p ∧ ¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LP and Relatives

- Graham Priest is certainly the philosopher most widely known as a champion of paraconsistent logic and dialetheism. He has written several books and quite many papers on almost all aspects of paraconsistency.
- In one of his earliest papers on the subject he introduces the "Logic of Paradox" (**LP**). This chapter is concerned with this system and some extensions and refinements of it.
- Priest also has introduced other systems (Chap. 6 will deal with one version of one of them), but often talks in his work of "the" logic of paraconsistency, especially when arguing against critics, to claim that some theorems don‘t hold there. Often he refers to **LP** then. **LP**, however, fails on the *Modus Ponens* Condition! Priest sometimes then uses a "logic" that has no conditional connective at all! **LP** so needs – as Priest says as well – a supplementation by a Relevant conditional. The logic presented in this chapter should, therefore, be seen as to be extended by a Relevant conditional (both in its syntax as well as in its semantics, that then will no longer be extensional).
- [**LP** will be an ingredient in Chap. 7 and Chap. 20 as well.]

p ∧ ¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Quantification

- The problems with standard-logic that we encountered arose on the level of propositional logic.
- The solutions offered in Relevant or other paraconsistent logics, therefore, are formulated foremost at this level.
- There is nothing wrong with standard quantificational logic, at least its problems have – usually – nothing to do with paraconsistency. The debate about non-referring proper *names*, for example, is independent from the debate on paraconsistency.
- To keep things simple, and not to engage in other debates, the paraconsistent propositional systems can be extended by a standard quantificational logic (as well, of course, as by your favourite non-standard quantificational logic). So we extend, e.g., **LP** to **QLP**.
- One important exception to this general remark is the paraconsistent theory of impossible objects in the tradition of Meinong, taken up by Routley and others. Discussing a non-standard ontology will require a non-standard quantificational theory. [see Chap. 17; in the last chapter we consider also a combination of paraconsistency and Free Logic.]

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Contradictions in **LP**

- Contradictions in **LP** are sentences that are both true and false.
- That means that the evaluation function *v* in **LP** semantics assigns to a sentence either {0} (being false), or {1} (being true), or {0,1} (being true and false, i.e. a contradiction).
- The negation of a contradiction is true, since the contradiction is false (not only true). So in case of A being a contradiction A has the set of truth values {0,1} and A and ¬A are both true.
- We can thus proceede to define what an evaluation function *v* from the set of **LP**-evaluations *V* should be like.
- [Note: To avoid confusions from now on *simply true* sentences are sentences that are true and not false, whereas true sentences might be sentences that are true and false, i.e. contradictions. The same for *simply false*.]
- [Note: Later we shall see that we really need an evaluation *relation* instead of an evaluation function *v* to avoid hypercontradictions (see Chap. 16). To keep things simple we avoid these complications here.]

$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Negation in **LP**

- Negation in **LP** has the semantic rule:

  (LP¬) (i) $1 \in v(\neg A) \Leftrightarrow 0 \in v(A)$
  (ii) $0 \in v(\neg A) \Leftrightarrow 1 \in v(A)$

- Negation in LP not only fulfils the two conditions on an intuitive acceptable negation from Chapter 2, (i) is the standard clause for negation. (ii) supplements it for the case of contradictions: a contradiction A has $0 \in v(A)$, so by (i) $1 \in v(\neg A)$; but we also have to have that ¬A is a contradiction (any negation of a contradiction is a contradiction itself, since a contradiction is also true), i.e. $0 \in v(\neg A)$, which we don‘t get by (i), since we cannot reason from the negation of the right hand side of (i) in case of a contradiction; thus we need (ii).
- **LP** negation is an extension of standard negation. One can say:
  (i‘) ¬A is at least true, if A is at least false.
  (ii‘) ¬A is at least false, if A is at least true.

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Extensionality

- Not only negation, but all connectives are treated extensionally, thus meeting the Extensionality Condition.
- With negation and conjunction the other connectives are definable. The LP conjunction has the rule:

  (LP∧) (i) $1 \in v(A \wedge B) \Leftrightarrow 1 \in v(A)$ and $1 \in v(B)$

  (ii) $0 \in v(A \wedge B) \Leftrightarrow 0 \in v(A)$ or $0 \in v(B)$

- [Note: Evaluation functions are – as usual – the means to decompose a long formula into its parts to derive at a final assignment of a truth value. They are *compositional*, as is the semantics of a natural language. Otherwise it could not be the language of finite beings, who are able to understand any new sentence.
  Dropping the compositionality requirement and gerrymandering a set of evaluations V consisting of both compositional and specially defined evaluations (cf. Marcos 2005, pp.291-93) yields results like **PC** allowing for inconsistent models without triviality (of the consequence relation). This hints at the importance of keeping results concerning highly artificial systems apart from systems close to *our* natural logic and natural languages.]

$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Extensionality (II)

- For the connectives we get:

  - the negation of a contradiction is a contradiction
  - a conjunction is simply true if both conjuncts are true only
  - a conjunction of true sentence with a contradiction is a contradiction
  - a disjunction is simply false if both disjuncts are simply false
  - one disjunct being simply true makes the disjunction simply true
  - the conditional is simply false only if a simply true antecedent has a simply false consequent [this corresponds to standard understanding]
  - a conditional with a simply false antecedent is simply true
  - a conditional with a simply true consequent is simply true

- For a more concise overview we can give **LP** truth tables:

$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LP Truth Tables

A	**B**	**¬A**	**A∧B**	**A∨B**	**A⊃B**	**A≡B**
1	1	0	1	1	1	1
1	0	0	0	1	0	0
1	0,1	0	0,1	1	0,1	0,1
0	1	1	0	1	1	0
0	0	1	0	0	1	1
0	0,1	1	0	0,1	1	0,1
0,1	1	0,1	0,1	1	1	0,1
0,1	0	0,1	0	0,1	0,1	0,1
0,1	0,1	0,1	0,1	0,1	0,1	0,1

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Lukasiewicz‘s Truth Tables

On first sight the **LP** truth tables look like the truth table of a 3-valued logic with "0,1" being the third value. Compare the truth tables of Lukasiewicz‘s 3-valued logic (with 3rd value "indet(erminate)"):

A	**B**	**¬A**	**A∧B**	**A∨B**	**A⊃B**
1	1	0	1	1	1
1	0	0	0	1	0
1	indet	0	indet	1	indet
0	1	1	0	1	1
0	0	1	0	0	1
0	indet	1	0	indet	1
indet	1	indet	indet	1	1
indet	0	indet	0	indet	indet
indet	indet	indet	indet	indet	indet

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Designated Values

- The truth tables correspond to each other (take "indet" for "0,1").
- But Lukasiewicz‘s logic is a restriction of standard logic in which, for example, Excluded Middle (A ∨ ¬A) fails, which **LP** will preserve. How is that possible?
- A tautology is a sentence which under any evaluation has a *designated* truth value. In standard logic the designated truth value is truth (designated by "1"), so that a tautology is a sentence the columns of which in a truth table has 1s only. A sentence that is true always.
- "indet" is no designated value in Lukasiewicz‘s semantics, since a sentence that is indeterminate need not be true. Therefore A ∨ ¬A fails: A being indeterminate makes ¬A indeterminate, and a disjunction of two indeterminate sentences is indeterminate, thus not true, thus the column of A ∨ ¬A has a line with a non-designated truth value.
- In **LP** on the other hand we can point out that a tautology is a sentence that is always true. A contradictory sentence is *true* nevertheless, thus a sentence which has a line with "0,1" in its column still has a "1" in that very line. "0,1" is a designated truth value in **LP**. So A ∨ ¬A holds.

$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LP Validity

- Taking the set of designated truth values to be {{1},{0,1}} we can define validity and consequence:
- (LPV) $\models_{LP} A \Leftrightarrow (\forall v)(1 \in v(A))$

A sentence is LP-valid iff it is at least true in all evaluations.

- (LPC) $\Gamma \models_{LP} A \Leftrightarrow (\forall v)(1 \in v(A) \vee (\exists B \in \Gamma)(\neg(1 \in v(B))))$

Sentence A is a LP-consequence of the premise set Γ iff it is either true or some sentence in Γ is simply false.

- From this we get a simple meta-theorem:
  (LPMT1) LP is an extension of PC.
  Justification: By (LPV) at most more sentence may receive a designated truth value, all PC-tautologies are tautologies still.
- Tautologies can be proven in **LP** by doing the truth table of a formula. As in **PC** truth table provide a decision procedure for (propositional) **LP**. The existence of the decision procedure proves (some kind of) completeness of **LP**, since any tautology can be found thus.
  (LPMT2) LP is complete.

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Universal Satisfiability

- We should note a further peculiarity of taking both {1} and {0,1} as designated values:
- We can no longer say – as we do in standard logic – that A|=B just in case {A, ¬B} is *unsatisfiable*, respectively that A|≠ B if {A,¬B} is *satisfiable*.
- Indeed {A,¬A} is satisfiable, and we have $|=_{LP}(A \supset A)$.
- Furthermore we can even have an LP-evaluation that interprets any atomic sentence as {0,1} and thus evaluates every sentence as {0,1}.
- This gives – since we can have such an interpretation for any set you like – the meta-theorem:

(LPMT3) There are no unsatisfiable sets of sentences in **LP.**

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Truth Table Example

As an example of doing truth tables in **LP** we show that the usual dual of material implication and disjunction holds:

(LPT1) $\neg p \vee q \equiv p \supset q$

p	**q**	**¬ p ∨ q**	**p ⊃ q**
1	1	0 1	1
1	0	0 0	0
1	0,1	0 0,1	0,1
0	1	1 1	1
0	0	1 1	1
0	0,1	1 1	1
0,1	1	0,1 1	1
0,1	0	0,1 0,1	0,1
0,1	0,1	0,1 0,1	0,1

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Expressing Truth Values within **LP**

- For later purposes [in this chapter and chapter 20] and as an interesting sideline we can express some of the semantic properties of LP-sentences with operators on object-language sentences.
- Let ΔA be the sentence saying that A is simply true. Let ∇A be the sentence saying that A is simply false. TA says that A is true. FA says that A is false. °A says that A is not contradictory. •A says that A is contradictory. We can define these operators by truth tables:

A	TA	FA	ΔA	∇A	°A	•A
1	1	0	1	0	1	0
0	0	1	0	1	1	0
0,1	1	1	0	0	0	1

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Expressing Truth Values within **LP** (II)

- To these truth table definitions correspond obvious definitions of the operators in terms of each other, like:

  $\nabla A \quad := \quad \Delta\neg A$

  $\circ A \quad := \quad \Delta A \vee \nabla A$

  $\bullet A \quad := \quad \neg\circ A$

- We could *extend* the language of **LP** to include these monadic operators.
- In this chapter we only will use these operators in the decision procedure for **LP** and the semi-decision procedure for **LPQ**.
- Note that all these operators are bivalent! In the light of our intended interpretation of them this means that making a statement about the truth value(s) of a sentence results never in a sentence that evaluates to {0,1}, i.e. is contradictory.
- Expressing semantic properties with these operators within a given PL will the one major objective in the chapter on DaCosta-systems.

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LP Standard Theorems and Rules

- From (LPMT1) one might follow that **LP** is Irrelevant since it keeps all standard tautologies. **LP** keeps, for example:

  (1) $p \wedge (p \supset q) \supset q$ [*Modus Ponens* Theorem]

  (2) $(p \supset (p \supset q)) \supset (p \supset q)$ [Absorption Theorem]

  The validity of (2) seems to endanger the Strong Anti-Triviality Condition. But the situation may be even worse, since LP has

  (3) $p \wedge \neg p \supset q$

  With sentence (3) even the Weak Anti-Triviality Condition seems to be violated.
- One has to distinguish however, the validity of sentences, for which (LPMT1) holds, from the validity of a consequence relation. Many standard consequences do not hold in **LP**. "$|=_{LP}$" does not mirror the conditional or logical true conditionals.
- Valid are the following consequence relations:

  $A|=_{LP}A\vee B$ $A, B |=_{LP}A\wedge B$

  $A \supset B |=_{LP} \neg B \supset \neg A$ $\neg\neg B|=_{LP} B$

  $A\wedge B |=_{LP}A$ $(A \supset (C \supset B)) |=_{LP}(C \supset (A \supset B))$

  So Contraposition, Negation-Elimination, Permutation, Addition, Conjunction-Introduction and -Elimination are all LP-consequences.

p ∧ ¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LP Non-Consequences

- From Chap. 3 we already know that there is a counterexample to Disjunctive Syllogism if one has true contradictions and standard negation. So: $\neg A, A \vee B \mid\!=\!/\!=_{LP} B$
- Other non-consequences of **LP** are:

  $\neg A \wedge A \mid\!=\!/\!=_{LP} B$ [Explosion, *ex contradictione qoudlibet*]
  $A \supset B, B \supset C \mid\!=\!/\!=_{LP} A \supset C$ [Transitivity of "$\supset$"]
  $A, A \supset B \mid\!=\!/\!=_{LP} B$ [*Modus Ponens*]
  $A \supset B, \neg B \mid\!=\!/\!=_{LP} \neg A$ [*Modus Tollens*]
  $A \supset B \wedge \neg B \mid\!=\!/\!=_{LP} \neg A$ [Negation-Introduction]
- The non-validity of Explosion ensures **LP** satisfying the Weak Anti-Triviality Condition, making **LP** paraconsistent. But this and the satisfaction of Strong Anti-Triviality depends on the non-validity of *Modus Ponens*. So **LP** fails on the Modus Ponens Condition, and – Transitivity also being not valid – with respect to "$\supset$" on the Minimal Damage Condition. **LP** has *no* acceptable conditional connective at all!
- Taking **LP** as the logic of paraconsistency, therefore, requires that we supplement **LP** with a real conditional connective. [Keep this in mind for Chap. 5 where you might chose your favourite "$\rightarrow$" Relevant axioms.]

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Testing for LP-Consequences

- Using truth-tables we can establish all LP-theorems.
- Can we also establish all true LP-consequences?
- Sofar we have given no decision procedure to establish which consequence relationships hold in LP.
- In LP the Deduction Theorem holds:

  (LPMT4) $A_1...A_n|=_{LP}B \Rightarrow A_1...A_{n-1}|=_{LP}A_n \supset B$

  Justification: If the antecedent of (LPMT4) is true (i.e. the consequence relationship holds), $A_1...A_n$ and B are each (at least) true; so by the truth table of "$\supset$" $A_n \supset B$ is (at least) true, so again the consequence relationship in the consequent of the theorem holds.
- Formalizing an argument we can employ the Deduction Theorem to frame it as a conditional. We can test this conditional by the truth-table method. If the conditional is not logically true, the consequence relationship does not obtain: We have a *negative test* for consequences. We have no *positive test* for consequences.
  [Taking the validity of the built conditional as such a test would be affirming the consequent with respect to (LPMT4).]

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LP-Trees

- Using truth-tables can be tiresome. Using trees as a decision procedure leads quicker to results. Anthony Bloesch developed a tree procedure for **LP**, which can be extended to quantificational **LP**.
- We use the statements expressing semantic properties in this procedure: TA, FA, ΔA, ∇A. We can say of these which are incompatible:

TA	**∇A**
FA	**ΔA**
∇A	**ΔA**

ΔA and ∇A are not contradictories, but incompatible. The other two oppositions are contradictories.

p ∧¬p

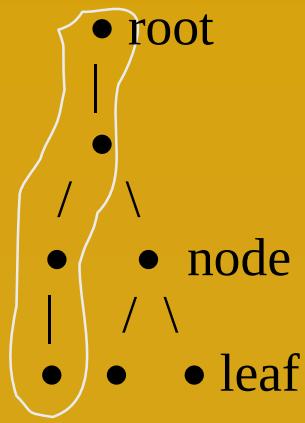
Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Trees

- Trees in general are objects that look like bottom up tress: the starting point is the *root*, points in general are called *nodes*, points which are connected are said to be on the same *branch*, nodes that are the last in a branch are called *leaves*. A branch extend as a series of nodes from the root to one leave. So a prototypical tree looks like that

encircled area
is a branch



- Trees provide a decisions procedure (in the propositional case) or a semi-decision procedure (giving the valid sentences in the quantificational case) by decorating the nodes with sentences and stepwise decreasing the syntactic complexity of a sentence. To do this we need (a) some general definitions of *closed* trees or branches relating to validity, and (b) specific rules for the logical vocabulary.

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Closed and Open Tree

- (LPB1) A branch of a tree is *satisfiable* iff all sentences on this branch can get a designated truth value in some interpretation *v*.
- This could also be expressed in terms of incompatible truth values:
  (LPB1‘) A branch of a tree is *satisfiable* iff no sentence on it has to be assigned incompatible truth values by the rules.
- (LPB2) A branch that is not satisfiable is *closed*.
- (LPB3) A tree is satisfiable if at least one branch is satisfiable.
- (LPB4) A tree is closed iff all its branches are closed.
- Trees are an indirect procedure: Given some supposed consequence relation $\Gamma|=A$ the root is decorated with TB for all $B\in\Gamma$ and ∇A; then the rules are employed. We get a tree *for* $\Gamma|=A$.
  In case of a supposed theorem Γ may be empty.
- (LPB5) A is a logical truth (resp. $\Gamma|=A$ holds) iff its tree is closed.
- The decisive meta-theorem states the adequacy of the tree procedure:
  (LPMT5) $\Gamma|=_{LP}A \Leftrightarrow$ the tree $TB_1...TB_n$ $(B\in\Gamma)$, ∇A is closed.

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Rules for LP-Trees

- We need rules to decrease syntactic complexity for every connective and for all four basic truth operators. The formulas written beneath are a next node in a given tree with the formula above occurring in it. A "|" indicates that at this point a branching occurs.

T(¬A) **F(A)**	**F(¬A)** **T(A)**	**∇(¬A)** **Δ(A)**	**Δ(¬A)** **∇(A)**
T(A ∧ B) **T(A)** **T(B)**	**F(A ∧ B)** **F(A) | F(B)**	**∇(A ∧ B)** **∇A | ∇B**	**Δ(A ∧ B)** **ΔA** **ΔB**
T(A ∨ B) **T(A) | T(B)**	**F(A ∨ B)** **F(A)** **F(B)**	**∇(A ∨ B)** **∇(A)** **∇(B)**	**Δ(A ∨ B)** **Δ(A) | Δ(B)**
T(A ⊃ B) **F(A) | T(B)**	**F(A ⊃ B)** **T(A)** **F(B)**	**∇(A ⊃ B)** **Δ(A)** **∇(B)**	**Δ(A ⊃ B)** **∇(A) | Δ(B)**
T(A ≡ B) **T(A) | F(A)** **T(B) | F(B)**	**F(A ≡ B)** **T(A) | F(A)** **F(B) | T(B)**	**∇(A ≡ B)** **∇(A) | ∇(B)** **Δ(B) | Δ(A)**	**Δ(A ≡ B)** **Δ(A) | ∇(B)** **Δ(B) | ∇(A)**

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LP-Tree Example

- To give an example of a LP-tree we show that *Modus Ponens* does not hold as a consequence relationship:

  T(p)
  T(p ⊃ q)
  ∇(q)
  / \
  F(p) T(q)

  There is only one rule applicable here, for "⊃". The right branch is *closed*, since T(q) and ∇(q) are incompatible, so there is no interpretation that can make both these sentences (at least) true. The left branch, however, is *satisfiable*, since T(p) and F(p) are not incompatible in case "p" was a contradictory sentence.
- Since one of its branches is satisfiable, the tree is satisfiable, so it is not closed, so the tree for *Modus Ponens* is not closed, so by (LPMT5) *Modus Ponens* is not LP-valid.

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LP-Tree Example (II)

- Absorption is valid in **LP**. So we get a closed tree for it:

$\nabla((p \supset (p \supset q)) \supset (p \supset q))$
| (rule $\nabla(A \supset B)$)
$\Delta(p \supset (p \supset q))$
$\nabla(p \supset q)$
| (rule ∇ $(A \supset B)$)
$\Delta(p)$
$\nabla(q)$
/ \ (rule $\Delta(A \supset B)$)
$\nabla(p)$ $\Delta(p \supset q)$

- One can stop here, although we could apply a rule once more on the right branch, since both branches are closed: The left branch is closed, since $\Delta(p)$ and $\nabla(p)$ are incompatible; the right branch is closed, since $\Delta(p \supset q)$ and $\nabla(p \supset q)$ are incompatible. Since all branches of the tree for Absorption are closed, the tree for Absorption is closed, therefore Absorption is LP-valid.
[Remember: A closed tree means that there is no interpretation *v* that gives a coherent assignment of truth values in the supposed *counterexample* to the sentence‘s validity.]

p ∧ ¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Quantificational **LP**

- **LP** like most PLs can be extended by standard quantificational logic as said before. As said we consider only the basic construction with a fixed domain of individuals. All extensions (like a theory of – possibly non-referring – names and descriptions, and an existence predicate) pose questions not directly related to paraconsistency, at least most of the time [see (Priest 1999) and Chap. 20].
- The interesting point is how in the semantics of predicates true contradictions can arise.
- Quantificational **LP**, **LPQ**, is a language that results from **LP** (and its syntax) by introducing n-ary predicates like "$F_1(,)^n$", individual constants like "a", individual variables like "x", and quantifiers, (∀x), (∃x), in the usual way. Only formulas where all variables are bound are called "sentence", otherwise "open formula".
- We use the same expressions in the meta-language as translations when talking about individual LPQ-sentences ("homophonetic translation"). As schematic expressions we use "$P_1(,)^n$" etc. for n-ary predicates/relations, "á" etc. for individual constants and variables for variables.
  [Note: This convention is from now on applied in all chapters!]

$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Quantificational Semantics

- An **LPQ** model *M‘* is a tuple <D, *v*, *G*>, where D is the domain of individuals we quantify over, *v* some interpretation function and *G* the set of all variable assignments. Two Models *M‘* and *M‘‘* differ either in their domain or in the interpretation used.
- (LPQS1) $v(\text{á}) = d \in D$
- (LPQS2) For $g \in G$ $g(x) = d \in D$
- So individual constants are interpreted over the domain, individual variables get assigned some member of the domain.
- For predicates P we assign them not only an extension (as usually done by taking some set of individuals of the domain), but also an *anti-extension*. Both are subsets of the domain D. We write: P^+ and P^-.
  (LPQS3) $v(P^n) = \langle P^+, P^- \rangle$, $P^+ \subseteq D^n$, $P^- \subseteq D^n$.
  D^n is the n^{th} Cartesian product of D, the set of all n-tuples in D.
- The idea is that there might be criteria the application of which may result in grouping an object within the extension of the predicate in question. There may also be other criteria the application of which may result in grouping an object within the anti-extension of the predicate in question.

$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Quantificational Semantics (II)

- Inasmuch as these criteria need not be the exact complements of each other it might be that some object is grouped in the extension *as well as* in the anti-extension of the predicate. This objects thus has the property/feature corresponding to the predicate, and it lacks the property (or has a complementary property).
- Let a^n be a n-tuple of individual constants. Thus we get for elementary sentences:
  (LPQS4) $d^n = v(a^n)$, $v(<P^n, a^n>)=\{1\}$ iff $d^n \in P^{n+}$ and $d^n \notin P^{n-}$,
  $v(<P^n, a^n>)=\{0\}$ iff $d^n \in P^{n-}$ and $d^n \notin P^{n+}$,
  $v(<P^n, a^n>)=\{0,1\}$ iff $d^n \in P^{n+}$ and $d^n \in P^{n-}$.
- If $P(x_1...x_n)$ is an open formula, the interpretation of P does its work with respect to $< P(x_1...x_n),<g(x_1)...g(x_n)>>$, i.e. with a variable assignment. We write $v(A, g)$ to say that interpretation v evaluates formula A relative to variable assignment g.
- A universal sentence is simply false if there is an object for which the open formula in question is simply false. A universal sentence is contradictory if it is not simply false and at least one object is in the extension as well as the anti-extension of the open formula.

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Quantificational Semantics (III)

- As a semantic rule:
  (LPQS5) Let A be a sentence of the form $(\forall x)P(x)$, then:
  $v(A, g)=\{1\}$ iff for all $g' \in G$: g' is different from g at most with respect to x, and $g'(x) \in P^+$ and $g'(x) \notin P^-$.
  $v(A, g)=\{0\}$ iff for some $g' \in G$: g' is different from g at most with respect to x, and $g'(x) \in P^-$ and $g'(x) \notin P^+$.
  $v(A, g)=\{0,1\}$ iff for all $g' \in G$: g' is different from g at most with respect to x, and $g'(x) \in P^+$, and for some $g'' \in G$: g'' is different from g at most with respect to x, and $g''(x) \in P^-$.
- For existential quantification the dual holds in **LPQ**:
  (LPQT1) $(\exists x)P(x) \equiv \neg (\forall x) \neg P(x)$
  so one gets a dual semantic rule for $(\exists x)P(x)$.
- We define LPQ-validity and consequence, *M* being the set of models:

(LPQV) $\models_{LPQ} A \Leftrightarrow (\forall M' \in M)(1 \in v'(A))$

(LPQC) $\Gamma \models_{LPQ} A \Leftrightarrow (\forall M' \in M)(\exists B \in \Gamma)(v'(B)=\{0\} \vee (1 \in v'(A))$

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LPQ-Trees

- We won‘t characterise **LPQ** axiomatically here, but derive some theorems by LPQ-trees as a semi-decision procedure.
- To do this we need rules for the two quantifiers with cases for the four semantic operators. Everything else we take over from LP-trees.
- As in standard instantiations of the quantifiers we have to take care that some formulas are instantiated to a new constant, some rules can be employed more than once.

T((∀x)P(x)) **T(P(á))** **+**	**F((∀x)P(x))** **F(P(á))** *****	**∇((∀x)P(x))** **∇(P(á))** *****	**Δ((∀x)P(x))** **Δ(P(á))** **+**
T((∃x)P(x)) **T(P(á))** *****	**F((∃x)P(x))** **F(P(á))** **+**	**∇((∃x)P(x))** **∇(P(á))** **+**	**Δ((∃x)P(x))** **Δ((P(á))** *****
***** **á has to be a constant that did not occur in the tree before.** **+** **This rule might be employed more than once in the tree.**			

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

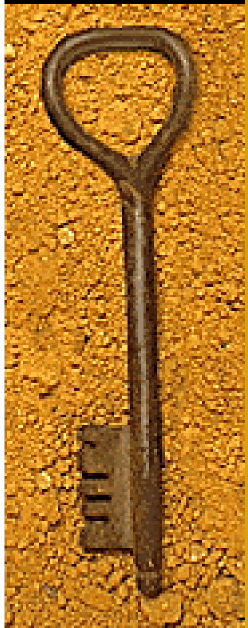
LPQ-Trees Examples

- (LPQT2) $(\forall x)F(x),(\forall x)G(x)|=_{LPQ}(\forall x)(F(x) \wedge G(x))$

  Proof:

  $T(\forall x)F(x)$
  $T(\forall x)G(x)$
  $\nabla(\forall x)(F(x) \wedge G(x))$
  $\quad|$ (rule $\nabla\forall$ with "a" new)
  $\nabla(F(a) \wedge G(a))$
  $\quad/ \quad \backslash$ (rule $\nabla\wedge$)
  $\nabla(F(a)) \quad \nabla(G(a))$
  $\quad| \quad\quad |$ (rule $T\forall$)
  $TF(a) \quad TG(a)$

- Both branches of this tree are closed, since TF(a) is incompatible with ∇(F(a)) and TG(a) with ∇(G(a). So the consequence relationship holds. Rules with an "*" are better employed at the beginning of a tree, since so one can later instantiate "+" rules on these constants. Beginning this tree with a usage of (T∀) would have been a waste, since the constants used then would have to be different from constants used with (∇∀).



$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

LPQ-Trees Examples (II)

- (LPQT3) $\models_{LPQ} (\forall x)F(x) \supset (\exists x)F(x)$

Proof

$$\nabla((\forall x)F(x) \supset (\exists x)F(x))$$
$$| \quad (\nabla\supset)$$
$$\Delta(\forall x)F(x)$$
$$\nabla(\exists x)F(x)$$
$$| \quad (\nabla\exists)$$
$$\nabla F(a)$$
$$| \quad (\Delta\forall)$$
$$\Delta F(a)$$

The incompatibility of the last two sentences closes this tree.

- Also valid are the following:
- (LPQT4) $(\forall x)(F(x) \supset G(x)) \models_{LPQ} (\forall x)(\neg G(x) \supset \neg F(x))$
  a quantified version of Contraposition.
- (LPQT5) $(\forall x)(F(x) \supset G(x)) \models_{LPQ} (\exists x)F(x) \supset (\exists x)G(x)$
- (LPQT6) $(\forall x)(P(x) \supset A) \models_{LPQ} (\exists x)P(x) \supset A$
  where x does not occur in A

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

LPQ-Trees Examples (III)

(LPQT7) $(\forall x)(F(x) \supset G(x)) \models_{LPQ} (\forall x)F(x) \supset (\forall x)G(x)$

Proof

$T(\forall x)(F(x) \supset G(x))$
$\nabla((\forall x)F(x) \supset (\forall x)G(x))$
| $(\nabla\supset)$
$\Delta(\forall x)F(x)$
$\nabla(\forall x)G(x)$
| $(\nabla\forall)$
$\nabla G(a)$
| $(T\forall)$
$T(F(a) \supset G(a))$
/ \ $(T\supset)$
$F(F(a))$ $T(G(a))$
| $(\Delta\forall)$
$\Delta F(a)$

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LPQ Adequacy

- We have seen that trees are mirroring **LP**-validity. Trees are also sound and complete for **LPQ**.
- Soundness is given by considering an invalid A. Since A is not valid there is an interpretation in some model that makes A simply false. The assumption that A is simply false at the root of the tree for A develops this interpretation in detail. The existence of this interpretation is mirrored by an open branch, so the tree is not closed. So A is not derivable by an **LPQ**-tree. Contrapositive: If A is derivable, A is valid.
- Completeness is given by considering a satisfiable tree. A sentence with its tree satisfiable is not derivable. A satisfiable tree determines by one of its branches an interpretation in some model. This interpretation includes the root of the branch, i.e. makes the sentence A simply false, so A is not valid. Contrapositive: If A is valid, A is derivable.
- (LPQMT1) $\Gamma \models_{LPQ} A \Leftrightarrow$ the tree $TB_1 \ldots TB_n$ $(B \in \Gamma)$, ∇A is closed.

[For the details in the propositional case see (Bloesch 1993), for the necessary refinement in an adequacy proof for trees in the quantificational case in general see (Ben-Ari 1993, pp.112-27).]

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LPQ with Identity

- Once we have **LPQ** we might want to extend **LPQ** with an identity relation (designated by "="). Let‘s call this language **LPQ**$^{=}$.
- How could we define identity? Does for any object, *inconsistent* objects included, (∀x)(x=x) hold?
- Often identity is defined by *Leibniz‘ Law*: *a* and *b* are identical if all properties that *a* has *b* has as well. In second order logic we can say:

  (LL) $(\forall x,y)(x = y \equiv (\forall F)(F(x) \equiv F(y)))$

- Since **LPQ** is first order, we cannot express (LL) in it.
  [A schematic representation of (LL) would hold only for some predicate P, or for too many constants, or requires infinite premises/knowledge for right to left detachment.]
- Even if we cannot define identity within **LPQ**, we may ask ourselves which rules/theorems should hold with respect to "=". The two obvious candidates for natural deduction rules are:

  (=I) $\ldots \Rightarrow$ á = á

  (=E) á = é, P(á) $\Rightarrow$ P(é)

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

LPQ with Identity (II)

- (=E) is an expression of substitution of identicals.
- Interestingly enough substitution of identicals does *not* hold!
  Given (LL) substitution of identicals comes down to detachment with respect to "≡". But like the conditional "≡" is not detachable in **LP**.
- The same failure of detachment blocks the validity of
  (*) $a = b, b = c \Rightarrow a = c$
  Given (LL) and "≡" in **LP** (*) comes down to transitivity of "≡", which does not hold in **LP**.
  So supervenient on that failure the transitivity of identity does not hold in $\mathbf{LPQ}^{=}$.
- This seems to be a major shortcoming for a definition along (LL) in second order **LPQ**.
  It may turn out, however, to be a merit (see Chap. 16)!

p ∧¬p

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Summary

- **LP** provides a paraconsistent logic with an easily understood semantics keeping the Extensionality Condition.
- **LP** can be easily employed, also in its quantificational extension **LPQ**, using trees.
- **LP**, however, fails on the Modus Ponens Condition and so contains no acceptable conditional. **LP**, therefore, can only be regarded as a possible ingredient for the real paraconsistent logic (e.g. one extending **LP** with a Relevant conditional). **LP** certainly *is not* "the logic of paradox".
- Keep in mind as assets for other PLs:
  - the standard treatment of negation
  - the usage of {0,1} as a designated truth value
  - the expressibility of bivalent "semantic" connectives with respect to the paraconsistent distribution of truth values
  - the idea of the pair of extension and anti-extension in the interpretation of a predicate/relation.

$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Questions

- (Q1) What is the philosophical justification to take {0,1} as a designated truth value? Isn‘t it in conflict with our ordinary concept of logical truth?
- (Q2) Consider **LPQ**. The criteria for the anti-extension of P apply to all objects. Does this make $(\exists x)P(x)$ false?
- (Q3) What is the difference between a "true contradiction" and a (merely) "false contradiction" in **LP**?
- (Q4) Given a sentence A such that $A \equiv B \wedge \neg B$, what are the truth values of TA and FA?
  What is the intuitive reading of these sentences?
- (Q5) Given a paraconsistent quantificational logic and its semantics should we allow for objects such that $a \neq a$?
- (Q6) Why would the extension of **LP** which has the truth operators no longer fulfil the universal satisfiability meta-theorem?

$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Exercises

- (Ex1) Give counter-examples for the consequence relations that do not hold in **LP** [*Modus Tollens* etc.].
- (Ex2) Show by truth tables that $\models_{LP}(p \supset q) \supset (\neg q \supset \neg p)$
- (Ex3) Show by a satisfiable **LP**-tree that transitivity of "$\supset$" does not hold as a consequence relationship. Do the same for standard Negation-Introduction.
- (Ex4) Show the validity of (LPQT4)-(LPQT6) by trees.
- (Ex5) Show by trees that the following are not **LPQ**-valid:
  $(\forall x)(F(x) \supset G(x)), (\forall x)F(x) \models (\forall x)G(x)$
  $(\forall x)(F(x) \supset G(x)), (\forall x)\neg G(x) \models (\forall x)\neg F(x)$
  $(\forall x)(F(x) \supset G(x)), (\forall x)(G(x) \supset H(x)) \models (\forall x)(F(x) \supset H(x))$
- (Ex6) Suppose we extended **LPQ** with the consistency/in-consistency operators "$\circ$"/"$\bullet$". We would need a further quantificational axiom to replace occurences of " $\bullet(\forall x)P(x)$ " (starting with "$\bullet$", defining "$\circ$"). Read off the axiom from the semantics of "$\forall$".

$p \wedge \neg p$

Manuel Bremer
Centre for Logic, Language and Information

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Further Reading

- The classic paper on **LP** is: Priest, Graham. "The Logic of Paradox", *Journal of Philosophical Logic*, 8 (1979), pp. 219-41.
- The propositional tableau method you find in: Bloesch, Anthony. "A Tableau Style Proof System for Two Paraconsistent Logics", *Notre Dame Journal of Symbolic Logic*, (1993), pp. 295-301.
- On Priest‘s dialectic use of **LP** see (Priest 1982, 1991, 1993, 1994, 1995, 1997, 1999 ...).