günthner [GG86], pages 117–224.

[Fra88] Franc Inc. *Common LISP. The Reference.* Addison Wesley, 1988.

[GG86] D. Gabbay and F. Günthner, editors. *Handbook of Philosophical Logic* Vol III*: Alternatives to Classical Logic.* D. Reidel, 1986.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Haa88] Susan Haack. *Deviant Logic. Some Philosophical Issues.* Cambridge University Press, 1988.

[Hal66] Sören Halldén. Preference logic and theory choice. *Synthese*, 16:307–320, 1966.

[Han68] Bengt Hannson. Fundamental axioms for preference relations. *Synthese*, 18:423–442, 1968.

[Han70] Bengt Hannson. *Preference Logic: Philosophical Foundations and Applications in the Philosophy of Science.* Lund, 1970.

[HC68] G. E. Huges and M. J. Cresswell. *An Introduction to Modal Logics.* Methuen and Co. Ltd., London, 1968.

[Hub72] Oswald Huber. *Axiomatische Präferenzenlogik in der psychologischen Entscheidungsforschung.* PhD thesis, Philosophische Fakultät, Salzburg, 1972.

[KdC89] J. Kotas and N. C. A. da Costa. Problems of modal and discussive logic. In Priest et al. [PRN89], pages 227–244.

[Kel78] Jerry S. Kelly. *Arrow Impossibility Theorems.* Academic Press, 1978.

[KM75] A. Kron and V. Milanović. Preference and choice. *Theory and Decision*, 6:185–196, 1975.

[Mar63] Richard M. Martin. *Intension and Decision. A Philosophical Study.* Prentice–Hall Inc., 1963.

[MD90] P. Mathieu and J.-P. Delahaye. The logical compilation of knowledge bases. In van Eijck [vE90], pages 386–399.

[MKRT75] V. Milanović, A. Kron, S. Radlovački, and B. Tauzović. Istraživački projekt: Projektovanje organizacije proizvodnih sistema. Technical report, Mašinski Fakultet Novi Sad, 1975.

[Mor71] Edgar Morscher. A matrix method for deontic logic. *Theory and Decision*, 2:16–34, 1971.

[Mor89] Chris Mortensen. Paraconsistency and $\mathbf{C}_1$. In Priest et al. [PRN89], pages 289–305.

[Ovc91] Sergei Ovchinnikov. Social choice and Łukasiewicz logic. *Fuzzy Sets and Systems*, 43:275–289, 1991.

[Por84] Jean Porte. Łukasiewicz's Ł–modal system and classical refutability. *Logique and Analyse*, 27:87–92, 1984.

[PRN89] G. Priest, R. Routley, and J. Norman, editors. *Paraconsistent Logic. Essays on the Inconsistent.* Philosophia Verlag, Munich, 1989.

[Res67] Nicholas Rescher, editor. *The Logic of Decision and Action.* Pittsburgh, 1967.

[Res68] Nicholas Rescher. *Topics in Philosophical Logic.* Synthese Library. D. Reidel Publishing Company, Dordrecht, 1968.

[Res69] Nicholas Rescher. *Many–valued Logic.* McGraw–Hill, 1969.

[Ste90] Guy L. Steele, Jr. *Common LISP. The Language*. Digital Press, second edition, 1990.

[Tok75] Marek Tokarz. Functions definable in Sugihara algebras and their fragments I. *Studia Logica*, 34:295–301, 1975.

[Tok76] Marek Tokarz. Functions definable in Sugihara algebras and their fragments II. *Studia Logica*, 35:279–283, 1976.

[Tok80] Marek Tokarz. *Essays in the Matrix Semantics of Relevant Logics*. Warszawa, 1980.

[Urq86] Alasdair Urquhart. Many–valued logic. In Gabbay and Günthner [GG86], pages 71–116.

[vE90] Jan van Eijck, editor. *Logics in AI. European Workshop JELIA '90*, LNAI 478. Springer Verlag, 1990.

[vW63] Georg Henrik von Wright. *The Logic of Preference. An Essay*. Edinburgh University Press, Edinburgh, 1963.

[vW72] Georg Henrik von Wright. The logic of preference reconsidered. *Theory and Decision*, 3:140–169, 1972.