L I P 6 C N R S

Logique et représentation des connaissances
Spécialités DAC et Android

Enseignants:

Nicolas Maudet (resp. Android)

Jean-Gabriel Ganascia (resp. DAC)

Amal El Fallah

Marie-Jeanne Lesot

Aurélia Léon

Gauvain Bourgne

L I P 6 C N R S

Informations Générales

- **Cours:** amphithéâtre 55a, mercredi, 13H45-15H45
- **Premier cours:** 16 septembre 2015
- **Dernier cours**: 2 décembre 2015
- **Examens répartis:**
  - **Examen réparti n°1: 4 novembre 2015**
  - **Examen réparti n°2: 6 janvier 2016**
- **TD/TME**
  - **Lundi après-midi (G1 et G2),**
  - **mardi après-midi (G3),**
  - **jeudi matin (G4)**
- **Piazza:** https://piazza.com/upmc.fr/fall2015/m1lrc/

Organisation TD/TME

- **Groupe 1**:

TD: lundi 13H45-15H45 salle T15

TME: lundi 16H-18H salle 302, couloir 14/15

- **Groupe 2**:

TD: lundi 13H45-15H45 salle 107, couloir 13/14

TME: lundi 16H-18H salle 303, couloir 14/15

- **Groupe 3**:

TD: mardi 13H45-15H45 salle en attente

TME: mardi 16H-18H salle 303, couloir 14/15

- **Groupe 4**:

TME: jeudi 8H30-10H30 salle 408, couloir 14/15

TD: jeudi 10H45-12H45 salle en attente

L I P 6 C N R S

Cours LRC: logique(s) et représentation des connaissances

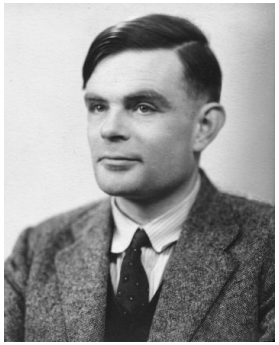
1. Introduction à la logique des propositions et des prédicats du premier ordre. Règles de résolution. Méthode des tableaux.
2. Représentations sémantiques, graphes conceptuels.
3. Logiques de description (syntaxe et sémantique).
4. Logiques de description: raisonnement automatique (méthode des tableaux et subsomption structurelle)

Logiques modales

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

Représentations du temps

5. Intervalles d'Allen
6. Automates temporisés
7. Réseaux de Petri



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** (…)

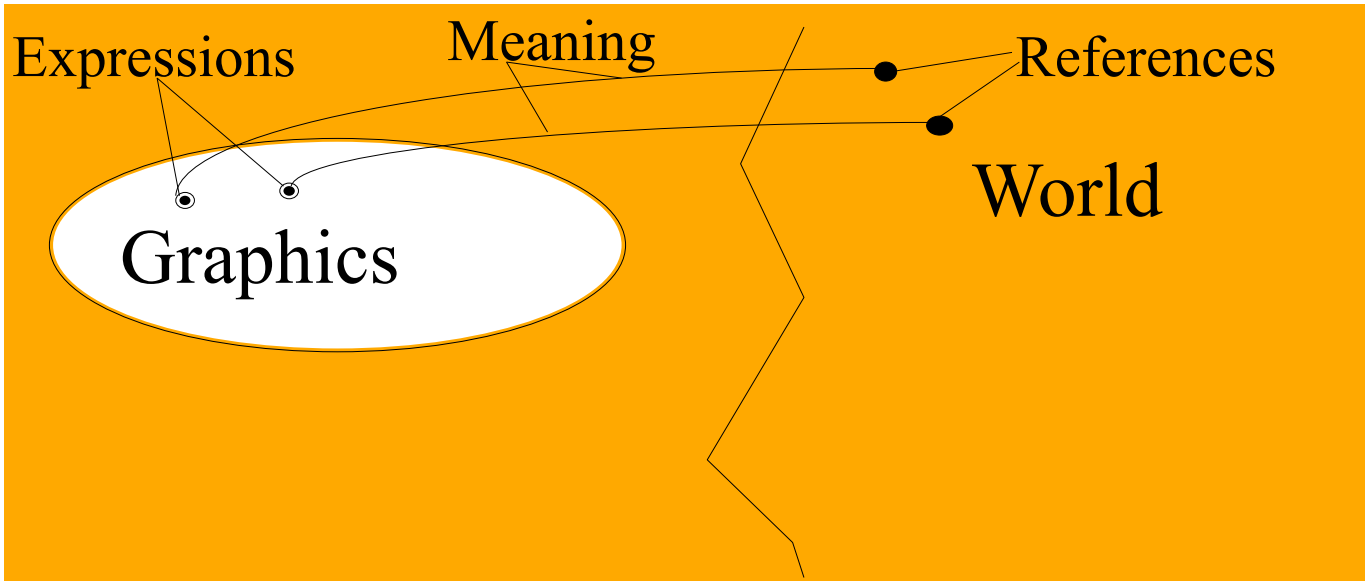
L I P 6 C N R S

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

Sign

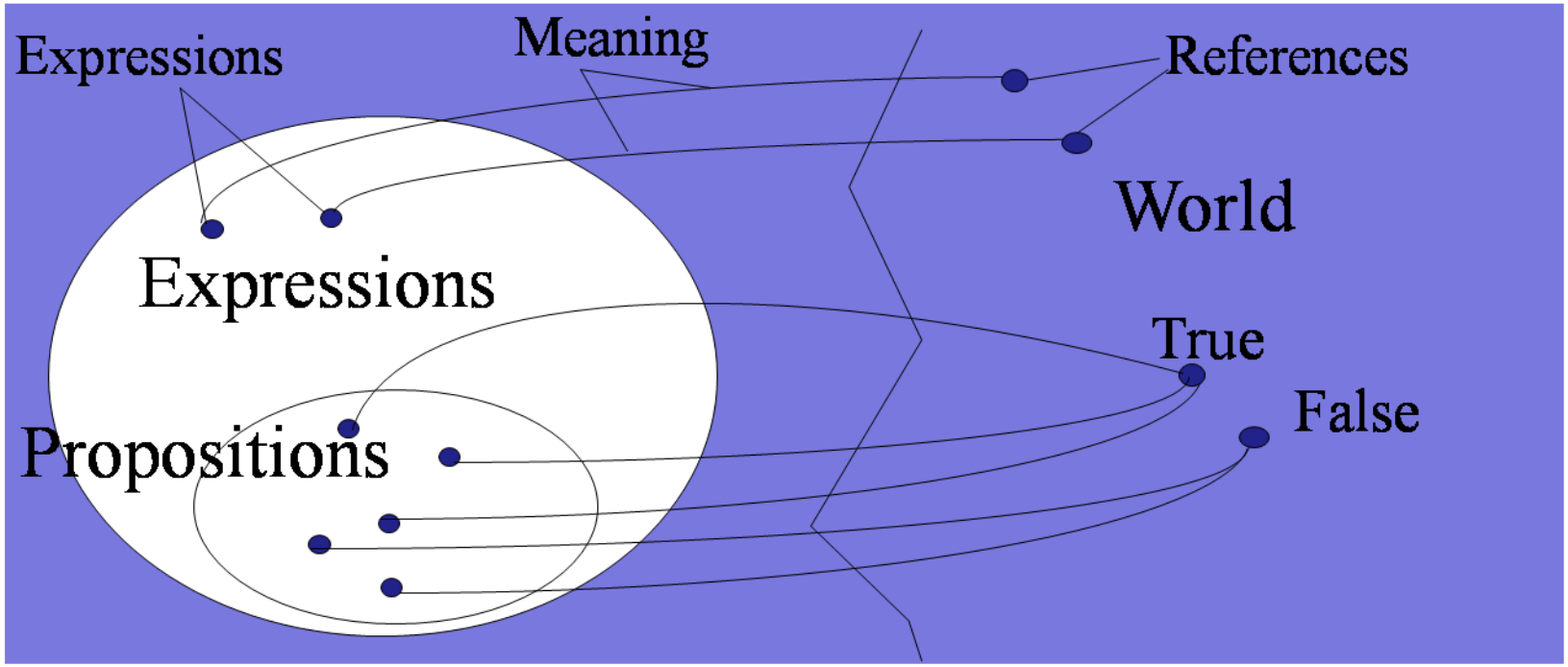
- Expression, Meaning, References



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

Proposition

- Meaning = truth value∈ {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: « ∧ », « ∨ », « → »
  - Unary: « ¬ »
- **Composed Propositions**
  - If A and B are propositions, « ¬ A», « A ∧ B », « A ∨ B », « A → 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

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**

Validity

- A formula A is said to be *valid* if it is true on all the lines of the truth table, i.e. in all the interpretations
  - Example: **(A⊃(B⊃A))**

A	B	(B ⊃A)	**(A ⊃(B ⊃A))**
v	v	v	**v**
v	f	v	**v**
f	v	f	**v**
f	f	v	**v**

Validity: an other example

- Another example:
  (A→(B→C))→((A→B) →(A→C))

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

Satisfiability

- A formula A is said to be *satisfiable* if it is **true** on at least one line of its truth table, i.e. **in at least on interpretation**
  - Example: **(A→ (B→¬A))**

A	B	(B →¬A)	(A → (B →¬A))
v	v	f	f
v	f	v	v
f	v	v	v
f	f	v	v

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*

Logical consequence
Semantic consequence
Entailment

- A is a **semantic consequence** of B if A is true on all the lines of the truth table where B is true
  - Example: (A→B) is a semantic consequence of B.

Notation

⊨ **A** means A is a valid formula

B ⊨ **A** means A is a semantic consequence of B

Deduction Theorem

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

A **Formal System** for the Propositional Logic

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

    SA1: (A → (B → A))

    SA2: ((A → (B → C)) → ((A → B) → (A → C)))

    SA3: ((¬A → ¬B) → (B → A))
  - **Inference rule**: « *Modus Ponens* » **A, (A → B) ⊢ B**

Theorems

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

Notation: $\vdash$ **A** means A is a theorem

Example: $\vdash$ **(A → A)**

***Proof*:**

1- (A → ((**A → A**) → A)) SA1

2- ((A → ((**A → A**) → A)) → ((A → (**A → A**)) → (A → A))) SA2

3- ((A → (**A → A**)) → (A → A)) Modus Ponens 2, 1

4- (A → (**A** → A)) SA1

5- **(A → A)** Modus Ponens 4, 3

Demonstration

Definition: a *proof* of a theorem A is a finite sequence of formulae $F_0, F_1, \ldots F_n$ 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: $A_1, \ldots A_{n-1} \vdash (A_n \rightarrow B)$ ssi $A_1, \ldots, A_{n-1}, A_n \vdash B$

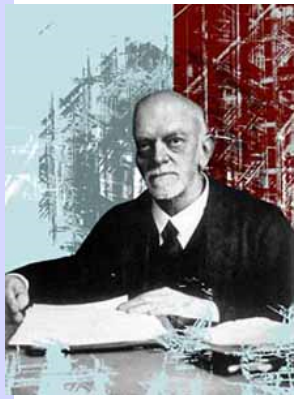
***Proof (direction1)*:**

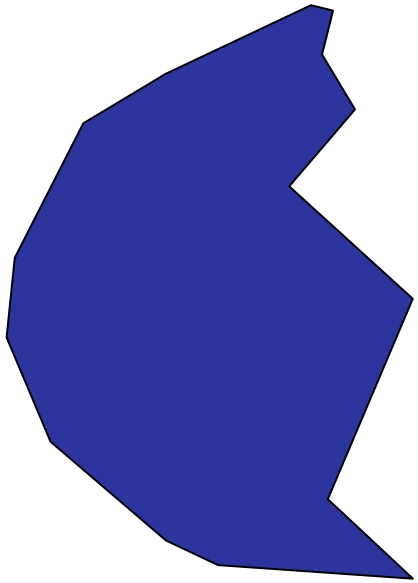
1. $(A_n \rightarrow B)$ Hypothesis 1
2. A_n Hypothesis 2
3. B *Modus Ponens* 1, 2

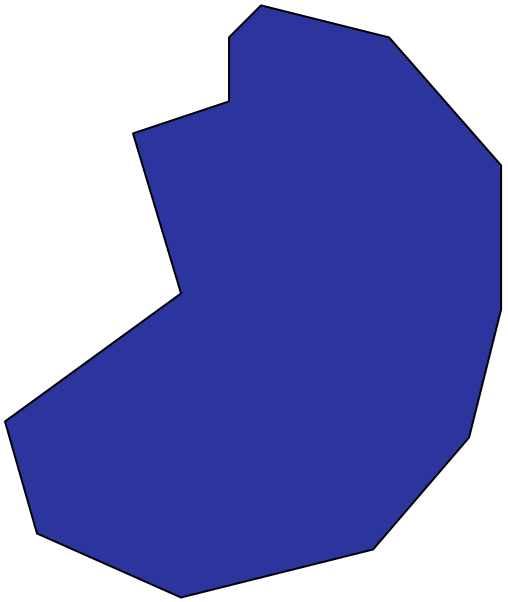
***Proof (direction 2)*:** four possibilities has to be investigated

1. B is an axiom
2. B is one of the hypotheses $A_1, \ldots A_{n-1}$
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*)

Notion of symbolic system





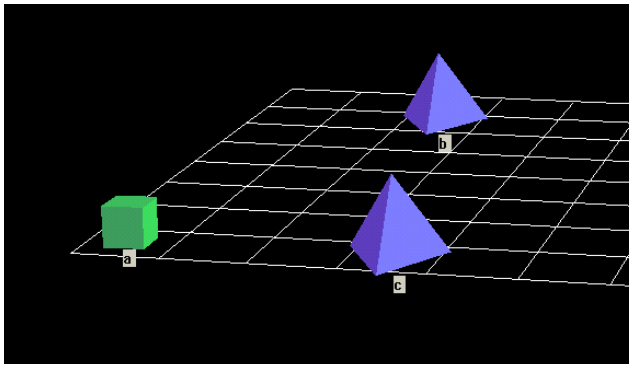


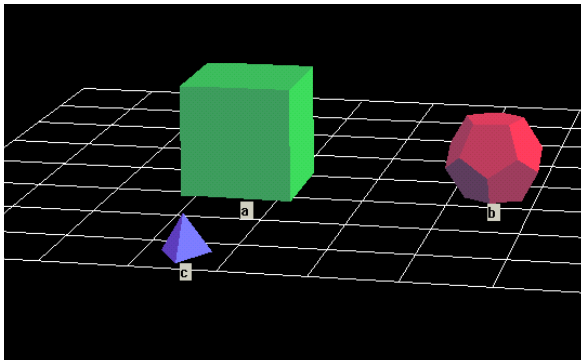
Symbolic system **Mathematical objects**

Consistency: each description of the symbolic system corresponds to an object in the reality, i.e. **∀ A if ⊢ A then ⊨ A**

Completeness: each object of the reality can be described in the symbolic system **∀ A if ⊨ A then ⊢ A**

Propositional Logic





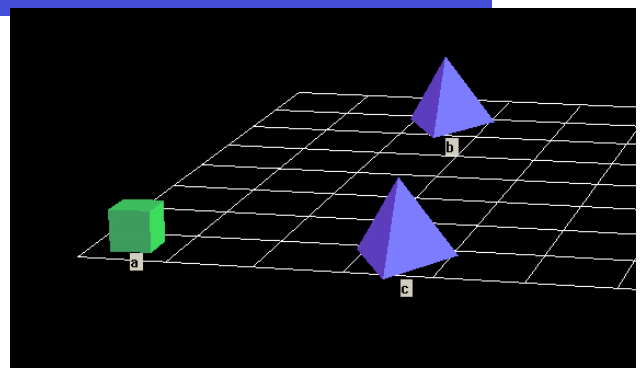
How to translate those sentences in propositional logic:

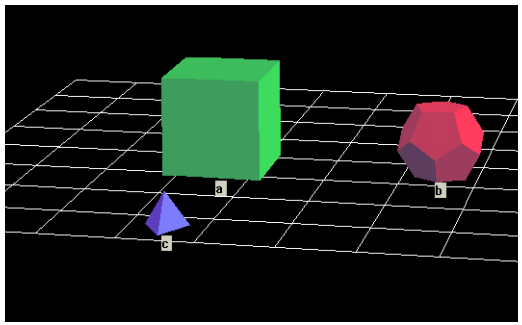
1. There is an object right to a green cube
2. Every successor of an even number is odd
3. 72 is an even number
4. The successor of 72 is odd

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, …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 **$\forall x F$** et **$\exists x F$** are also formulae
    - **$\forall x F$** means "for all x, F"
    - **$\exists x F$** means "there exists x, F"
- **Formulae**:
  - Atoms are formulae
  - If F and G are formulae, «**$\neg F$**», «**$F \wedge G$**», «**$F \vee G$**», «**$F \rightarrow G$**» are formulae
  - If $x \in \mathcal{V}$, **$\forall x F$** and **$\exists x F$** 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 $i[p_n]$

1st 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 all the interpretations of all domains
- A formula is "**satisfiable**" if it is true for at least one interpretation of one domain.
- A formula is "**unsatisfiable**" if it is never true in any domain, i.e. if it is false in all interpretations of all domains.

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" 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:
  - $I_{mv}(P(t_1, t_2, \ldots t_n)) = \text{t}$ if and only if $i(P(t_1, t_2, \ldots t_n)) = \text{t}$
  - $I_{mv}(\neg F) = \neg i(F)$
  - $I_{mv}(F \wedge G) = I_{mv}(F) \,.\, I_{mv}(G)$
  - $I_{mv}(F \vee G) = I_{mv}(F) + I_{mv}(G)$
  - $I_{mv}(F \rightarrow G) = I_{mv}(\neg F \vee G)$

Example 1

$$F = \exists y P(x, y)$$

Domain: N

Meaning of the predicate: i(P(n, m)) = t iff n≤m

Valuation:

- *v(x) = 5 and v(y) = 3*

➢ *We are seeking if for a valuation v' such that*

- *v'(x) = 5*
- *v'(y) = p ∈N*
- *5 ≤p*

For instance, p = 10

$I_{mv'}(P(x, y))$ = t therefore $I_{mv'}(\exists y P(x, y))$ = t

Example 2

$$F = \forall x \exists y P(x, y)$$

Domain: N

Meaning of the predicate: i(P(n, m)) = t iff n≤m

Valuation:

- *v(x) = 5 and v(y) = 3*

➢ *We are seeking if for all valuation v_1 such that*

- *$v_1(x) = p \in N$ and $v_1(y) = 3$*

$I_{mv1}(\exists y P(x, y)) = \text{t}$

➢ *For all valuation v_1 we are seeking for the existence of a valuation v_2 such that $v_1(x) = v_2(x) = p \in N$ and $v_2(y) = q \in N$, $I_{mv2}(P(x, y)) = \text{t}$*

1st order logic semantics

- 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 $\mathcal{M}$

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))

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)

L I P 6 C N R S

Automatic Theorem Proving

Resolution in Propositional Logic
Tableau Method

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 $\neg A$ 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 ⊃ B/B) and the *Modus Tolens* (¬B, A ⊃ B/¬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 $S \cup \{\neg C\}$

Example

- Consider S = {C_1, C_2, C_3}

C_1 = ¬man ∨ mortal = man ⊃ mortal

C_2= ¬socrate ∨ man = socrate ⊃ man

C_3= socrate, C_4= ¬mortal,

- To prove S ⊨ mortel, it is sufficient to derive the empty clause, i.e. □, from S ∪{C_4}

C_5: ¬man res(C_1, C_4; mortal, ¬mortal)

C_6: ¬socrate res(C_2, C_5; man, ¬man)

C_5: □ res(C_3, C_6; socrate, ¬socrate)

QED

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

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

$$R_\vee \frac{T}{T \cup \{\varphi\} \mid T \cup \{\psi\}} \quad \text{If } \varphi \vee \psi \in T \text{ and } \varphi \notin T \text{ and } \psi \notin T$$

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.

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

$$R_{\vee}\ \frac{T}{T\cup\{\varphi\} \mid T\cup\{\psi\}}\ \boxed{\text{If } \varphi\vee\psi \in T \text{ and } \varphi \notin T \text{ and } \psi \notin T}$$

Properties

1. There is finite sequence of applications of rules
   $\mathcal{S}_0 = \mathsf{T} \rightarrow \mathcal{S}_1 \rightarrow \mathcal{S}_2 \rightarrow \ldots$
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

Third step

- T being a complete tableau

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

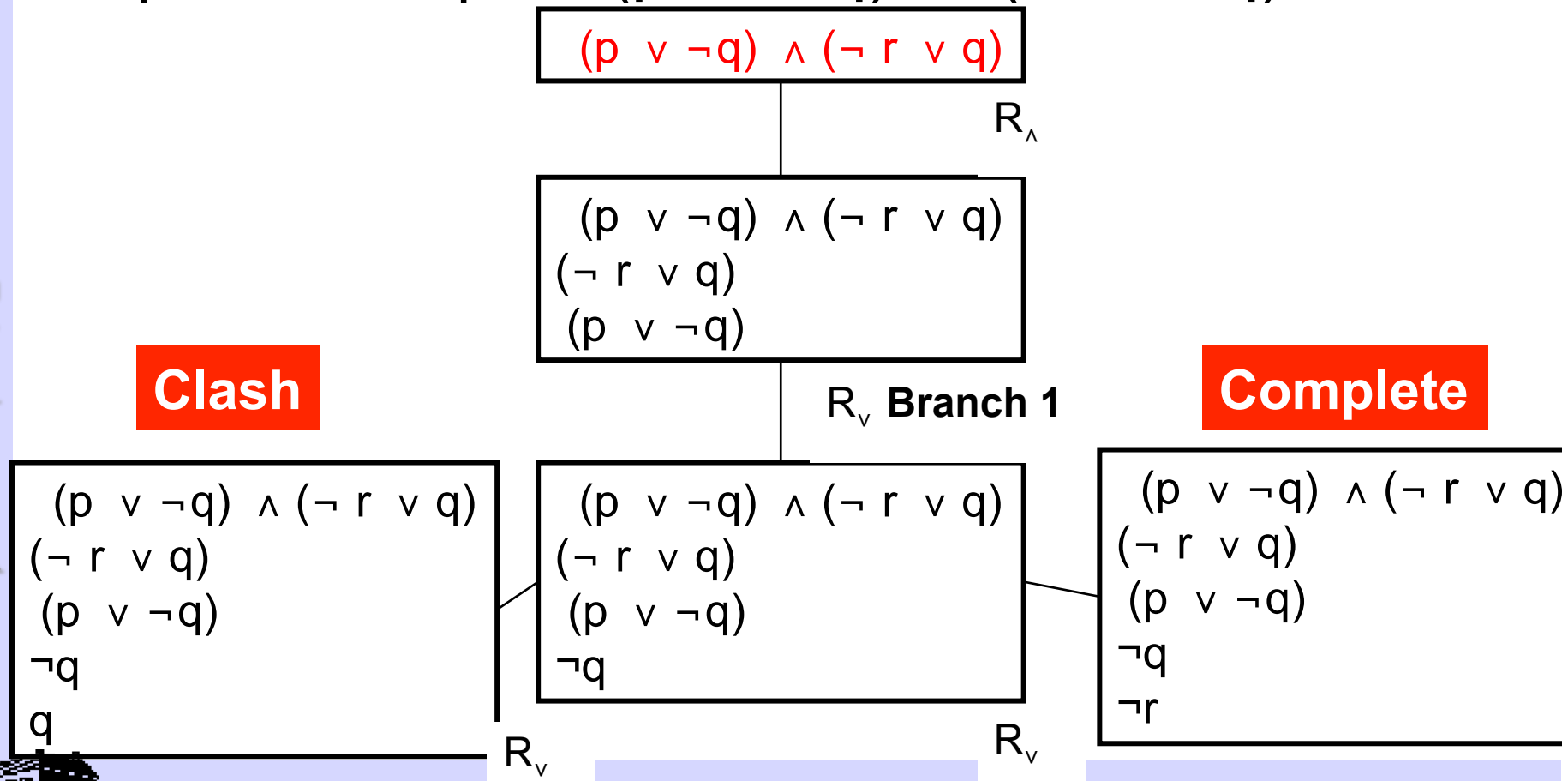
If p, a proposition, belongs to T, then p is true in M[T],

else p is false

Example

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

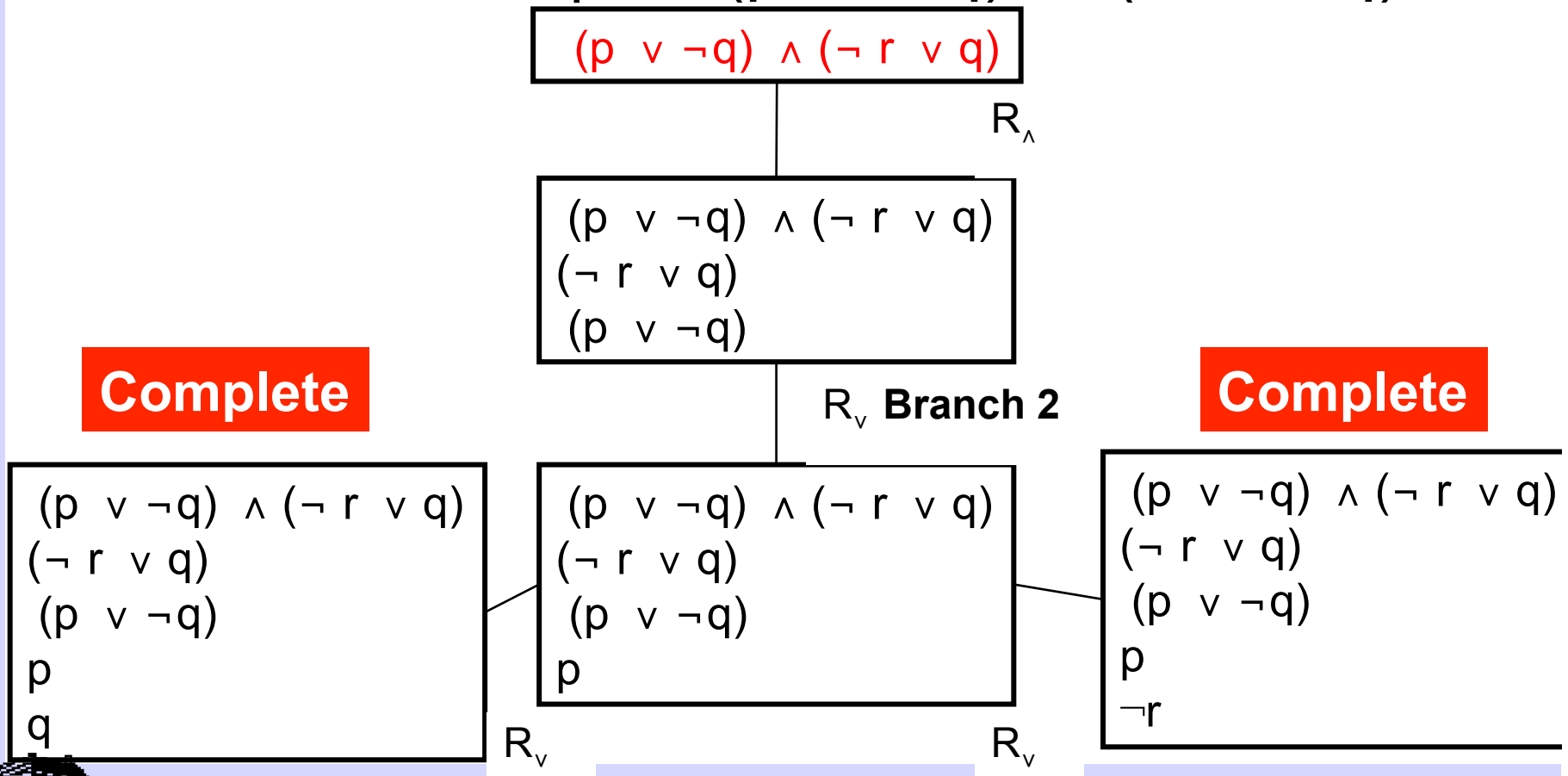
φ rewrite: $\varphi = (p \vee \neg q) \wedge (\neg r \vee q)$



Example - follows

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

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



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*:

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

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

Tableau Method

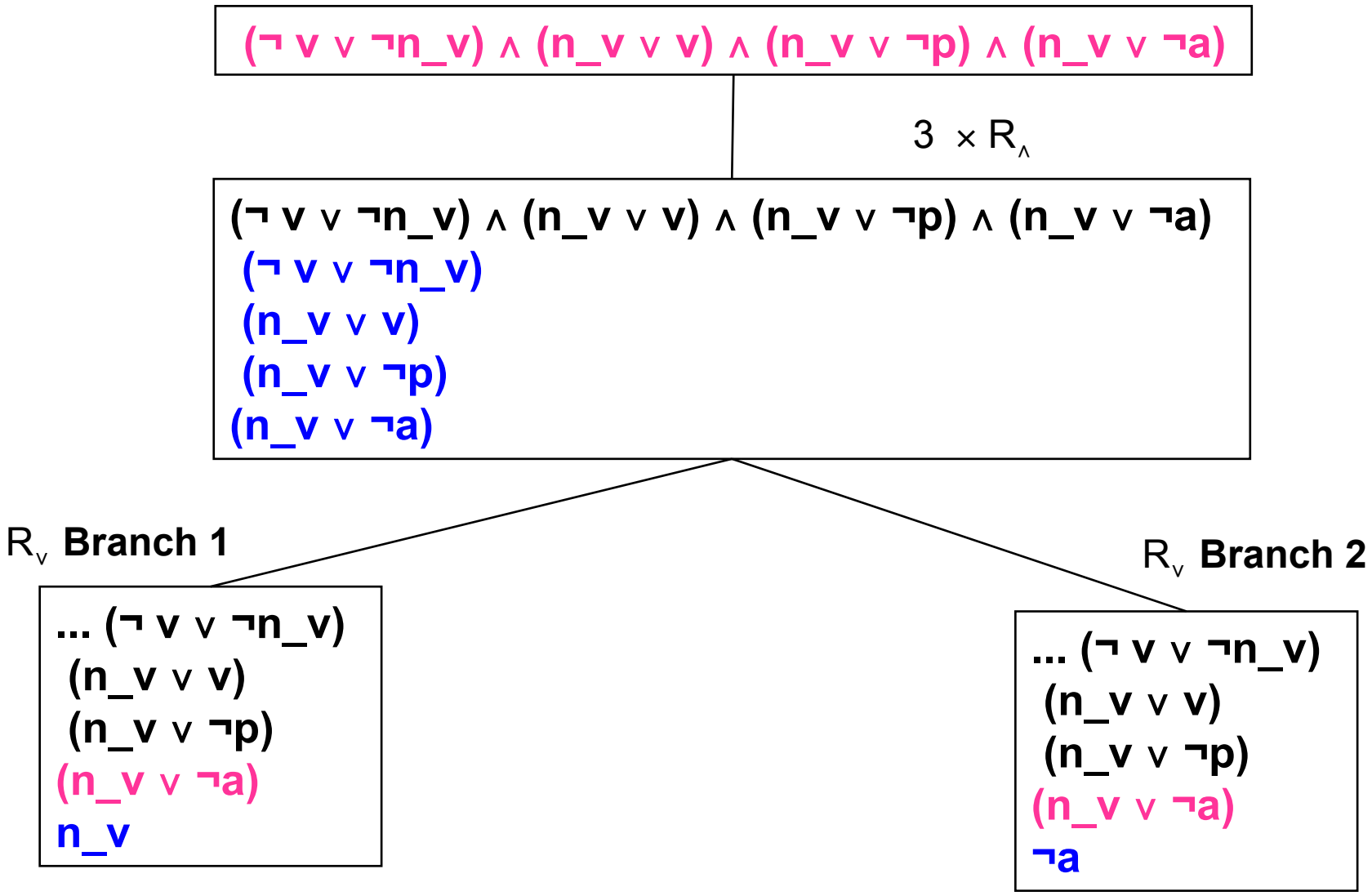


Tableau Method – *left part*

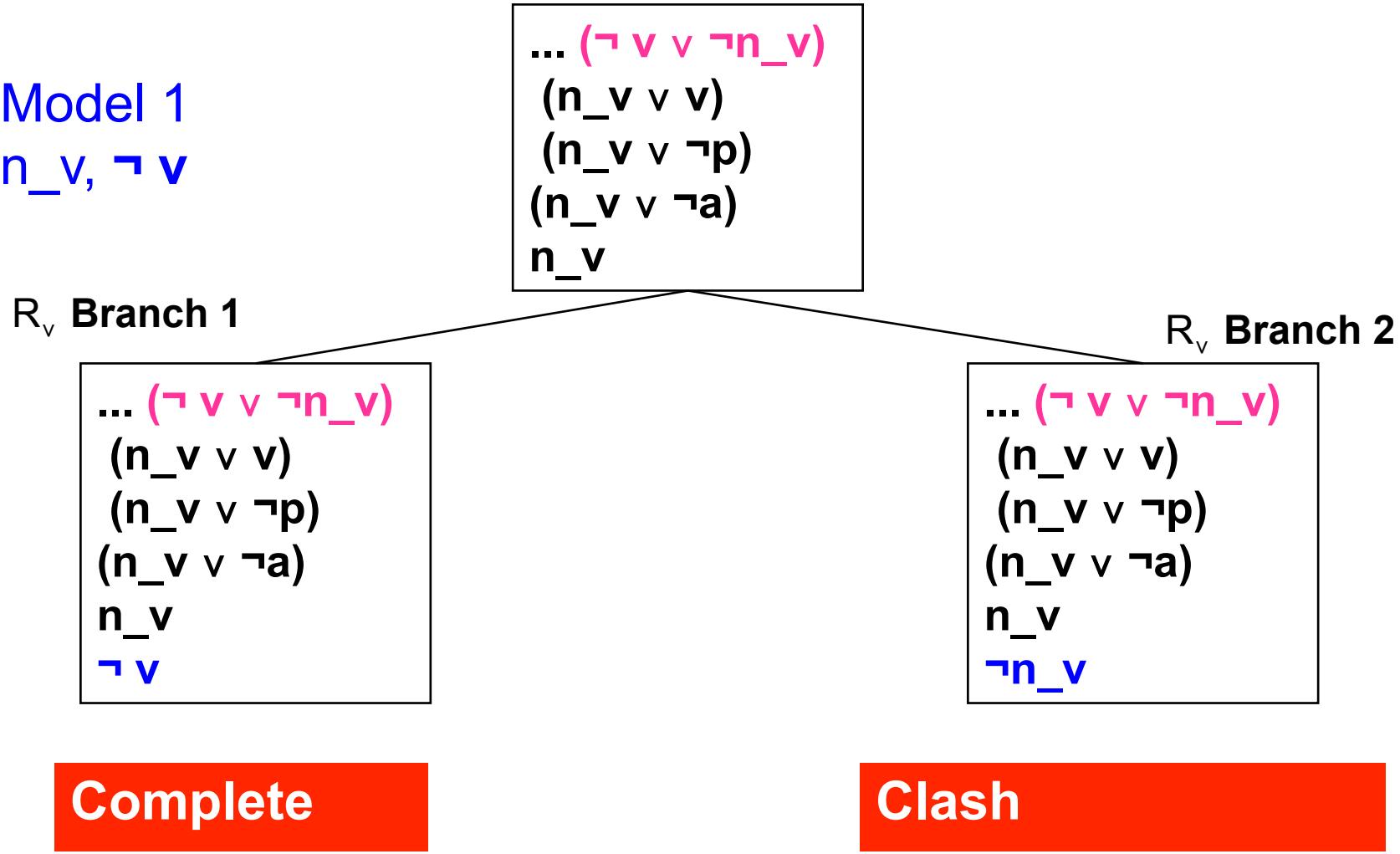
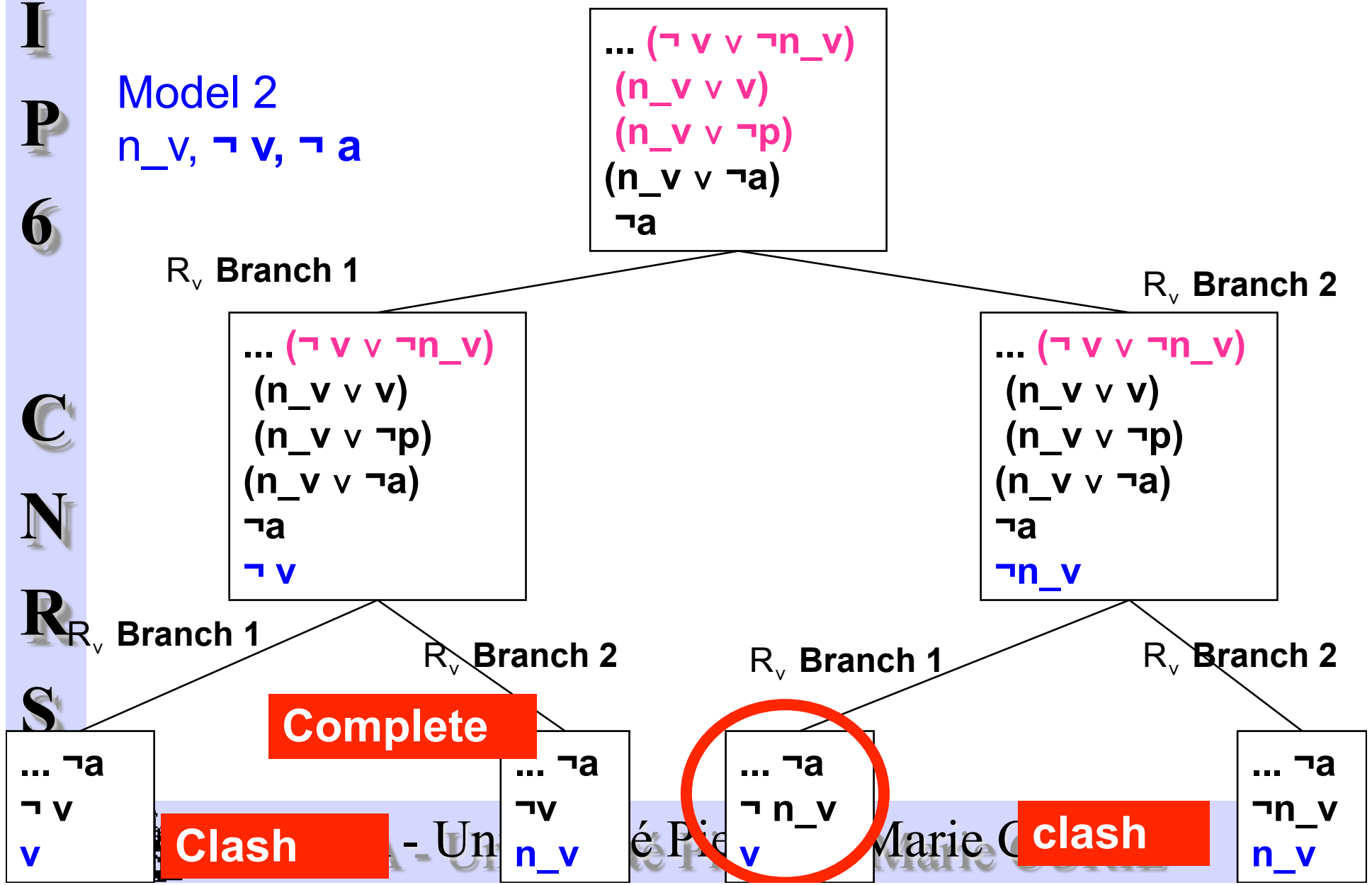
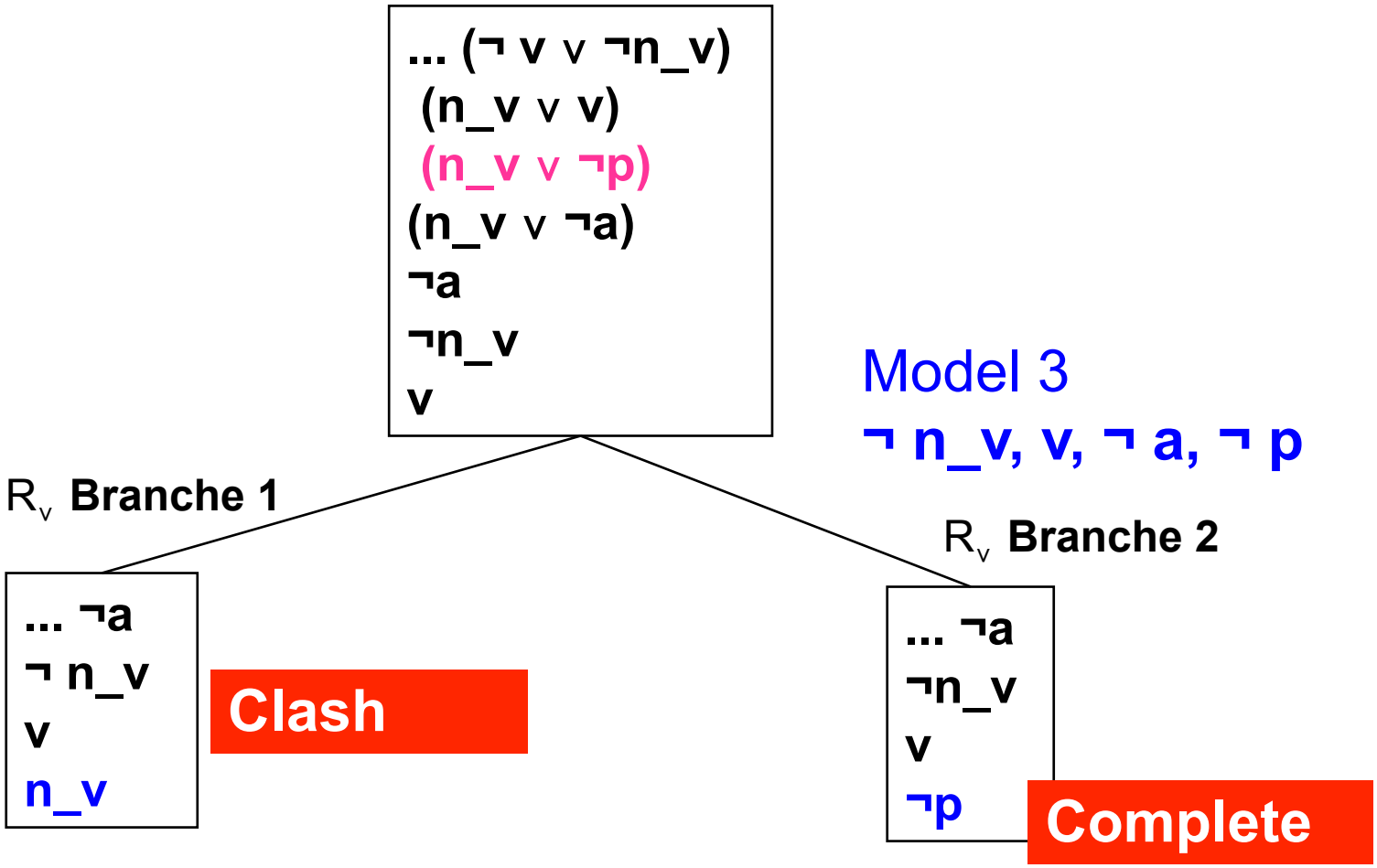


Tableau Method – *right part*



Following – *right*



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, ¬ v
Model 2: n_v, ¬ v, ¬ a
Model 3: ¬ n_v, v, ¬ a, ¬ p