LIP6 CNRS

Logique et représentation des connaissances
Spécialités DAC et Androide

Enseignants:

Nicolas Maudet (resp. Androide)
Jean-Gabriel Ganascia (resp. DAC)
Gauvain Bourgne
Colette Faucher* *(projet)*
Marie-Jeanne Lesot

Informations Générales

- **Cours:** amphithéâtre 55B puis 34A, mercredi, 13H45-15H45
- **Premier cours:** 14 septembre 2022
- **Dernier cours**: 7 décembre 2022
- **Examens répartis:**
  - **Examen réparti n° 1:** semaine du 7 novembre 2022
  - **Examen réparti n° 2:** 11 janvier 2023
- **Projet:**
  - **Sujet donné:** semaine du 17 octobre
  - **Rendu:** 1er décembre au plus tard
- **Barème:**
  - **Note =** Examen réparti 1*0,4 + Examen réparti 2*0,4 + projet*0,2
  - **Note rattrapage =** Examen rattrapage*0,8 + projet*0,2

  Remarque: La note de projet est conservée pour le rattrapage

Informations Générales *(suite)*

- **TD/TME**
  - **Lundi après-midi (G1 et G2),**
  - **jeudi matin(G4 et G5)**

Moodle: https://moodle-sciences-22.sorbonne-universite.fr/course/view.php?id=2799

Agenda: https://cal.ufr-info-p6.jussieu.fr/master/

Cours LRC: logique(s) et représentation(s) des connaissances

1. Introduction à la logique des propositions et des prédicats du premier ordre.
   Règles de résolution.
   Méthode des tableaux.

Représentation des connaissances

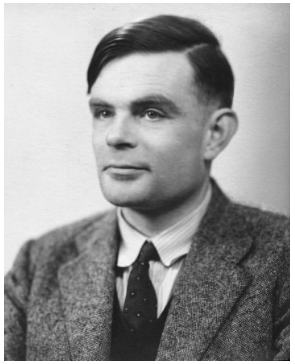
2. Unification et Prolog
   Représentations sémantiques, graphes conceptuels.
3. Logiques de description *(syntaxe et sémantique)*.
4. Logiques de description: raisonnement automatique *(méthode des tableaux et subsomption structurelle)*

Logiques modales

5. Introduction aux logiques modales
6. Logiques épistémique
7. Connaissances communes, connaissances partagées

Représentations du temps

8. Intervalles d'Allen
9. Réseaux de Petri
10. Automates temporisés



History of Logic (-500 – …)

1. Describing the correct way of reasoning – **Philosophy** (*Aristotle, …*)
2. Mathematizing Logic – **Philosophy** (*Leibniz, Boole, …*)

3. Mechanizing Mathematics – **Mathematics** (*Hilbert, Herbrand, Tarski…*)
4. Computerizing the Proof – **Mathematics, Logic, AI & Computer Science** (…)

L
I
P
6

C
N
R
S

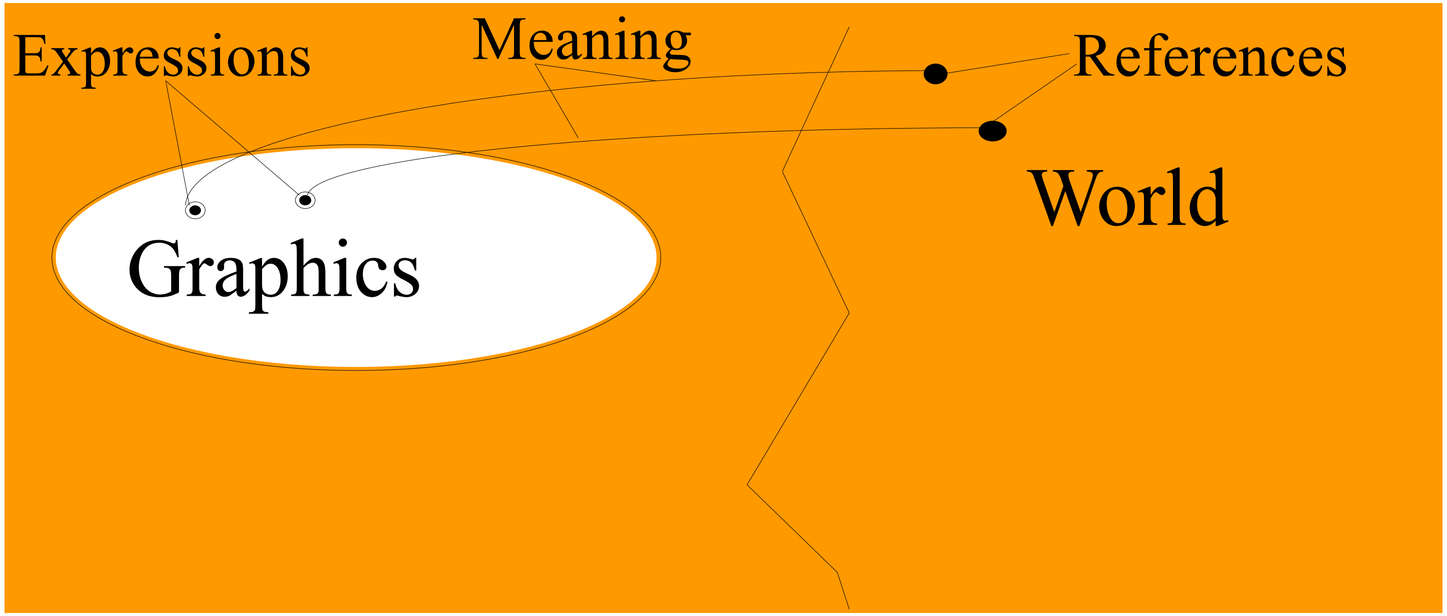
Représentation en logique

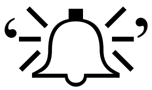
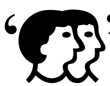
- Signe, sens, sémantique et vérité
- Proposition
- Le langage de la logique propositionnelle
- Limites de la logique propositionnelle
- La logique des prédicats
- Sémantique de la logique des prédicats

L I P 6 C N R S

Sign

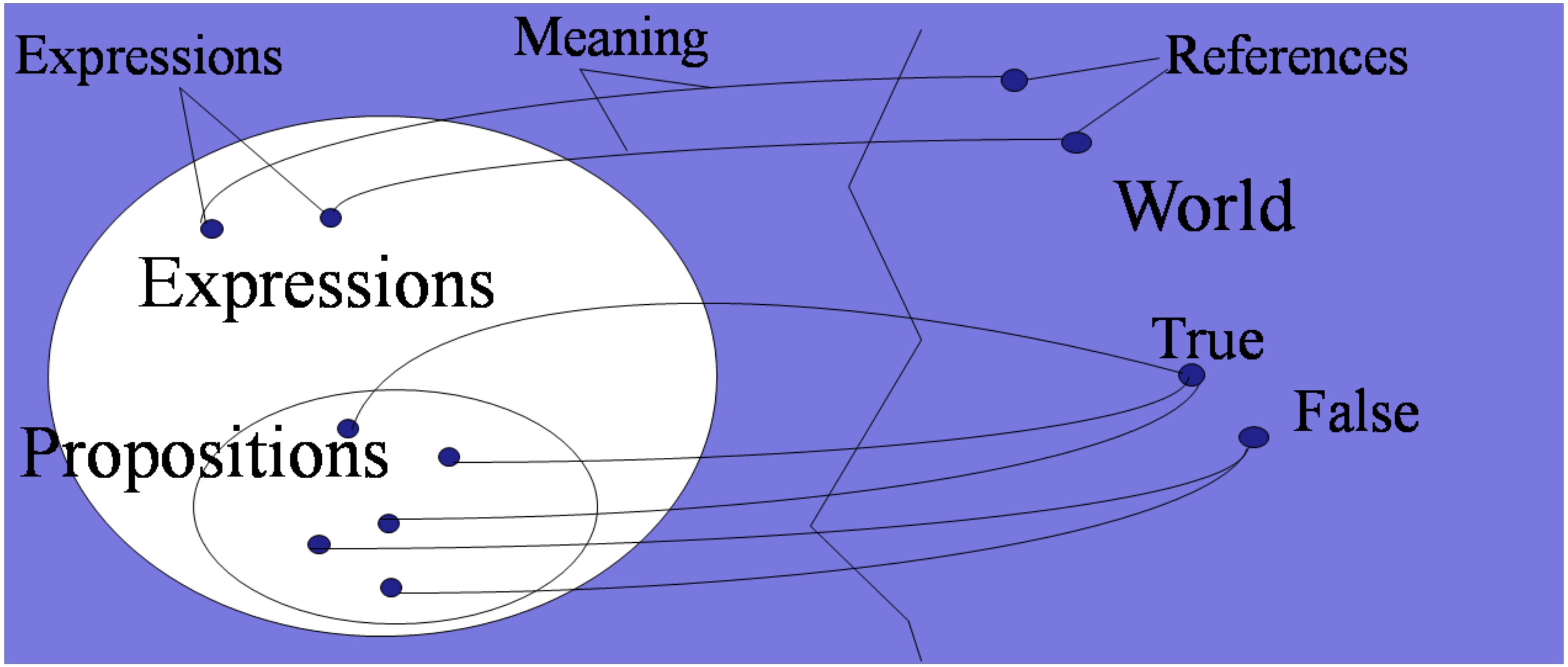
- Expression, Meaning, References



- Sign: graphic with meaning (i.e. intended to express)
  - **Examples**: “horse”, “reality”, “John”, “!”, “)”, “”, “”

Proposition

- Meaning = truth value $\in$ {true, false}
  - "Earth is blue"
  - "The green square is behind the red triangle"
  - "A book was published"



The Propositional Language

- **Atom: elementary propositions**
  - "It was raining"
  - "The cube is green"
- **Connectors**
  - Binary: « $\wedge$ », « $\vee$ », « $\rightarrow$ »
  - Unary: « $\neg$ »
- **Composed Propositions**
  - If A and B are propositions, « $\neg$ A», « A $\wedge$ B », « A $\vee$ B », « A $\rightarrow$ B » are propositions
- **Examples:**
  - $F_1=((p \rightarrow (q \rightarrow r)) \rightarrow ((p\rightarrow q)\rightarrow(p\rightarrow r)))$
  - $F_2=((p \wedge q \wedge \neg r) \vee ((\neg p\vee q) \rightarrow (\neg p\vee r)))$

Propositional Logic: semantic

- Truth Tables
- Satisfiability, validity, unsatisfiability
- Logical Consequence
- Limitation of Truth Tables

L I P 6 C N R S

Truth Table

- Every atom a has a *truth value*:
  true (v) or false (f)
- How to compute the truth value of a composed proposition?

« A ∧ B »

A\B	v	f
v	v	f
f	f	f

« A ∨ B »

A\B	v	f
v	v	v
f	v	f

« A → B »

A\B	v	f
v	v	f
f	v	v

« ¬ A »

A	
v	f
f	v

Example

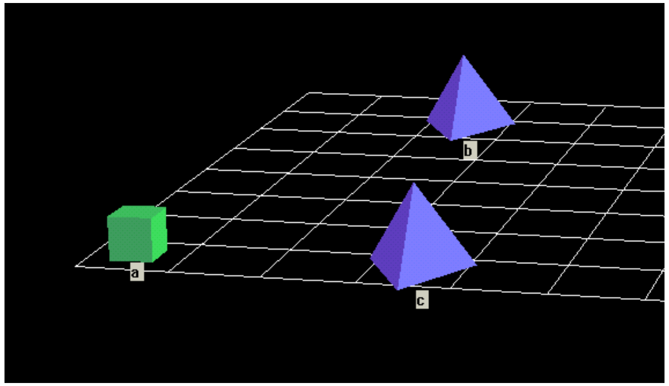
A	B	C	(B→C)	**(A→(B →C))**
v	v	v	v	**v**
v	v	f	f	**f**
v	f	v	v	**v**
v	f	f	v	**v**
f	v	v	v	**v**
f	v	f	f	**v**
f	f	v	v	**v**
f	f	f	v	**v**

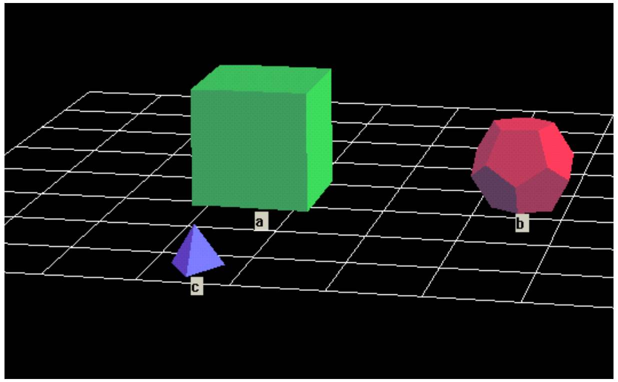
- Each line of the truth table is called an **interpretation**

Satisfiability – Unsatisfiability – Validity

- **A formula is said to be**
  - **“satisfiable”** if and only if it is true on at least one line of the truth table
  - **“valid”** if and only if it is true on all lines of the truth table
  - **“unsatisfiable”** if and only if it is false on all lines of the truth table, i.e. if it is never true
- **Remarks** :
  - F is *unsatisfiable* iff ¬F is *valid*
  - F is *unsatisfiable* iff F is not *satisfiable*

Propositional Logic





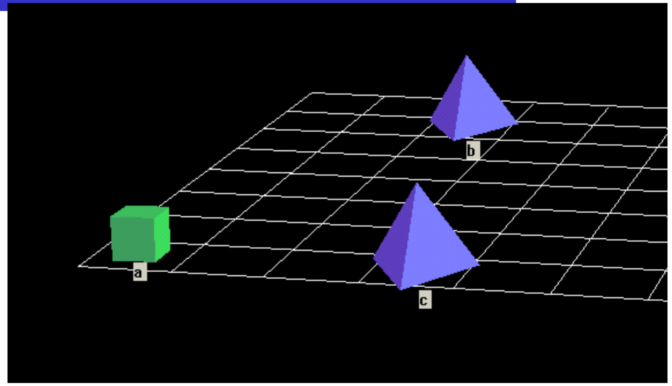
How to translate those sentences in propositional logic:

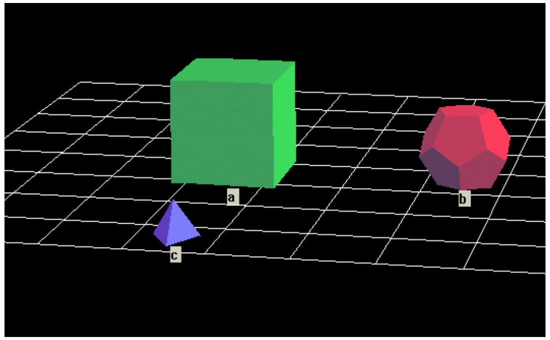
1. There is an object right to a green cube
2. Every successor of an even number is odd
3. 72 is an even number
4. The successor of 72 is odd

Introduction of propositional functions (*predicates*):

even(X): « X is even » is a proposition for all values of X.

Representation in Predicate Logic





Notion of propositional function:

1. X is even: `even(X)`
2. 72 is an even number `even(72)`
3. The successor of 72 is odd
   `odd(successor(72))`
4. Every successor of an even number is odd
   `∀x (even(x) → odd(successor(x)))`
5. There is an object to the right of a green cube
   `∃x ∃y (right_of(x, y)&cube(y)&green(y))`

Predicate Logic Language

- **Terms** ($\mathcal{T}$):
  - Being given $\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_n \cup \ldots$ a set of *function symbols* [$\mathcal{F}_i$ *corresponds to functions of arity* i]
  - Being given $\mathcal{V}$ a set of *variable symbols*

1. Each element of $\mathcal{V}$ is a **term**
2. For any n-tuple of terms $(t_1, t_2, \ldots t_n)$, for any function f_n of $\mathcal{F}_n$, $f_n(t_1, t_2, \ldots t_n)$ is a **term**

Predicate Logic Language

- **Atom** ($\mathcal{A}$):
  - Being given $Q = Q_0 \cup Q_1 \cup Q_2 \cup \ldots \cup Q_n \cup \ldots$ a set of *predicate symbols* [Q_i *corresponds to arity* i *predicates*]

For any n-tuplet of terms $(t_1, t_2, \ldots t_n)$,

For any predicate q_n of Q_n,

$q_n(t_1, t_2, \ldots t_n)$ is an **atom**

Remark: a predicate is a propositional function, i.e. It is a function of which values belong to {true, false}

Examples

- **Terms**:
  - successor(73), plus(72, 1), multiply(72, 2), plus(X, 1).
  - father(Jean), father(X)…
- **Atoms**:
  - even(72), odd(73), odd(successor(X)).
  - smoke(goat, cigar)
  - father(Jean, Peter)…

Formulae

- **Quantifier**:
  - If $x \in \mathcal{V}$ (x is a variable) and F a formula, then **∀xF** et **∃xF** are also formulae
    - **∀xF** means “for all x, F”
    - **∃xF** means “there exists x, F”
- **Formulae**:
  - Atoms are formulae
  - If F and G are formulae, «**¬ F**», «**F ∧ G**», «**F ∨ G**», «**F → G**» are formulae
  - If $x \in \mathcal{V}$, **∀xF** and **∃xF** are also formulae

Free – bounded variable

- An **occurrence** of a variable is said to be **free** if it does not appear under the scope of a quantifier (∀ or ∃), else it is called **bounded**

Example: in the following formula

$\forall x \exists y\ (\text{mother}(x, y) \wedge \text{married}(x, z))$

x and y are bounded

z is free.

Remark: a variable may have both free and bounded occurrences in the same formula

$\forall x\ (\ \text{married}(x, \underline{y}) \wedge \exists \underline{y}\ \text{mother}(x, \underline{y})\ )$

Interpretation

- Propositional Logic:
  - Truth values attributed to the different atomic propositions (*line of the truth table*)
- First Order Logic (i.e. predicate logic):
  - A non empty domain $\mathcal{D}$
  - An attribution of "value" to each symbol:
    - To each n-ary function f_n of $\mathcal{F}_n$, a function from $\mathcal{D}^n$ to $\mathcal{D}$, which is denoted $i[f_n]$
    - To each n-ary predicate symbol p_n of $\mathcal{P}_n$, a function from $\mathcal{D}^n$ to $\{v, f\}$, which is denoted $i[p_n]$

1^{st} order logic semantics

- Being given a domain $\mathcal{D}$
- We call "interpretation" a function i which attributes:
  - A n-ary function f_n of $\mathcal{F}_n$, a function from $\mathcal{D}^n$ to $\mathcal{D}$, which is denoted $i[f_n]$
  - To each n-ary predicate symbol p_n of $\mathcal{P}_n$, a function from $\mathcal{D}^n$ to $\{v, f\}$, which is denoted $i[p_n]$

- A formula is "**valid**" if it is true in <u>all</u> the interpretations of <u>all</u> domains
- A formula is "**satisfiable**" if it is true for <u>at least one</u> interpretation of <u>one</u> domain.
- A formula is "**unsatisfiable**" if it is never true in any domain, i.e. if it is false in <u>all</u> interpretations of <u>all</u> domains.

L
I
P
6

C
N
R
S

Examples

- Validity of $\neg\exists x P(x) \rightarrow \forall x(\neg P(x))$
- Unsatisfiability of $\forall x P(x) \wedge \exists y(\neg P(y))$
- Satisfiability of $\forall x(\neg \text{mâle}(x) \rightarrow \text{femelle}(x))$
- Invalidity of $\forall x(\neg \text{mâle}(x) \rightarrow \text{femelle}(x))$
- Validity of $\forall x((\neg \text{mâle}(x) \rightarrow \text{femelle}(x)) \vee$
  $(\neg \text{mâle}(x) \wedge \neg \text{femelle}(x))$

Model

Being given a domain $\mathcal{D}$ and an interpretation i

- We call "model" (or a structure) a couple $\mathcal{M} = < \mathcal{D}, i >$
- We call "valuation" a function v: $\mathcal{V} \rightarrow \mathcal{D}$
- $I_{mv}(F)$, the truth value of the formula F is defined as follows:
  - $I_{mv}(P(t_1, t_2, \ldots t_n)) = \mathrm{t}$ if and only if $i(P(t_1, t_2, \ldots t_n)) = \mathrm{t}$

  i.e. $(I_{mv}(t_1), I_{mv}(t_2), \ldots, I_{mv}(t_n)) \in \mathrm{i}(I_{mv}(P))$
  - $I_{mv}(\neg F) = \neg\, I_{mv}(F)$
  - $I_{mv}(F \wedge G) = I_{mv}(F) \,.\, I_{mv}(G)$
  - $I_{mv}(F \vee G) = I_{mv}(F) + I_{mv}(G)$
  - $I_{mv}(F \rightarrow G) = I_{mv}(\neg F \vee G)$

1^{st} order logic semantics

L I P 6 C N R S

- Being given a domain $\mathcal{D}$ and an interpretation I
  a model $\mathcal{M}$ is a tuple $<\mathcal{D}, i>$ and
  a valuation v if a function $\mathcal{V} \rightarrow \mathcal{D}$
- A formula is "**valid**" if it is true in all the interpretations of all domains,
  i.e. if it is true for any models $\mathcal{M}$ and any valuation v
- A formula is "**satisfiable**" if it is true for at least one interpretation of one domain.
  i.e. if there exists a model $\mathcal{M}$ such that it is true
- A formula is "**unsatisfiable**" if it is never true in any domain, i.e. if it is false in all interpretations i of all domains $\mathcal{D}$. i.e. if it is false in all models $\mathcal{M}$

Logical consequence
Semantic consequence
Entailment

- A is a **semantic consequence** of B if A is true for all the interpretations where B is true, i.e. for all models if $I_{mv}(B)$ *then* $I_{mv}(A)$
  - Example: (A→B) is a semantic consequence of B.

Notation

$\models$ **A** means A is a valid formula

B $\models$ **A** means A is a semantic consequence of B

L I P 6 C N R S

Deduction Theorem

Deduction Theorem: B is a semantic consequence of A ($A \models B$) *if and only if* A→B is valid ($\models A \rightarrow B$)

Equivalent definitions:

- $A \models$ if and only if $\models (\neg A)$
- $A \models B$ if and only if $\models A \rightarrow B$
- $\models A$ if and only if $(\neg A) \models A$

L
I

Provability
Using a **Formal System**

- *Recall*: a formal system is composed of
  - A formal language
  - A set of axioms
  - A set of inference rules

- Example of formal systems:
  - Hilbert-Ackermann
  - Natural Deduction
  - Sequent Calculus
  - …

L

I

A Formal System for the Propositional Logic (Hilbert-Ackermann)

- *Recall*: a formal system is composed of
  - A formal language
  - A set of axioms
  - A set of inference rules
- <u>Formal system for the propositional logic</u>:
  - **Formal language**: set A of atoms, one unary connector, « $\neg$ », one binary connector, « $\rightarrow$ »
  - **Axioms**: formulae which are obtained by replacing A, B et C by any propositional logic formula in the following axiom schemas:

    SA1: $(A \rightarrow (B \rightarrow A))$

    SA2: $((A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)))$

    SA3: $((\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A))$
  - **Inference rule**: « *Modus Ponens* » **$A, (A \rightarrow B) \vdash B$**

L I P 6 C N R S

Theorems

Definition: any formula which is derived from the axioms by iteratively applying inference rules is a *theorem*.

Notation: $\vdash \mathbf{A}$ means A is a theorem

Example: $\vdash \mathbf{(A \rightarrow A)}$

***Proof*:**

1- $(A \rightarrow (\mathbf{(A \rightarrow A)} \rightarrow A))$ SA1

2- $((A \rightarrow (\mathbf{(A \rightarrow A)} \rightarrow A)) \rightarrow ((A \rightarrow \mathbf{(A \rightarrow A)}) \rightarrow (A \rightarrow A)))$ SA2

3- $((A \rightarrow \mathbf{(A \rightarrow A)}) \rightarrow (A \rightarrow A))$ Modus Ponens 2, 1

4- $(A \rightarrow \mathbf{(A} \rightarrow A))$ SA1

5- $\mathbf{(A \rightarrow A)}$ Modus Ponens 4, 3

L I P 6 C N R S

Demonstration

Definition: a *proof* of a theorem A is a finite sequence of formulae F_0, F_1, …F_n such that

- $F_n = A$
- $\forall i \in [0,n]$ F_i is either an axiom, or obtained by applying the *modus ponens* to the two formulae F_j and F_k where j and k < i

Deduction theorem

Theorem: $A_1, \dots A_{n-1} \vdash (A_n \rightarrow B)$ ssi $A_1, \dots, A_{n-1}, A_n \vdash B$

***Proof (direction1)*:**

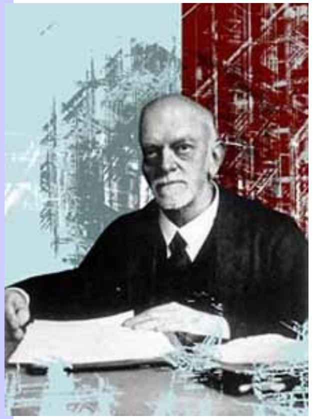
1. $(A_n \rightarrow B)$ Hypothesis 1
2. A_n Hypothesis 2
3. B *Modus Ponens* 1, 2

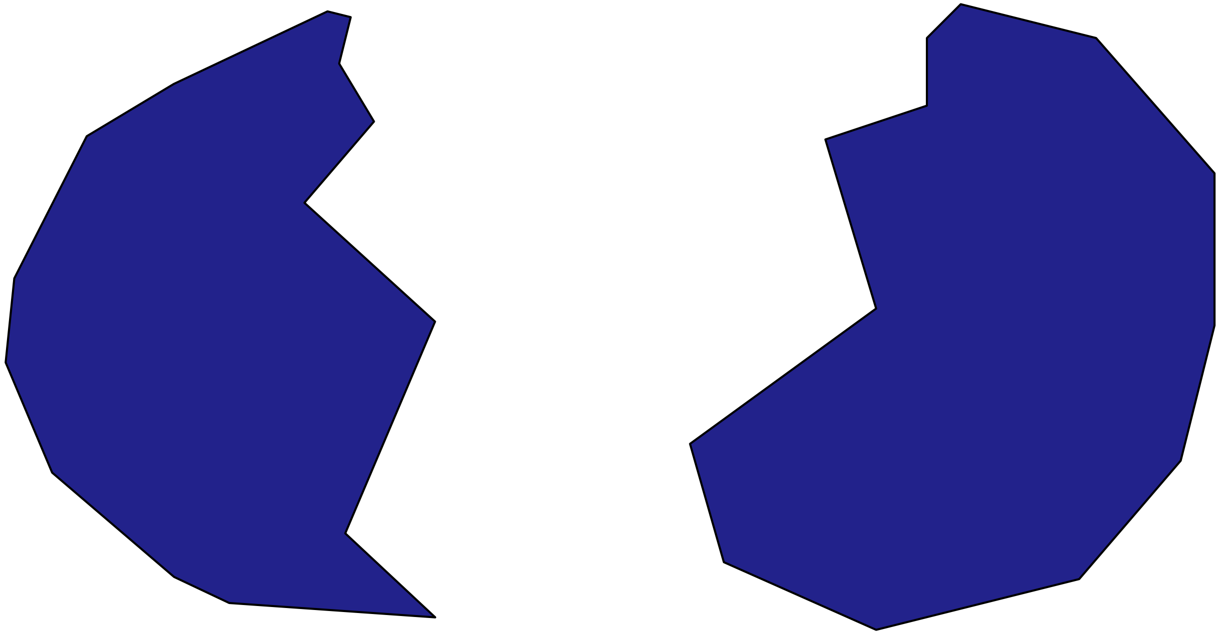
***Proof (direction 2)*:** four possibilities has to be investigated

1. B is an axiom
2. B is one of the hypotheses $A_1, \dots A_{n-1}$
3. B is the hypothesis A_n
4. B is obtained by applying the *modus ponens* to $(G \rightarrow B)$ and G (*proof by induction on the size of the demonstration*)

L I P 6 C N R S

Notion of symbolic system





Consistency: each description of the symbolic system corresponds to an object in the reality, i.e. $\forall A$ **if** $\vdash A$ **then** $\models A$

Completeness: each object of the reality can be described in the symbolic system $\forall A$ **if** $\models A$ **then** $\vdash A$

Automatic Theorem Proving

Tableau Method

Resolution in Propositional Logic

Unification

Resolution in First Order Logic

LIP6

CNRS

Automatic Theorem Proving

Tableau Method

Resolution in Propositional Logic

Unification

Resolution in First Order Logic

Tableau Method in Propositional Logic

Step 1: normalization – transformation into NNF – Negative Normal Form

The negations occurs only before atomic propositions

Push negations inwards the formulas using de Morgan laws and the suppression of the double negation

$\neg\neg\varphi \equiv \varphi$

$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$

$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$

Tableau Method in Propositional Logic (following)

Step 2: build a tableau

Root: the formula under *NNF – Negative Normal Form*

Build successors of T using two rules $R_\wedge$ et $R_\vee$.

We stop when we are unable to apply any rule

$$R_\wedge \ \frac{T}{T \cup \{\varphi, \psi\}} \quad \boxed{\begin{array}{l}\text{If } \varphi\wedge\psi \in T \\ \text{and } (\varphi \notin T \\ \quad\text{or } \psi \notin T)\end{array}}$$

$$R_\vee \ \frac{T}{T\cup\{\varphi\} \ \ | T\cup\{\psi\}} \quad \boxed{\begin{array}{l}\text{If } \varphi\vee\psi \in T \\ \text{and } \varphi \notin T \\ \text{and } \psi \notin T\end{array}}$$

LIP6 CNRS

Tableau Method

Definitions:

- Contradictory tableau: it contains simultaneously p and ¬p (it is said that it contains a *clash*)
- Complete tableau: no rule can be applied
- A tableau is said to be *closed* if it contains a clash, *open* else

On applies systematically the rules on all the tableau.

The answer is said to be “satisfiable” if one of the generated tableau is open, “unsatisfiable” else.

$$R_{\wedge}\ \frac{T}{T \cup \{\varphi, \psi\}}$$

If $\varphi\wedge\psi \in T$ and ($\varphi \notin T$ or $\psi \notin T$)

$$R_{\vee}\ \frac{T}{T\cup\{\varphi\} \quad | T\cup\{\psi\}}$$

If $\varphi\vee\psi \in T$ and $\varphi \notin T$ and $\psi \notin T$

Properties

1. There is finite sequence of applications of rules
$\mathcal{S}_0 = T \rightarrow \mathcal{S}_1 \rightarrow \mathcal{S}_2 \rightarrow \ldots$
2. If $\mathcal{S}$' is obtained from a finite set $\mathcal{S}$ of tableau by applying transformation rules, then $\mathcal{S}$' is consistent iff $\mathcal{S}$ is.
3. All clash tableau are unconsistent
4. All complete and open tableau is consistent

L I P 6 C N R S

Third step

- T being a complete tableau

The model M[T] that satisfies φ is built as follows:

If p, a proposition, belongs to T, then p is true in M[T],

else p is false

Example

$\varphi = \neg(\neg p \wedge q) \wedge \neg(r \wedge \neg q)$

φ rewrite: $\varphi = (p \vee \neg q) \wedge (\neg r \vee q)$

$(p \vee \neg q) \wedge (\neg r \vee q)$

$R_\wedge$

$(p \vee \neg q) \wedge (\neg r \vee q)$
$(\neg r \vee q)$
$(p \vee \neg q)$

$R_\vee$ **Branch 1**

$(p \vee \neg q) \wedge (\neg r \vee q)$
$(\neg r \vee q)$
$(p \vee \neg q)$
$\neg q$

$R_\vee$

Clash

$(p \vee \neg q) \wedge (\neg r \vee q)$
$(\neg r \vee q)$
$(p \vee \neg q)$
$\neg q$
q

$R_\vee$

Complete

$(p \vee \neg q) \wedge (\neg r \vee q)$
$(\neg r \vee q)$
$(p \vee \neg q)$
$\neg q$
$\neg r$

L
I
P
6

C
N
R
S

Example - follows

$\varphi = \neg(\neg p \wedge q) \wedge \neg(r \wedge \neg q)$

- Réécriture: $\varphi = (p \vee \neg q) \wedge (\neg r \vee q)$

$(p \vee \neg q) \wedge (\neg r \vee q)$

$R_\wedge$

$(p \vee \neg q) \wedge (\neg r \vee q)$
$(\neg r \vee q)$
$(p \vee \neg q)$

$R_\vee$ **Branch 2**

Complete

$(p \vee \neg q) \wedge (\neg r \vee q)$
$(\neg r \vee q)$
$(p \vee \neg q)$
p
q

$R_\vee$

$(p \vee \neg q) \wedge (\neg r \vee q)$
$(\neg r \vee q)$
$(p \vee \neg q)$
p

$R_\vee$

Complete

$(p \vee \neg q) \wedge (\neg r \vee q)$
$(\neg r \vee q)$
$(p \vee \neg q)$
p
$\neg r$

Conclusion

$\varphi = \neg(\neg p \wedge q) \wedge \neg$
$(r \wedge \neg q)$

φ owns three models:

1. $\neg q, \neg r$
2. $p, \neg r$
3. p, q

Another application of the tableau method

$F = \neg(v \wedge n\_v) \wedge (n\_v \vee v) \wedge (n\_v \vee \neg(p \vee a))$

Transformation – *NNF*:

$$F = (\neg v \vee \neg n\_v) \wedge (n\_v \vee v) \wedge (n\_v \vee (\neg p \wedge \neg a))$$

$$F = \boxed{\begin{array}{l}(\neg v \vee \neg n\_v) \wedge (n\_v \vee v) \wedge \\ (n\_v \vee \neg p) \wedge \quad (n\_v \vee \neg a)\end{array}}$$

Tableau Method

$(\neg v \vee \neg n\_v) \wedge (n\_v \vee v) \wedge (n\_v \vee \neg p) \wedge (n\_v \vee \neg a)$

$3 \times R_{\wedge}$

$(\neg v \vee \neg n\_v) \wedge (n\_v \vee v) \wedge (n\_v \vee \neg p) \wedge (n\_v \vee \neg a)$
$(\neg v \vee \neg n\_v)$
$(n\_v \vee v)$
$(n\_v \vee \neg p)$
$(n\_v \vee \neg a)$

$R_{\vee}$ **Branch 1**

... $(\neg v \vee \neg n\_v)$
$(n\_v \vee v)$
$(n\_v \vee \neg p)$
$(n\_v \vee \neg a)$
n_v

$R_{\vee}$ **Branch 2**

... $(\neg v \vee \neg n\_v)$
$(n\_v \vee v)$
$(n\_v \vee \neg p)$
$(n\_v \vee \neg a)$
$\neg a$

LIP6 CNRS

Tableau Method – *left part*

Model 1

$n\_v, \neg v$

$\dots (\neg v \lor \neg n\_v)$
$(n\_v \lor v)$
$(n\_v \lor \neg p)$
$(n\_v \lor \neg a)$
n_v

$R_\lor$ **Branch 1**

$\dots (\neg v \lor \neg n\_v)$
$(n\_v \lor v)$
$(n\_v \lor \neg p)$
$(n\_v \lor \neg a)$
n_v
$\neg v$

Complete

$R_\lor$ **Branch 2**

$\dots (\neg v \lor \neg n\_v)$
$(n\_v \lor v)$
$(n\_v \lor \neg p)$
$(n\_v \lor \neg a)$
n_v
$\neg n\_v$

Clash

Tableau Method – *right part*

Model 2

n_v, $\neg v$, $\neg a$

Root:

... $(\neg v \vee \neg n\_v)$
$(n\_v \vee v)$
$(n\_v \vee \neg p)$
$(n\_v \vee \neg a)$
$\neg a$

$R_{\vee}$ **Branch 1**

... $(\neg v \vee \neg n\_v)$
$(n\_v \vee v)$
$(n\_v \vee \neg p)$
$(n\_v \vee \neg a)$
$\neg a$
$\neg v$

$R_{\vee}$ **Branch 1**

... $\neg a$
$\neg v$
v

Clash

$R_{\vee}$ **Branch 2**

... $\neg a$
$\neg v$
n_v

Complete

$R_{\vee}$ **Branch 2**

... $(\neg v \vee \neg n\_v)$
$(n\_v \vee v)$
$(n\_v \vee \neg p)$
$(n\_v \vee \neg a)$
$\neg a$
$\neg n\_v$

$R_{\vee}$ **Branch 1**

... $\neg a$
$\neg n\_v$
v

$R_{\vee}$ **Branch 2**

... $\neg a$
$\neg n\_v$
n_v

clash

L I P 6 C N R S

Following – *right*

... $(\neg v \lor \neg n\_v)$
$(n\_v \lor v)$
$(n\_v \lor \neg p)$
$(n\_v \lor \neg a)$
$\neg a$
$\neg n\_v$
v

Model 3
$\neg$ n_v, v, $\neg$ a, $\neg$ p

$R_\lor$ **Branche 1**

... $\neg a$
$\neg$ n_v
v
n_v

Clash

$R_\lor$ **Branche 2**

... $\neg a$
$\neg$n_v
v
$\neg$p

Complete

Recap

- We obtain three models for F:

$F = \neg(v \wedge n\_v) \wedge (n\_v \vee v) \wedge (n\_v \vee \neg(p \vee a))$

Model 1: $n\_v, \neg v$

Model 2: $n\_v, \neg v, \neg a$

Model 3: $\neg n\_v, v, \neg a, \neg p$

Tableau method in propositional logic Generalization

Goal: search a model for a set of formulas $\mathcal{F}$

- **Initialisation**: create a non marked root node initialized with $\mathcal{F}$
- **Iterative decomposition**: choose iteratively all the non treated nodes (set of formulae with $\mathcal{F}$) and mark them once treated
  - If the node contained complementary literals, mark the node as closed (clash)
  - Else, if all the formulae associated to the node are propositional variables (i.e. atom or atom negation), mark it as opened
  - Else, choose a non propositional formula F belonging to the current node
    - If F is **α-type**: create one new node non marked with the set $\mathcal{F}-\{F\}\cup\{\alpha_1, \alpha_2\}$
    - If F is β-type: create two new non marked nodes with the sets $\mathcal{F}-\{F\}\cup\{\beta_1\}$ and $\mathcal{F}-\{F\}\cup\{\beta_2\}$

Decomposition rules

Formula α	α_1	α_2
$\neg\neg\varphi$	φ	
$\varphi_1 \wedge \varphi_2$	φ_1	φ_2
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\neg(\varphi_1 \rightarrow \varphi_2)$	φ_1	$\neg\varphi_2$
$\varphi_1 \leftrightarrow \varphi_2$	$\varphi_1 \rightarrow \varphi_2$	$\varphi_2 \rightarrow \varphi_1$

Formula β	β_1	β_2
$(\varphi_1 \vee \varphi_2)$	φ_1	φ_2
$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$(\varphi_1 \rightarrow \varphi_2)$	$\neg\varphi_1$	φ_2
$\neg(\varphi_1 \leftrightarrow \varphi_2)$	$\neg(\varphi_1 \rightarrow \varphi_2)$	$\neg(\varphi_2 \rightarrow \varphi_1)$

Formula α	α_1	α_2
$\neg\neg\varphi$	φ	
$\varphi_1 \wedge \varphi_2$	φ_1	φ_2
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\neg(\varphi_1 \rightarrow \varphi_2)$	φ_1	$\neg\varphi_2$
$\varphi_1 \leftrightarrow \varphi_2$	$\varphi_1 \rightarrow \varphi_2$	$\varphi_2 \rightarrow \varphi_1$

Example

$$\varphi = \neg(\neg p \wedge q) \wedge \neg(r \wedge \neg q)$$

$$\boxed{\neg(\neg p \wedge q) \wedge \neg(r \wedge \neg q)}$$

Type α

$$\downarrow$$

$$\boxed{\begin{array}{l}\neg(\neg p \wedge q)\\ \neg(r \wedge \neg q)\end{array}}$$

Formula β	β_1	β_2
$(\varphi_1 \vee \varphi_2)$	φ_1	φ_2
$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$(\varphi_1 \rightarrow \varphi_2)$	$\neg\varphi_1$	φ_2
$\neg(\varphi_1 \leftrightarrow \varphi_2)$	$\neg(\varphi_1 \rightarrow \varphi_2)$	$\neg(\varphi_2 \rightarrow \varphi_1)$

L I P 6 C N R S

Example

$\varphi = \neg(\neg p \wedge q) \wedge \neg(r \wedge \neg q)$

Formula α	α_1	α_2
$\neg\neg\varphi$	φ	
$\varphi_1 \wedge \varphi_2$	φ_1	φ_2
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\neg(\varphi_1 \rightarrow \varphi_2)$	φ_1	$\neg\varphi_2$
$\varphi_1 \leftrightarrow \varphi_2$	$\varphi_1 \rightarrow \varphi_2$	$\varphi_2 \rightarrow \varphi_1$

$\neg(\neg p \wedge q) \wedge \neg(r \wedge \neg q)$

Type α

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

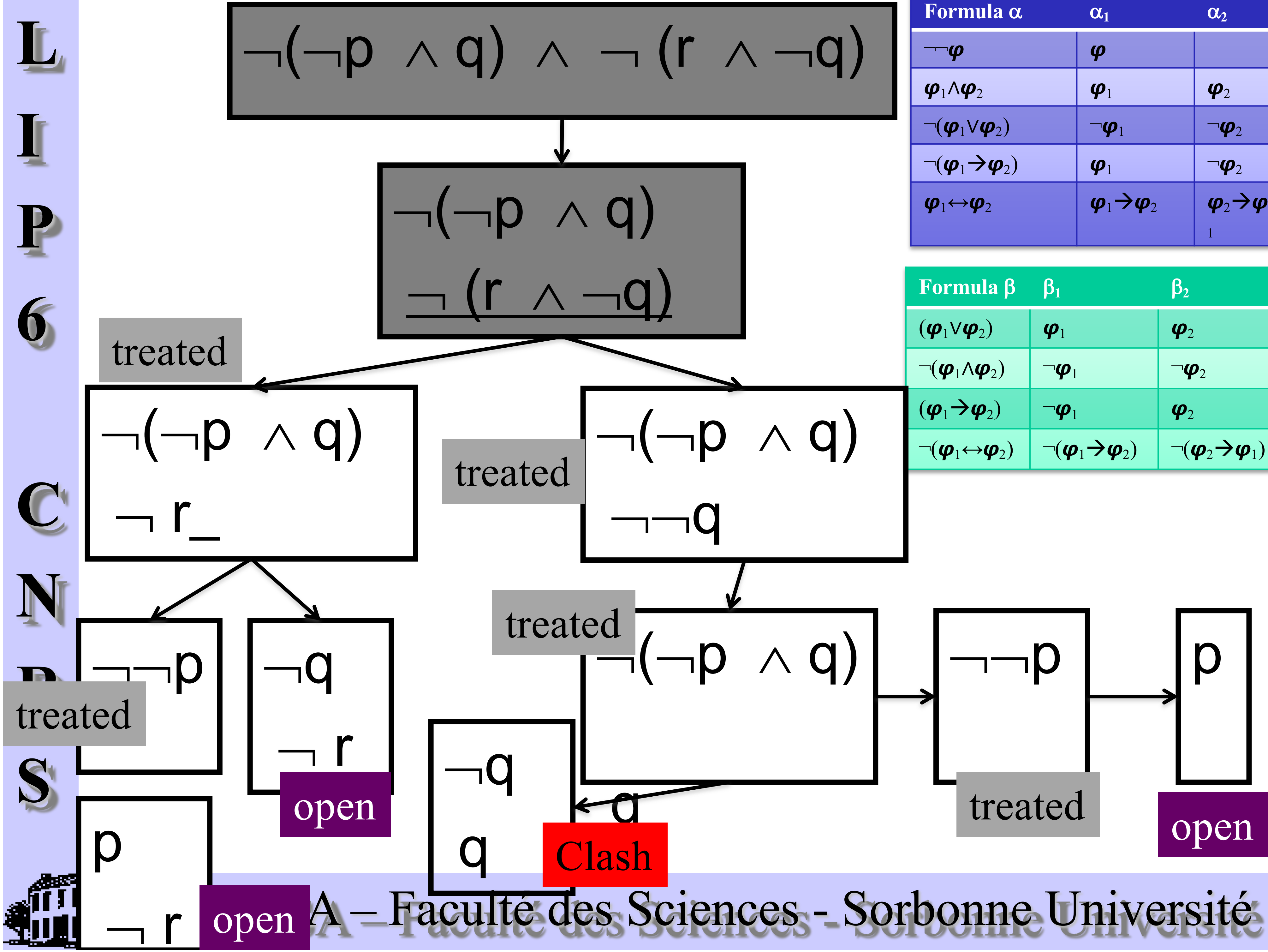
$\underline{\neg(r \wedge \neg q)}$

Type β

$\neg(\neg p \wedge q)$	$\neg(\neg p \wedge q)$
$\neg r$	$\neg\neg q$

Formula β	β_1	β_2
$(\varphi_1 \vee \varphi_2)$	φ_1	φ_2
$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$(\varphi_1 \rightarrow \varphi_2)$	$\neg\varphi_1$	φ_2
$\neg(\varphi_1 \leftrightarrow \varphi_2)$	$\neg(\varphi_1 \rightarrow \varphi_2)$	$\neg(\varphi_2 \rightarrow \varphi_1)$

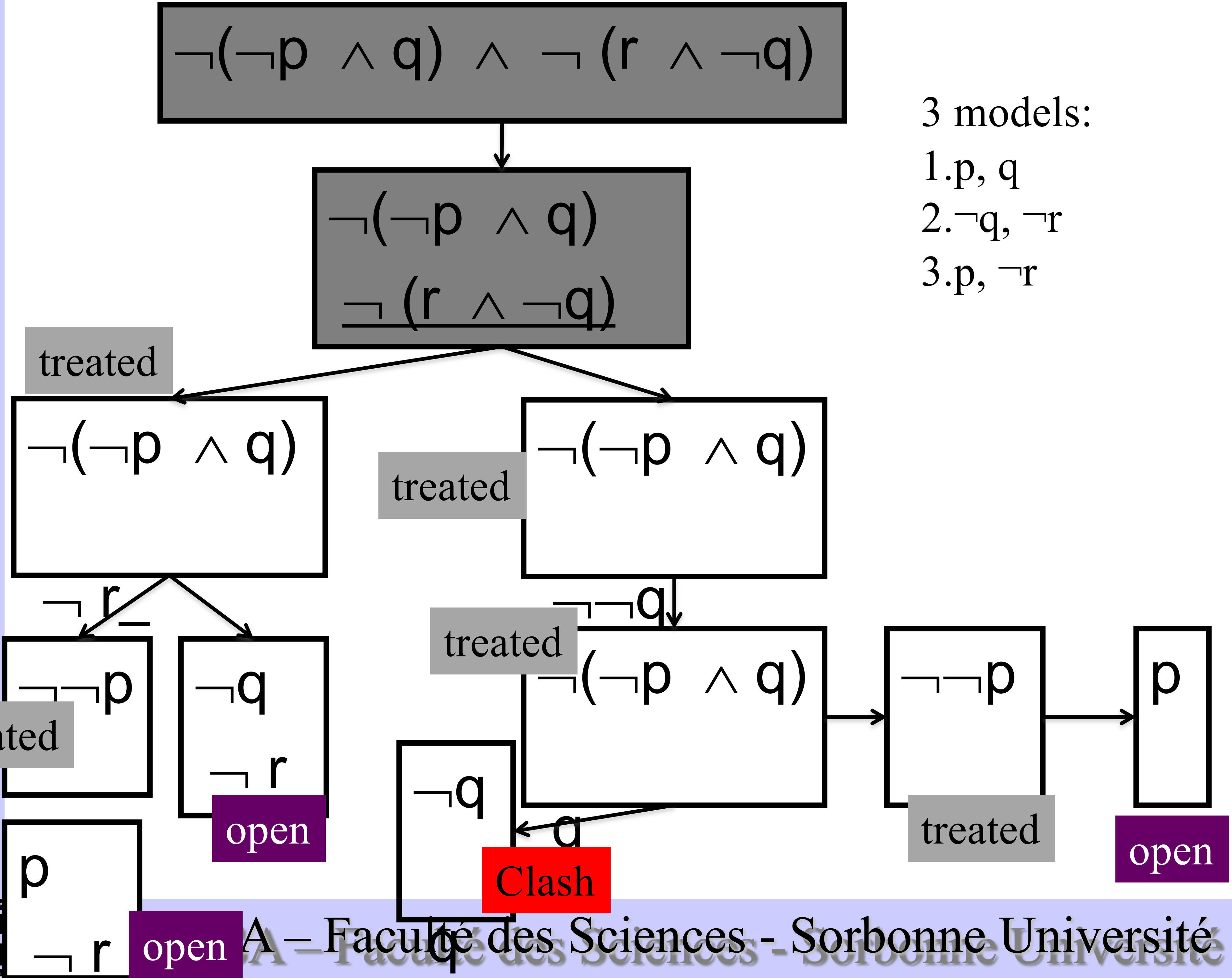
LIP6 CNRS



Formula α	α_1	α_2
$\neg\neg\varphi$	φ	
$\varphi_1\wedge\varphi_2$	φ_1	φ_2
$\neg(\varphi_1\vee\varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\neg(\varphi_1\rightarrow\varphi_2)$	φ_1	$\neg\varphi_2$
$\varphi_1\leftrightarrow\varphi_2$	$\varphi_1\rightarrow\varphi_2$	$\varphi_2\rightarrow\varphi_1$

Formula β	β_1	β_2
$(\varphi_1\vee\varphi_2)$	φ_1	φ_2
$\neg(\varphi_1\wedge\varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$(\varphi_1\rightarrow\varphi_2)$	$\neg\varphi_1$	φ_2
$\neg(\varphi_1\leftrightarrow\varphi_2)$	$\neg(\varphi_1\rightarrow\varphi_2)$	$\neg(\varphi_2\rightarrow\varphi_1)$

LIP6 CNRS



3 models:

1.p, q

2.¬q, ¬r

3.p, ¬r

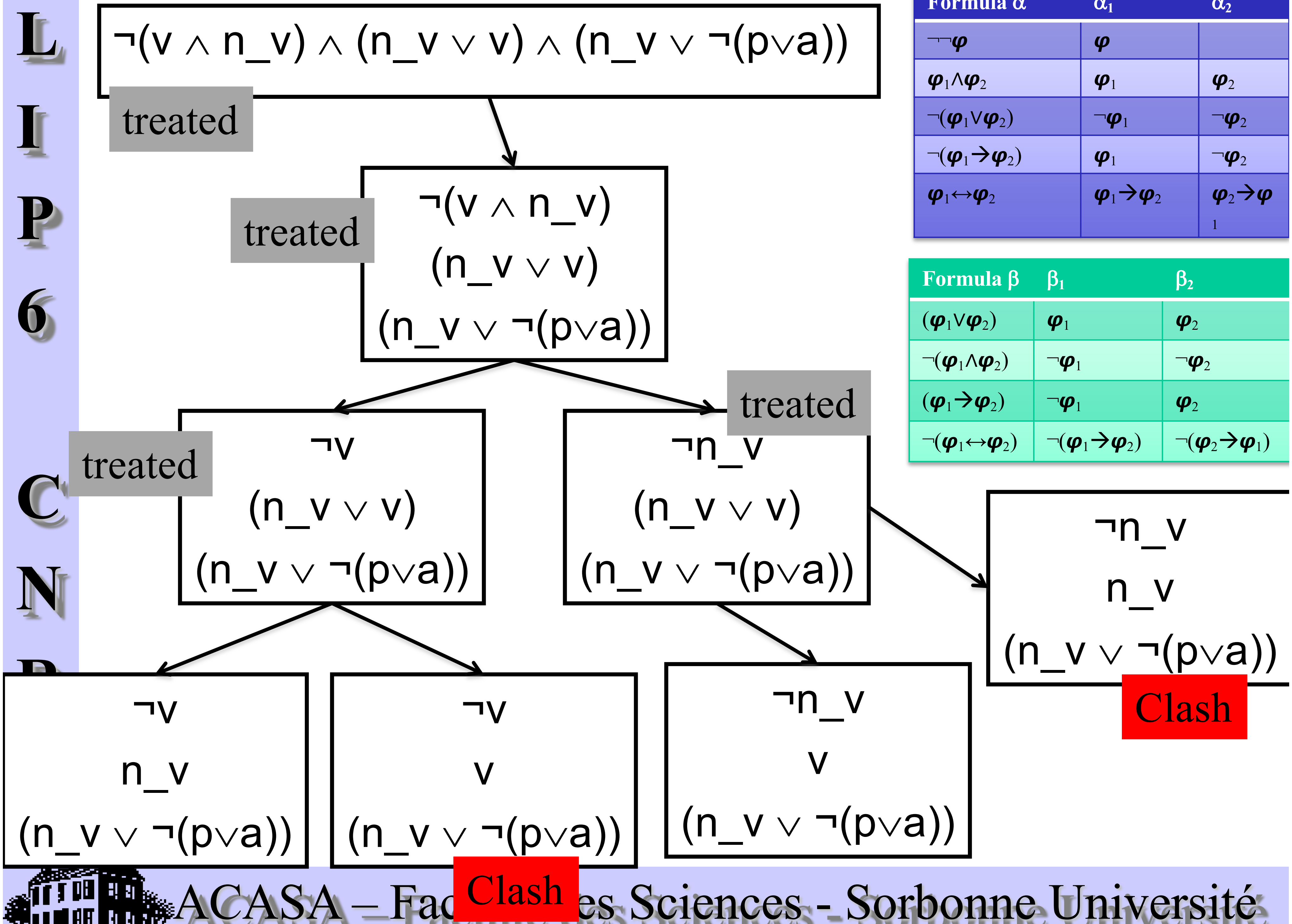
Another example

$\neg(v \wedge n\_v) \wedge (n\_v \vee v) \wedge (n\_v \vee \neg(p \vee a))$

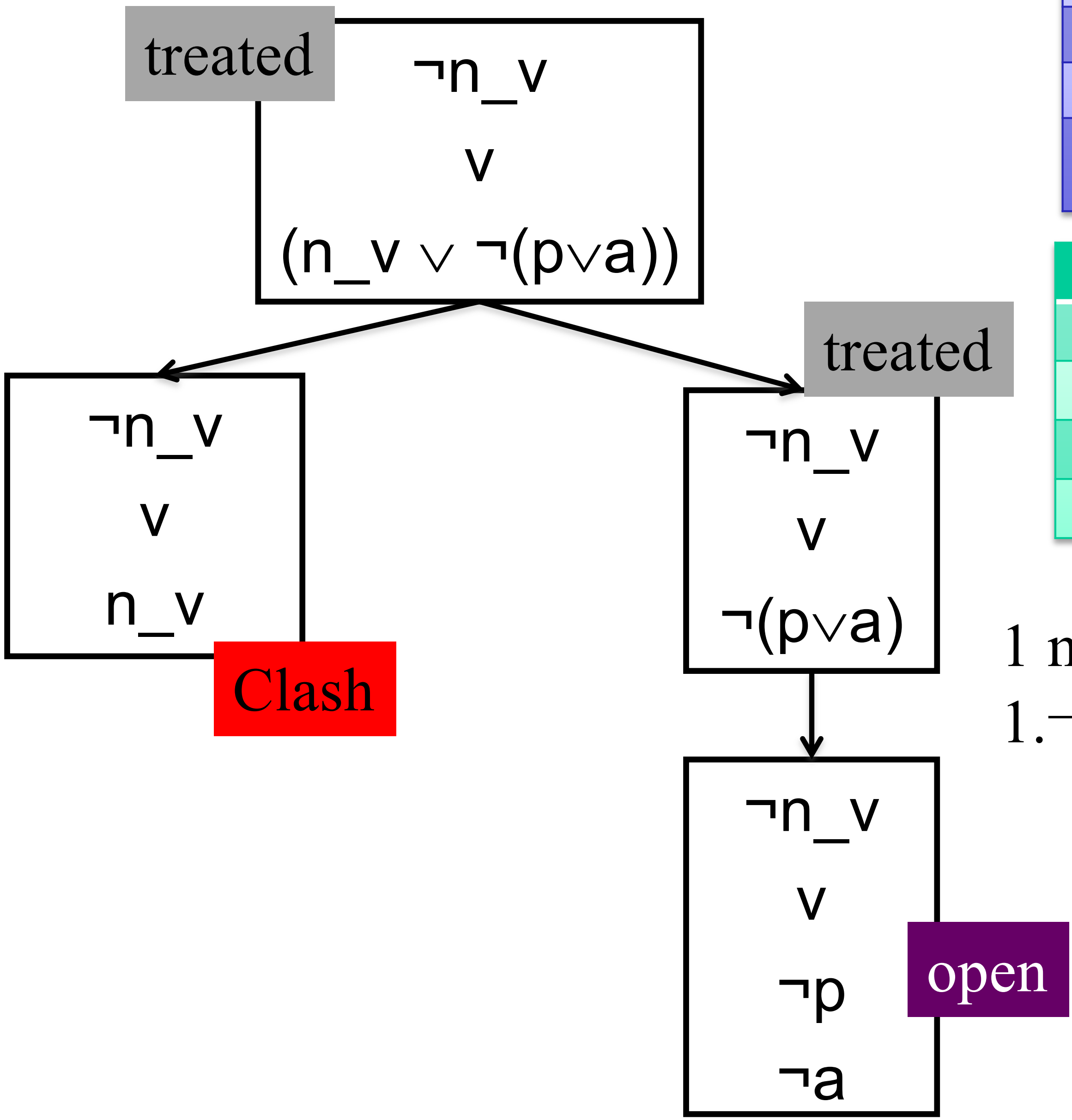
LIP6 CNR

Formula α	α_1	α_2
$\neg\neg\varphi$	φ	
$\varphi_1 \wedge \varphi_2$	φ_1	φ_2
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\neg(\varphi_1 \to \varphi_2)$	φ_1	$\neg\varphi_2$
$\varphi_1 \leftrightarrow \varphi_2$	$\varphi_1 \to \varphi_2$	$\varphi_2 \to \varphi_1$

Formula β	β_1	β_2
$(\varphi_1 \vee \varphi_2)$	φ_1	φ_2
$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$(\varphi_1 \to \varphi_2)$	$\neg\varphi_1$	φ_2
$\neg(\varphi_1 \leftrightarrow \varphi_2)$	$\neg(\varphi_1 \to \varphi_2)$	$\neg(\varphi_2 \to \varphi_1)$



Right node



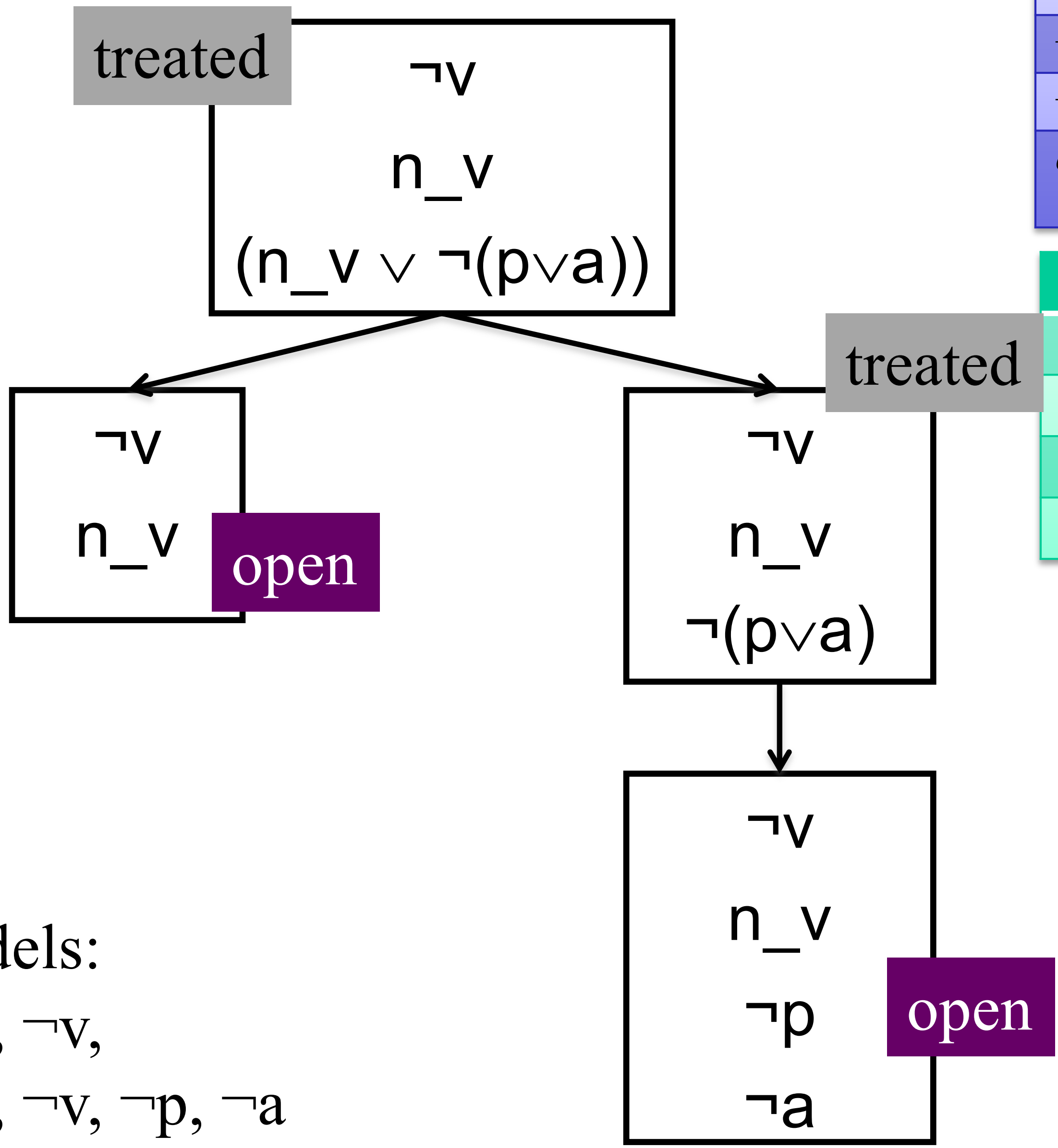
Formula α	α_1	α_2
$\neg\neg\varphi$	φ	
$\varphi_1 \wedge \varphi_2$	φ_1	φ_2
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\neg(\varphi_1 \rightarrow \varphi_2)$	φ_1	$\neg\varphi_2$
$\varphi_1 \leftrightarrow \varphi_2$	$\varphi_1 \rightarrow \varphi_2$	$\varphi_2 \rightarrow \varphi_1$

Formula β	β_1	β_2
$(\varphi_1 \vee \varphi_2)$	φ_1	φ_2
$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$(\varphi_1 \rightarrow \varphi_2)$	$\neg\varphi_1$	φ_2
$\neg(\varphi_1 \leftrightarrow \varphi_2)$	$\neg(\varphi_1 \rightarrow \varphi_2)$	$\neg(\varphi_2 \rightarrow \varphi_1)$

1 model:

1. ¬n_v, v, ¬p, ¬a

Left node



Formula α	α_1	α_2
$\neg\neg\varphi$	φ	
$\varphi_1 \wedge \varphi_2$	φ_1	φ_2
$\neg(\varphi_1 \vee \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\neg(\varphi_1 \rightarrow \varphi_2)$	φ_1	$\neg\varphi_2$
$\varphi_1 \leftrightarrow \varphi_2$	$\varphi_1 \rightarrow \varphi_2$	$\varphi_2 \rightarrow \varphi_1$

Formula β	β_1	β_2
$(\varphi_1 \vee \varphi_2)$	φ_1	φ_2
$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$(\varphi_1 \rightarrow \varphi_2)$	$\neg\varphi_1$	φ_2
$\neg(\varphi_1 \leftrightarrow \varphi_2)$	$\neg(\varphi_1 \rightarrow \varphi_2)$	$\neg(\varphi_2 \rightarrow \varphi_1)$

2 models:

1. n_v, ¬v,
2. n_v, ¬v, ¬p, ¬a

Automatic Theorem Proving

Tableau Method

Resolution in Propositional Logic

Unification

Resolution in First Order Logic

clause

- **Definition**:
  - A **literal** is either an atom or its negation:

    Example: ¬even(72), odd(72), successor(72, 73)
  - A **clause** is a disjunction of literals

    Example: ¬even(X) ∨ odd(successor(X))

Remark: a clause is a logical entailment (implication) because (¬A ∨ B) is equivalent to (A ⊃ B)

Example: even(X) ⊃ odd(successor(X))

ACASA – Faculté des Sciences - Sorbonne Université

Clausal Form

Theorem: any closed formula F can be transformed into a logically equivalent conjunction of clauses

Example:

$\forall$x square(x) $\equiv$ $\exists$y multiply(y, y, x)

Can be transformed as a conjunction of two clauses:

$\neg$square(x) $\vee$ multiply(r(x), r(x), x) et

square(x) $\vee$ $\neg$ multiply(r(x), r(x), x)

Resolution Rule in Propositional Logic

- Been given two clauses C_1 and C_2
- Been given an atomic proposition such that $A \in C_1$ and $\neg A \in C_2$
- The resolution of C_1 and C_2 by A and ¬A is:

$$C = res(C_1, C_2; A, \neg A)$$
$$= [C_1 - \{A\}] \vee [C_2 - \{\neg A\}]$$

Example: if $C_1 = \neg man \vee mortal$,
$C_2 = \neg socrate \vee man$, $C_3 = socrate$ and
$C_4 = \neg mortal$

$$C = res(C_3, C_2; socrate, \neg socrate) = man$$

$$C' = res(C_1, C_4; mortal, \neg mortal) = \neg man$$

Resolution (*example*)

$C_1 = \neg man \vee mortal = man \supset mortal$

$C_2 = \neg socrate \vee man = socrate \supset man$

$C_3 = socrate$, $C_4 = \neg mortal$,

$C = res(C_3, C_2; socrate, \neg socrate) = man$

$C' = res(C_1, C_4 ; mortal, \neg mortal) = \neg man$

- The resolution is more general than the *Modus Ponens* ($A, A \supset B/B$) and the *Modus Tolens* ($\neg B, A \supset B/\neg A$)

Generality of the Resolution

- To prove $S \models C$, it is sufficient to prove that $S \cup \{\neg C\}$ is contradictory, i.e. that $S \cup \{\neg C\} \models \square$

$\square$ denotes the empty clause, i.e. the false

- **Theorem**: $S \models C$ if and only if it is possible to derive the empty clause by iterative application of the resolution rule on $S \cup \{\neg C\}$

L
I
P
6

C
N
R
S

Example

- Consider S = $\{C_1, C_2, C_3\}$

$C_1 = \neg man \vee mortal = man \supset mortal$

$C_2 = \neg socrate \vee man = socrate \supset man$

$C_3 =$ socrate, $C_4 = \neg mortal$,

- To prove S $\models$ mortel, it is sufficient to derive the empty clause, i.e. $\square$, from $S \cup \{C_4\}$

C_5: $\neg$man res(C_1, C_4; mortal, $\neg$mortal)

C_6: $\neg$socrate res(C_2, C_5; man, $\neg$man)

C_5: $\square$ res(C_3, C_6; socrate, $\neg$socrate)

QED

L
I
P
6

C
N
R
S

Automatic Theorem Proving

Tableau Method

Resolution in Propositional Logic

Unification

Resolution in First Order Logic

L I P 6

C N R S

Substitution

Definition:

- Being $\mathcal{V}$ the set of variables.
- Being $\mathcal{F}$ the set of functions.
- Being $\mathcal{T}$ the set of terms built on $\mathcal{V}$ and $\mathcal{F}$.

A *substitution* σ is an application from $\mathcal{V}$ to the set of terms $\mathcal{T}$ which is the identity *almost everywhere*

A *substitution* is characterized by a finite set of pairs "x_i / t_i", x_i being a variable and t_i a term.

It is denoted $\sigma = \{v_1/t_1, \ldots, v_n/t_n\}$

Example: $\sigma = \{x/3, y/(u - 7)\}$

Term instance

Definition:

– Being E a term of $\mathcal{T}$ (set of terms built on $\mathcal{V}$ and $\mathcal{F}$)

– Being a substitution σ of variables $\mathcal{V}$ by terms of $\mathcal{T}$ $\sigma = \{v_1/t_1, \ldots, v_n/t_n\}$

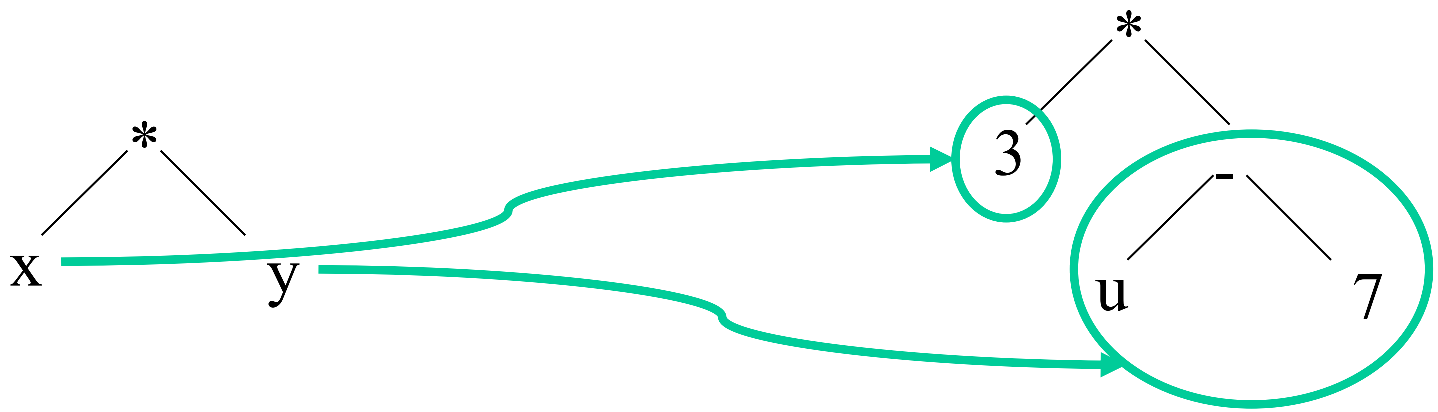
Being $E\sigma$ the expression built by replacing each occurrence of free variables v_i of E by t_i.

$E\sigma$ is called an *instance* of E

Term instance (following)

Example: if $\sigma = \{x/3, y/(u - 7)\}$ et $E = (x*y)$

then $E\sigma = (3 * (u - 7))$ is an instance of E



Substitution composition

- Being $\theta=\{t_1/x_1,\dots, t_n/x_n\}$ and $\lambda=\{u_1/y_1,\dots, u_m/y_m\}$ two substitutions.

- The *composition* of θ and λ is the substitution $\lambda \bigcirc \theta$ which is obtained from the set $\boldsymbol{\{t_1\lambda/x_1,\dots, t_n\lambda/x_n, u_1/y_1,\dots, u_m/y_m\}}$ by deleting all the $\boldsymbol{t_j\lambda/x_j}$ such that $\boldsymbol{t_j\lambda=x_j}$ and all the $\boldsymbol{u_i/y_i}$ such that $\boldsymbol{y_i\in\{x_1,\dots, x_n\}}$

Substitution composition

- The *composition* of two substitution θ and λ is the substitution $\lambda \bigcirc \theta$ which is obtained from $\{t_1\lambda/x_1,\ldots, t_n\lambda/x_n, u_1/y_1,\ldots, u_m/y_m\}$ by deleting all the $t_j\lambda/x_j$ such that $t_j\lambda=x_j$ and all the u_i/y_i such that $y_i \in \{x_1,\ldots, x_n\}$

Example: $\theta=\{t_1/x_1, t_2/x_n\}=\{f(y)/x, z/y\}$ et $\lambda=\{u_1/y_1, u_2/y_2, u_3/y_3\} = \{a/x, b/y, y/z\}$

$\{t_1\lambda/x_1,\ldots, t_n\lambda/x_n, u_1/y_1,\ldots, u_m/y_m\} =$
$\{f(b)/x, \underline{y/y}, \underline{a/x}, \underline{b/y}, y/z\} = \{f(b)/x, y/z\}$

$$\theta \bigcirc \lambda = \{f(b)/x, y/z\}$$

L I P 6 C N R S

LIP6 CNRS

Pattern matching

Is a term an instance of another?

Pattern matching: the term t_1 match with the term t_2 if and only if there exists a substitution σ such that: $\mathbf{t_1\sigma = t_2}$