



Université
de Toulouse

LFlex - Structures discrètes 1

Département d'informatique de la FSI
(2024 – 2025 sem. 1)



Plan

1 Introduction

■ Problématique

- Histoire de la logique
- Focus sur la logique des propositions
- Quelques personnages emblématiques
- Applications en informatique
- Preuve par induction

2 Logique des propositions

3 Logique des prédictats

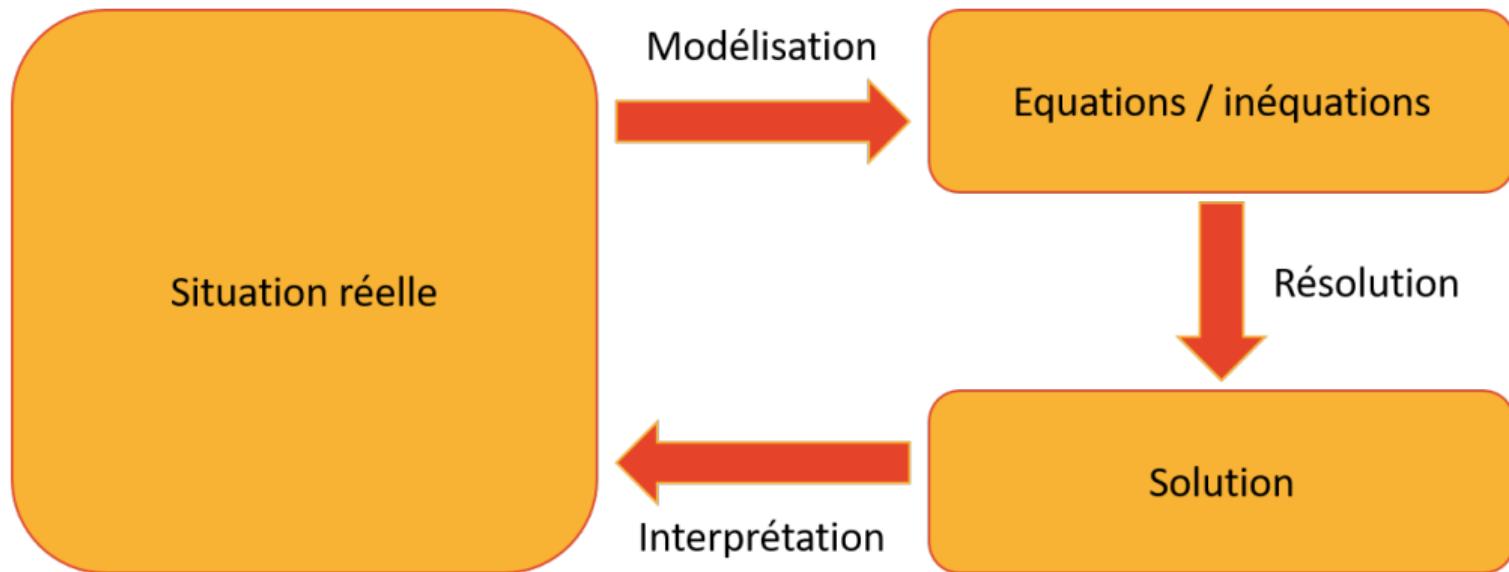


Introduction

Problématique



Modélisation mathématique (principe)





Modélisation mathématique (exemple)

- **Énoncé à modéliser :** en plongée, le taux de variation de la tension d'un gaz neutre dans un compartiment varie linéairement avec le gradient de tension de ce gaz par rapport à sa tension finale dans ce compartiment ;

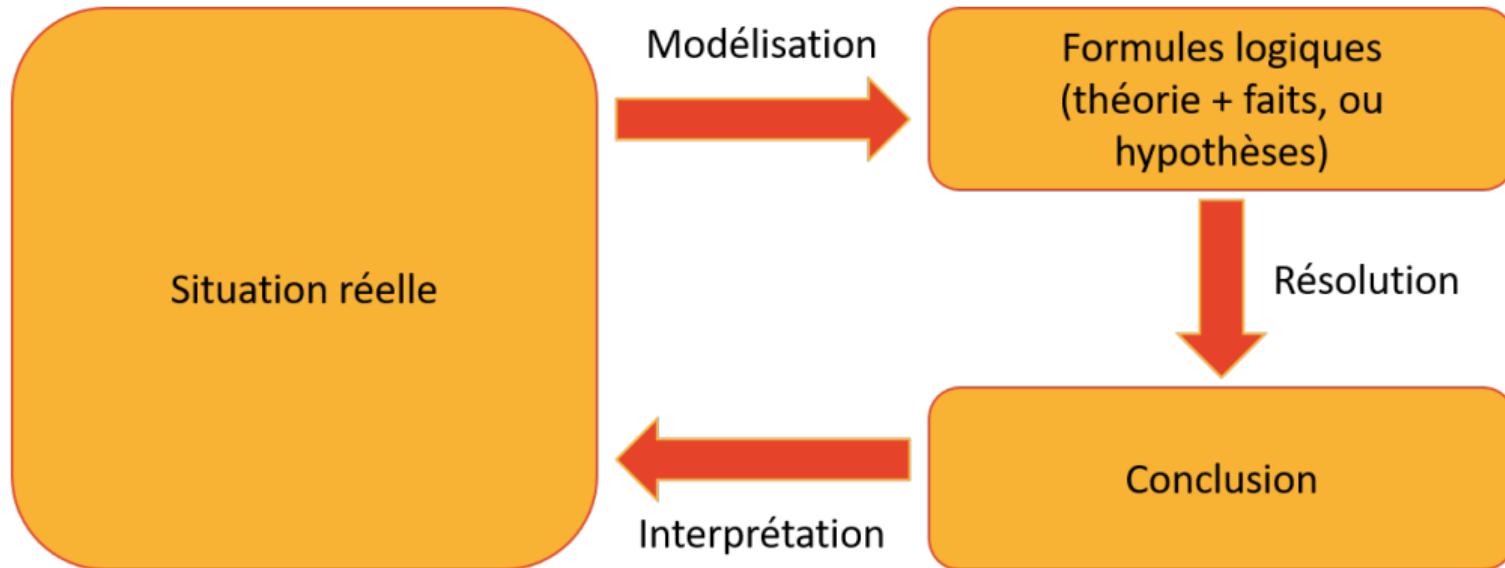


Modélisation mathématique (exemple)

- **Énoncé à modéliser :** en plongée, le taux de variation de la tension d'un gaz neutre dans un compartiment varie linéairement avec le gradient de tension de ce gaz par rapport à sa tension finale dans ce compartiment ;
- **Modélisation mathématique :** $\frac{dT_{gn}(t)}{dt} = -k(T_{gn}(t) - T_f).$



Modélisation logique (principe)





Modélisation logique (exemple)

■ Énoncé à modéliser :

- Une prise en charge médicale (PCM) est invasive si et seulement si elle nécessite une injection ou une sonde respiratoire ;
- Si une PCM ne nécessite pas d'injection alors si l'électrocardiogramme (ECG) n'est pas normal alors elle nécessite une sonde respiratoire ;
- S'il y a présence d'ondes P alors l'ECG n'est pas normal ;
- Conclusion ?



Modélisation logique (exemple)

■ Modélisation logique de l'exemple de la PCM :

- L'ensemble d'hypothèses $\{i \leftrightarrow q \vee s, \neg q \rightarrow \neg n \rightarrow s, p \rightarrow \neg n\}$
- entraîne-t-il la conclusion : $\neg i \rightarrow \neg p$



Modélisation logique (exemple)

- Modélisation logique de l'exemple de la PCM :
 - L'ensemble d'hypothèses $\{i \leftrightarrow q \vee s, \neg q \rightarrow \neg n \rightarrow s, p \rightarrow \neg n\}$
 - entraîne-t-il la conclusion : $\neg i \rightarrow \neg p$
- Vous apprendrez à :
 - Modéliser (formaliser)
 - Résoudre
 - Interpréter



Introduction

Histoire de la logique



Vocabulaire : énoncé vs proposition

- Un **énoncé** est une phrase pouvant être vraie ou fausse :
 - « Vous aurez Logique lundi 30/01 ou mardi 31/01 » ;
 - « La vitesse de la lumière est constante » ;
 - « La fonction $f(x) = 2 \cdot \cos(x)$ est continue et dérivable sur \mathbb{R} » ;



Vocabulaire : énoncé vs proposition

- Un **énoncé** est une phrase pouvant être vraie ou fausse :
 - « Vous aurez Logique lundi 30/01 ou mardi 31/01 » ;
 - « La vitesse de la lumière est constante » ;
 - « La fonction $f(x) = 2 \cdot \cos(x)$ est continue et dérivable sur \mathbb{R} » ;
- Une **proposition** est un énoncé atomique/élémentaire (*i.e.* ne contenant pas de sous-proposition) :
 - « Vous aurez Logique lundi 30/01 » ;
 - « La vitesse de la lumière est constante » ;
 - « $2^{137107291} - 1$ est un nombre premier ».



Vocabulaire : théorie, règle de raisonnement et jugement

- Une **théorie** est un ensemble d'énoncés ;



Vocabulaire : théorie, règle de raisonnement et jugement

- Une **théorie** est un ensemble d'énoncés ;
- une **règle de raisonnement** est l'affirmation d'un lien de conséquence entre une théorie (les hypothèses) et un énoncé (la **conclusion**) ;



Vocabulaire : théorie, règle de raisonnement et jugement

- Une **théorie** est un ensemble d'énoncés ;
- une **règle de raisonnement** est l'affirmation d'un lien de conséquence entre une théorie (les hypothèses) et un énoncé (la **conclusion**) ;
- un **jugement** est l'application d'une règle de raisonnement sur une théorie et une conclusion particulières.



Théorie, règle de raisonnement et jugement (exemple)

■ Règle de raisonnement :

- il est vrai que x ou y , mais comme x est faux, c'est donc que y est vrai;

■ théorie :

- il vient lorsque je suis malade ou lorsqu'il veut me demander quelque chose
- or je ne suis pas malade

■ jugement (utilisation de la règle de raisonnement ci-dessus)

- donc

■ conclusion :

- il vient me demander quelque chose.



Validité d'une règle de raisonnement

- Une règle de raisonnement est valide *ssi* une théorie vraie conduit nécessairement à une conclusion vraie.



Validité d'une règle de raisonnement

- Une règle de raisonnement est valide *ssi* une théorie vraie conduit nécessairement à une conclusion vraie.
- corollaire : une règle de raisonnement qui, à partir d'une théorie vraie, permet d'obtenir une conclusion fausse est *invalidé ou non valide*.



Validité d'une règle de raisonnement

- Une règle de raisonnement est valide *ssi* une théorie vraie conduit nécessairement à une conclusion vraie.
- corollaire : une règle de raisonnement qui, à partir d'une théorie vraie, permet d'obtenir une conclusion fausse est **invalidé ou non valide**.
- exemples :
 - « il est vrai que x ou y , mais comme x est faux, c'est donc que y est vrai » est **valide**



Validité d'une règle de raisonnement

- Une règle de raisonnement est valide *ssi* une théorie vraie conduit nécessairement à une conclusion vraie.
- corollaire : une règle de raisonnement qui, à partir d'une théorie vraie, permet d'obtenir une conclusion fausse est **invalidé ou non valide**.
- exemples :
 - « il est vrai que x ou y , mais comme x est faux, c'est donc que y est vrai » est **valide**
 - mais « il est vrai que x ou y , mais comme x est vrai, c'est donc que y est faux » est **non valide**!
(si x et y sont tous les deux vrais, la théorie est vraie mais pas la conclusion)



Validité d'un jugement

- Une judgement est valide *ssi* il utilise une règle de raisonnement valide ET se base sur une théorie est vraie.



Validité d'un jugement

- Une jugement est valide *ssi* il utilise une règle de raisonnement valide ET se base sur une théorie est vraie.
- **Corollaire** : un jugement dont la règle de raisonnement n'est pas valide OU dont la théorie sur laquelle il se base est fausse, est un jugement non valide appelé **jugement fallacieux**.



Validité d'un jugement

- Une jugement est valide *ssi* il utilise une règle de raisonnement valide ET se base sur une théorie est vraie.
- **Corollaire** : un jugement dont la règle de raisonnement n'est pas valide OU dont la théorie sur laquelle il se base est fausse, est un jugement non valide appelé *jugement fallacieux*.
- **Remarque** : un jugement dont la règle de raisonnement est valide et dont la théorie est vraie *ne peut pas* conduire à une conclusion fausse (cela contredit le fait que la règle de raisonnement est valide ou que la théorie est vraie).



Jugement fallacieux (paralogisme vs. sophisme)

- Pour la petite histoire, c'est un jugement :
 - ayant les apparences de la vérité,
 - élaboré :
 - soit de bonne foi (erreur involontaire) ➔ **paralogisme**,
 - soit pour tromper délibérément ➔ **sophisme**.



Jugement fallacieux (paralogisme vs. sophisme)

- Pour la petite histoire, c'est un jugement :
 - ayant les apparences de la vérité,
 - élaboré :
 - soit de bonne foi (erreur involontaire) ➔ **paralogisme**,
 - soit pour tromper délibérément ➔ **sophisme**.
- Note :
 - **Sophistes** : penseurs grecs à l'origine de la rhétorique ;
 - **rhétorique** : art de la persuasion, potentiellement aux dépends de la vérité (la publicité en est parfois une forme moderne).



Jugement fallacieux (exemple 1)

- Soit le **jugement** suivant :
 - les oiseaux pondent des oeufs,
 - les chats sont des oiseaux,
 - **donc** les chats pondent des œufs.
- nature du **problème** ?



Jugement fallacieux (exemple 1)

- Soit le **jugement** suivant :
 - les oiseaux pondent des oeufs,
 - les chats sont des oiseaux,
 - **donc** les chats pondent des œufs.
- nature du **problème**?
 - conclusion fausse ou absurde ➔ cela peut venir d'une règle de raisonnement non valide ou d'une théorie fausse



Jugement fallacieux (exemple 1)

- Soit le **jugement** suivant :
 - les oiseaux pondent des oeufs,
 - les chats sont des oiseaux,
 - **donc** les chats pondent des œufs.
- nature du **problème**?
 - conclusion fausse ou absurde ➔ cela peut venir d'une règle de raisonnement non valide ou d'une théorie fausse
 - la règle de raisonnement est valide



Jugement fallacieux (exemple 1)

- Soit le **jugement** suivant :
 - les oiseaux pondent des oeufs,
 - les chats sont des oiseaux,
 - donc les chats pondent des œufs.
- nature du **problème**?
 - conclusion fausse ou absurde ➡ cela peut venir d'une règle de raisonnement non valide ou d'une théorie fausse
 - la règle de raisonnement est valide
 - par contre la théorie contient un énoncé qui n'est pas vrai (les chats ne sont pas des oiseaux)



Jugement fallacieux (exemple 1)

- Soit le **jugement** suivant :
 - les oiseaux pondent des oeufs,
 - les chats sont des oiseaux,
 - donc les chats pondent des œufs.
- nature du **problème**?
 - conclusion fausse ou absurde ➡ cela peut venir d'une règle de raisonnement non valide ou d'une théorie fausse
 - la règle de raisonnement est valide
 - par contre la théorie contient un énoncé qui n'est pas vrai (les chats ne sont pas des oiseaux)
- **sophisme** ou **paralogisme** : ???



Jugement fallacieux (exemple 2)

- Soit le **jugement** suivant :
 - les chats sont des mammifères,
 - les chiens sont des mammifères,
 - les chiens ont des poils,
 - **donc** les chats ont des poils.
- **nature du problème ?**



Jugement fallacieux (exemple 2)

- Soit le **jugement** suivant :
 - les chats sont des mammifères,
 - les chiens sont des mammifères,
 - les chiens ont des poils,
 - **donc** les chats ont des poils.
- **nature du problème** ?
 - théorie et conclusion sont vraies ➔ la règle de raisonnement serait-elle non valide ?
 - oui ! (Par exemple, remplacer « chats » par « dauphins » : la théorie reste vraie, pourtant la conclusion est fausse !)



Le syllogisme d'Aristote

- Type de jugement qui comporte trois propositions :
 - 2 pour la théorie, appelées prémisses,
 - 1 pour la conséquence du jugement, appelée conclusion ;

1. tous, aucun, certains, quelque(s),...



Le syllogisme d'Aristote

- Type de jugement qui comporte trois propositions :
 - 2 pour la théorie, appelées prémisses,
 - 1 pour la conséquence du jugement, appelée conclusion ;
- chaque proposition :
 - a une forme simple (propriété caractérisant un seul élément),
 - a la forme : quantificateur¹ + sujet + verbe + complément ;

1. tous, aucun, certains, quelque(s),...



Le syllogisme d'Aristote

- Type de jugement qui comporte trois propositions :
 - 2 pour la théorie, appelées prémisses,
 - 1 pour la conséquence du jugement, appelée conclusion ;
- chaque proposition :
 - a une forme simple (propriété caractérisant un seul élément),
 - a la forme : quantificateur¹ + sujet + verbe + complément ;
- plus généralement, un syllogisme peut avoir plus de deux prémisses.

1. tous, aucun, certains, quelque(s),...



Le syllogisme d'Aristote (exemple)

- Prémisses :
 - les mammifères sont des animaux ,
 - les chats sont des mammifères,
- conclusion :
 - Donc les chats sont des animaux.



Le syllogisme d'Aristote (exemple)

■ Prémisses :

- les mammifères sont des animaux ,
- les chats sont des mammifères,

■ conclusion :

- Donc les chats sont des animaux.

■ Remarque :

- prémisses et conclusion sont toujours vraies ;
- le lien de conséquence prémisses/conclusion est correct ;
- ➡ ce jugement est **valide**



Syllogisme fallacieux (exemple)

- Soit le **jugement** suivant :
 - Ce qui est rare est cher;
 - les lingots d'or bon marché sont rares ;
 - donc les lingots d'or bon marché sont chers.
- Quel est le **problème** ?



Syllogisme fallacieux (exemple)

- Soit le **jugement** suivant :
 - Ce qui est rare est cher;
 - les lingots d'or bon marché sont rares ;
 - donc les lingots d'or bon marché sont chers.
- Quel est le **problème** ? **erreur de modélisation** :
 - première prémissse non valide
(le trèfle à 4 feuilles est un contre-exemple)
 - la conclusion est absurde !
(bien que la règle de raisonnement soit valide)



Le syllogisme d'Aristote (structure générale)

- Instance :
 - Prémisse majeure :
Tous les hommes sont mortels
 - Prémisse mineure :
Tous les Grecs sont des hommes
 - Conclusion :
Tous les Grecs sont mortels



Le syllogisme d'Aristote (structure générale)

■ Instance :

■ Prémisse majeure :

Tous les hommes sont mortels

■ Prémisse mineure :

Tous les Grecs sont des hommes

■ Conclusion :

Tous les Grecs sont mortels

■ En général :

■ Prémisse majeure :

Tous les B sont des C

■ Prémisse mineure :

Tous les A sont des B

■ Conclusion :

Tous les A sont des C



La syllogistique

- C'est la science des syllogismes :
 - Aucun M n'est P , or tout S est M , DONC aucun S n'est P ;
 - Tout P est M , or quelque² S n'est pas M , DONC quelque S n'est pas P ;
 - Tout P est M , or tout M est S , DONC quelque S est P ;
 - ...

2. Quelque, au singulier : au moins un



La syllogistique

- C'est la science des syllogismes :
 - Aucun M n'est P , or tout S est M , DONC aucun S n'est P ;
 - Tout P est M , or quelque² S n'est pas M , DONC quelque S n'est pas P ;
 - Tout P est M , or tout M est S , DONC quelque S est P ;
 - ...
- qui en toute généralité, peuvent contenir un nombre quelconque de prémisses.

2. Quelque, au singulier : au moins un



Comment déterminer si un syllogisme est valide ?

- On peut reconnaître un **jugement valide** par sa **forme syntaxique** sans regarder la signification des mots (leur **sémantique**)
 - ➡ la syllogistique recense de tels jugements ;



Comment déterminer si un syllogisme est valide ?

- On peut reconnaître un **jugement valide** par sa **forme syntaxique** sans regarder la signification des mots (leur **sémantique**)
 - ➔ la syllogistique recense de tels jugements ;
- **MAIS** il peut exister des syllogismes non valides
 - ➔ jugements fallacieux ;



Comment déterminer si un syllogisme est valide ?

- On peut reconnaître un **jugement valide** par sa **forme syntaxique** sans regarder la signification des mots (leur **sémantique**)
 - ➔ la syllogistique recense de tels jugements ;
- **MAIS** il peut exister des syllogismes non valides
 - ➔ jugements fallacieux ;
- comment déterminer si un syllogisme est valide ou non ?
 - ➔ voir diagrammes de Venn (slide 26) et classes de jugement (slide 34).



Énoncés universels vs existentiels

- Les **énoncés universels** sont trivialement vrais lorsque leur domaine est vide :
 - « Tous les martiens sont verts » est trivialement vrai puisqu'il n'existe aucun martien !
 - autrement dit : si l'ensemble des martiens est vide, celui-ci est trivialement inclus dans l'ensemble des individus verts ;



Énoncés universels vs existentiels

- Les **énoncés universels** sont trivialement vrais lorsque leur domaine est vide :
 - « Tous les martiens sont verts » est trivialement vrai puisqu'il n'existe aucun martien !
 - autrement dit : si l'ensemble des martiens est vide, celui-ci est trivialement inclus dans l'ensemble des individus verts ;
- Les **énoncés existentiels** supposent l'existence d'un élément dans le domaine :
 - de tels énoncés excluent donc que le domaine soit vide ;
 - « Quelque martien est vert » est faux dès lors qu'on suppose que les martiens n'existent pas.



Lien avec la théorie des ensembles

- Traduction moderne des quatre formes d'énoncé :
 - Tout A est B : $A \subseteq B$;
 - Aucun A n'est B : $A \cap B = \emptyset$;
 - Quelque A est B : $A \cap B \neq \emptyset$;
 - Quelque A n'est pas B : $A \not\subseteq B$;
- les **diagrammes de Venn** s'appuient sur cette représentation ensembliste.



Les diagrammes de Venn (principe)

- Pour savoir si le raisonnement suivant est valide :
 - (P_1) Tout A est B ; (prémissse 1)
 - (P_2) Quelque C est A ; (prémissse 2)
 - (C) Quelque C est B ; (conclusion)



Les diagrammes de Venn (principe)

- Pour savoir si le raisonnement suivant est valide :
 - (P_1) Tout A est B ; (prémissse 1)
 - (P_2) Quelque C est A ; (prémissse 2)
 - (C) Quelque C est B ; (conclusion)
- on représente (P_1) et (P_2) par des ensembles et on obtient une ou plusieurs figures possibles ;



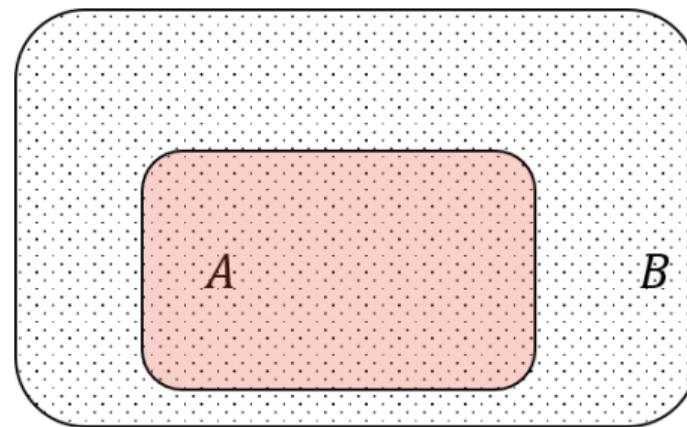
Les diagrammes de Venn (principe)

- Pour savoir si le raisonnement suivant est valide :
 - (P_1) Tout A est B ; (prémissse 1)
 - (P_2) Quelque C est A ; (prémissse 2)
 - (C) Quelque C est B ; (conclusion)
- on représente (P_1) et (P_2) par des ensembles et on obtient une ou plusieurs figures possibles ;
- on teste la représentation de (C) :
 - si elle est **compatible avec toutes** les représentations possibles de prémisses, le jugement est valide ;
 - sinon, le jugement n'est pas valide.



Les diagrammes de Venn (représentation de (P_1))

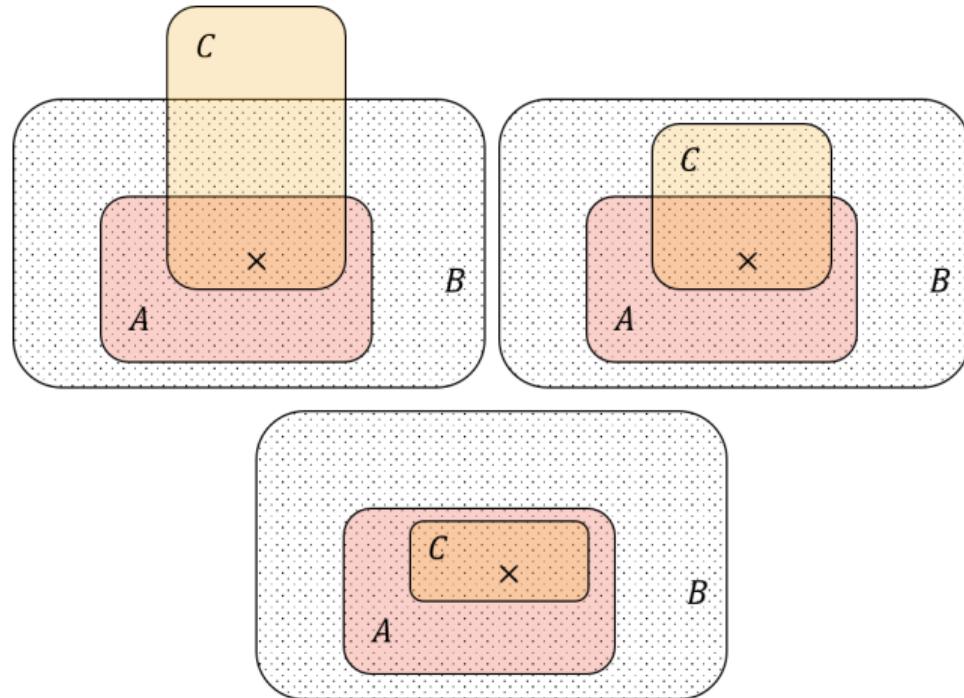
- (P_1) Tout A est B





Les diagrammes de Venn (ajout de (P_2))

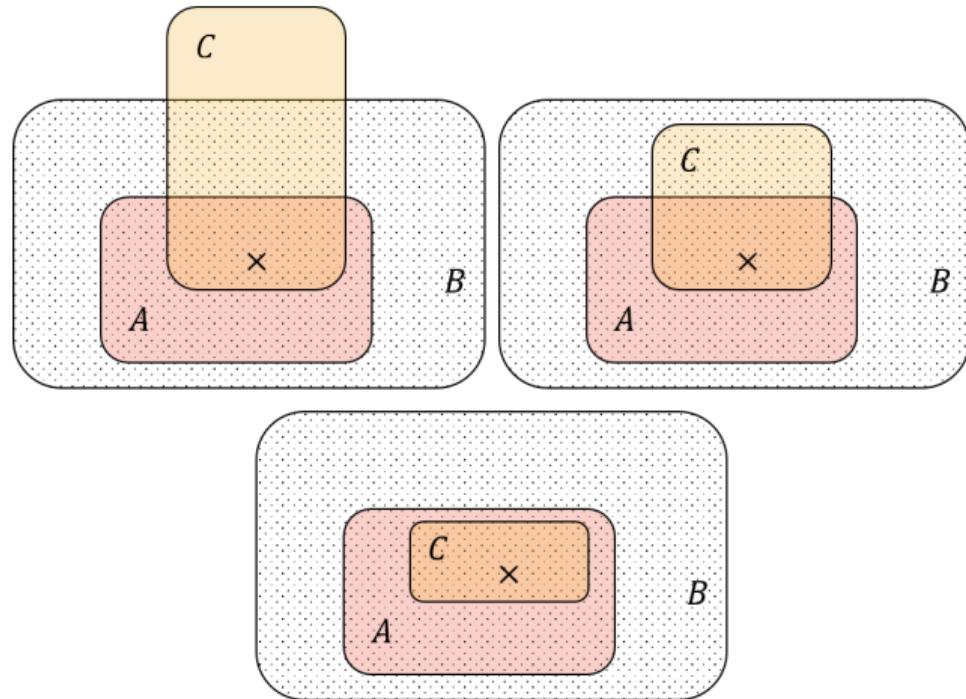
- (P_2) Quelque C est A
 - 3 cas possibles pour que $A \cap C \neq \{\}$;
 - Remarque : pour représenter un ensemble contenant au moins un élément (quantification existentielle) on met un \times dans l'ensemble en question.





Les diagrammes de Venn (vérification compatibilité de (C))

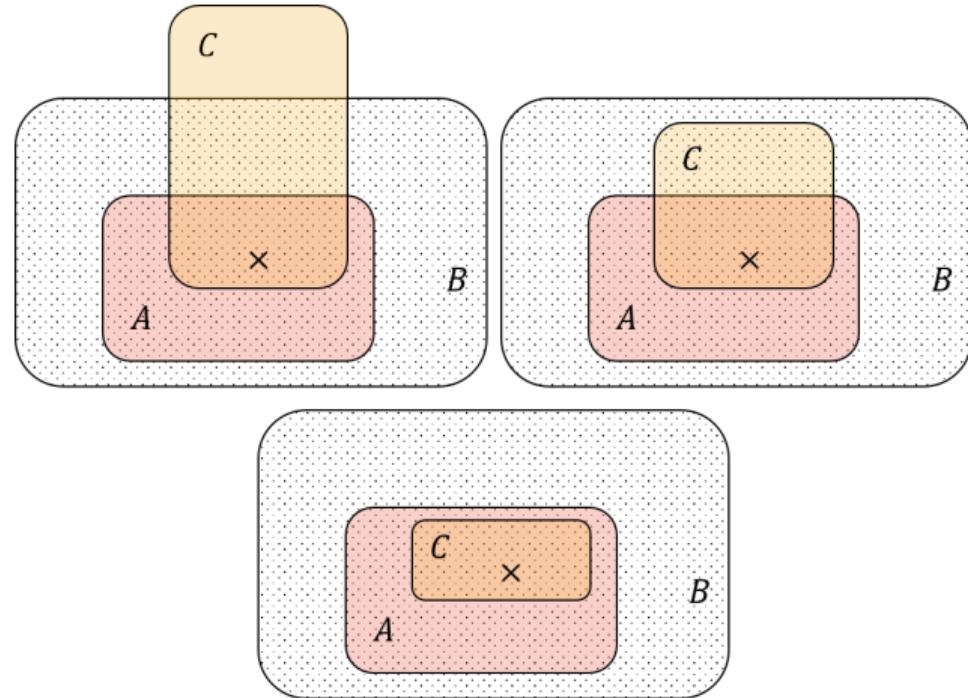
- (C) Quelque C est B
 - La contrainte $C \cap B \neq \{\}$ est compatible avec les 3 cas;
 - Le jugement est valide.





Les diagrammes de Venn (cas d'un jugement invalide)

- On suppose (P_1) et (P_2) inchangés ;
- (C') remplace (C) ;
- (C') : Tout C est B
 - La contrainte $C \subseteq B$ est incompatible avec les 1^{er} cas ;
 - Le jugement n'est pas valide.





Les diagrammes de Venn (exemple 1)

- (P₁) Aucun chat n'est humain
- (P₂) Tout humain est bipède
- (C) Donc aucun chat n'est bipède



Les diagrammes de Venn (exemple 1)

- (P₁) Aucun chat n'est humain
chats \cap *humains* = {}
- (P₂) Tout humain est bipède
- (C) Donc aucun chat n'est bipède

Chats

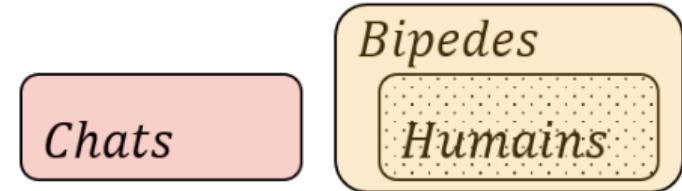
Humains



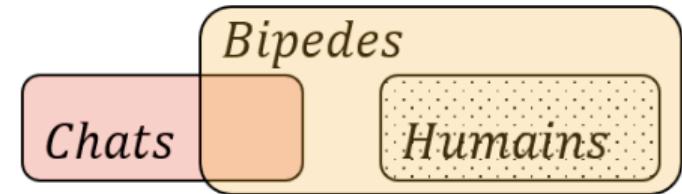
Les diagrammes de Venn (exemple 1)

- (P₁) Aucun chat n'est humain
 $chats \cap humains = \{\}$
- (P₂) Tout humain est bipède
 $humains \subseteq bipedes$
- (C) Donc aucun chat n'est bipède

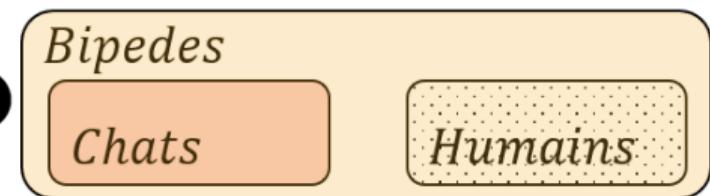
1



2



3





Les diagrammes de Venn (exemple 1)

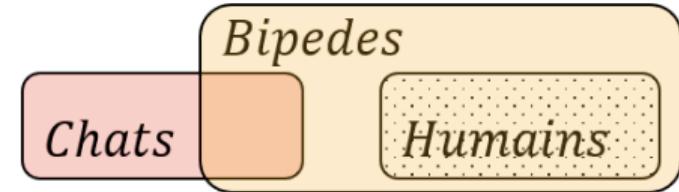
- (P₁) Aucun chat n'est humain
 $chats \cap humains = \{\}$

1



- (P₂) Tout humain est bipède
 $humains \subseteq bipedes$

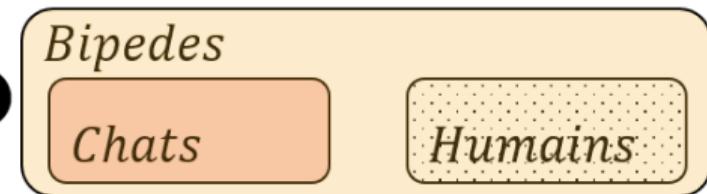
2



- (C) Donc aucun chat n'est bipède

$chats \cap bipedes = \{\}$

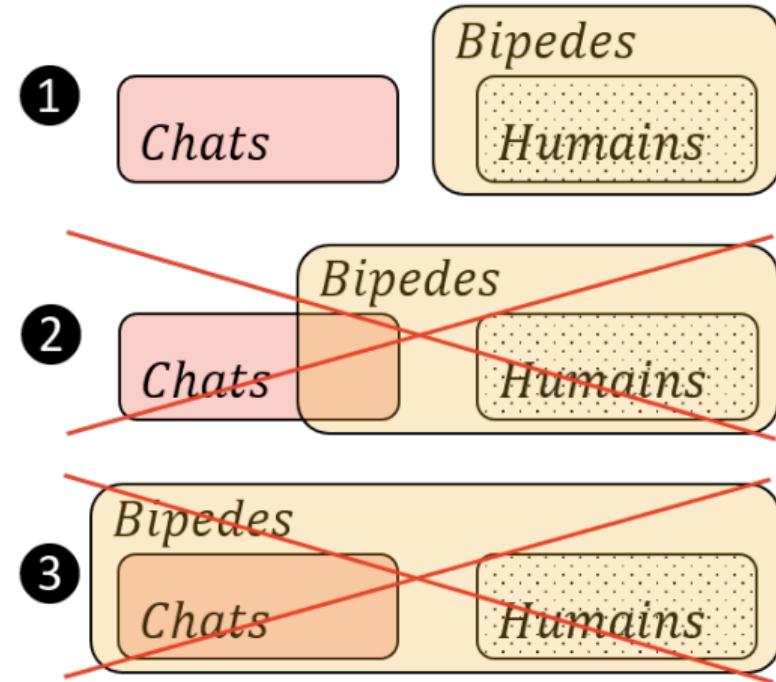
3





Les diagrammes de Venn (exemple 1)

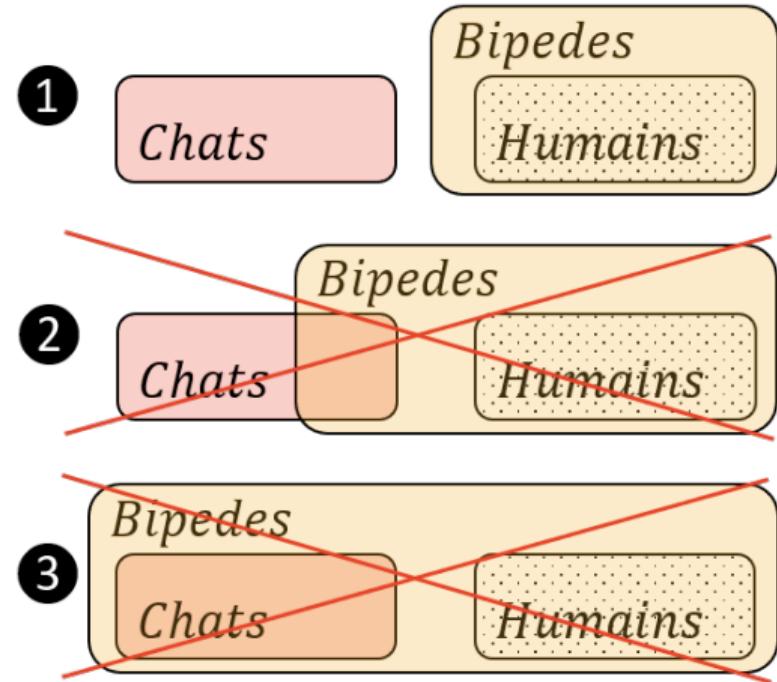
- (P₁) Aucun chat n'est humain
 $chats \cap humains = \{\}$
- (P₂) Tout humain est bipède
 $humains \subseteq bipedes$
- (C) Donc aucun chat n'est bipède
 $chats \cap bipedes = \{\}$
➡ cas ② et ③ incompatibles avec (C)





Les diagrammes de Venn (exemple 1)

- (P₁) Aucun chat n'est humain
 $chats \cap humains = \{\}$
- (P₂) Tout humain est bipède
 $humains \subseteq bipedes$
- (C) Donc aucun chat n'est bipède
 $chats \cap bipedes = \{\}$
 ➡ cas ② et ③ incompatibles avec (C)
 ➡ raisonnement non valide





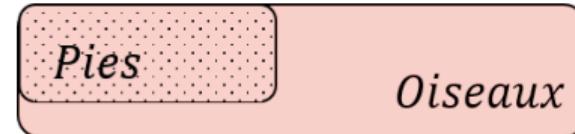
Les diagrammes de Venn (exemple 2)

- (P_1) Les pies sont des oiseaux
- (P_2) Les pies sont bicolores
- (C) Donc quelque oiseau est bicolore



Les diagrammes de Venn (exemple 2)

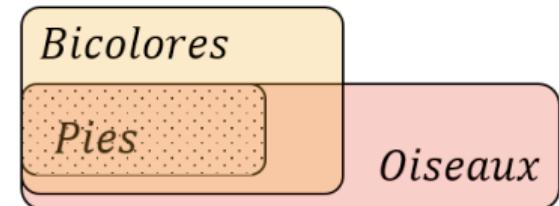
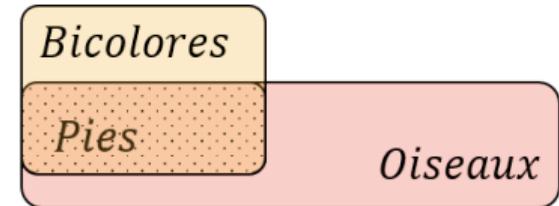
- (P₁) Les pies sont des oiseaux
pies ⊆ oiseaux
- (P₂) Les pies sont bicolores
- (C) Donc quelque oiseau est bicolore





Les diagrammes de Venn (exemple 2)

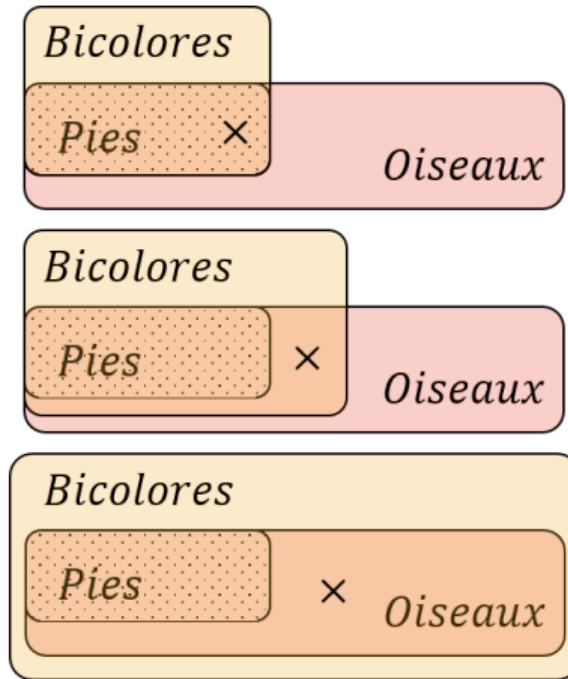
- (P_1) Les pies sont des oiseaux
 $\text{pies} \subseteq \text{oiseaux}$
- (P_2) Les pies sont bicolores
 $\text{pies} \subseteq \text{bicolores}$
- (C) Donc quelque oiseau est bicolore





Les diagrammes de Venn (exemple 2)

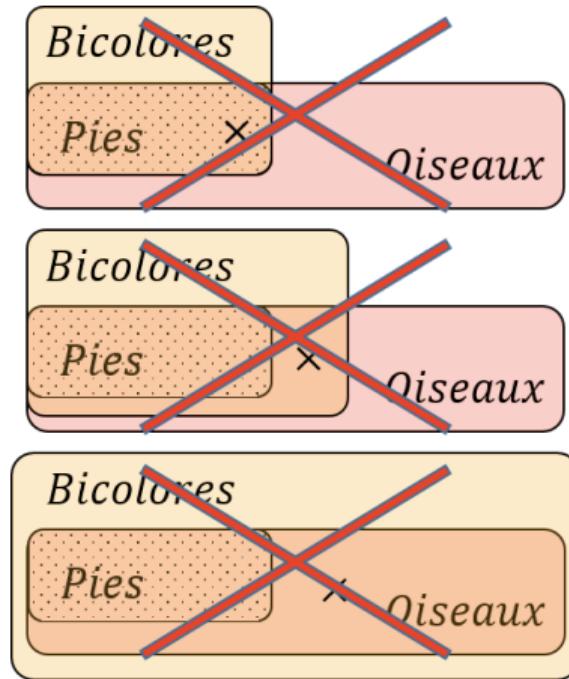
- (P_1) Les pies sont des oiseaux
 $\text{pies} \subseteq \text{oiseaux}$
- (P_2) Les pies sont bicolores
 $\text{pies} \subseteq \text{bicolores}$
- (C) Donc quelque oiseau est bicolore
 $\text{oiseaux} \cap \text{bicolores} \neq \{\}$





Les diagrammes de Venn (exemple 2)

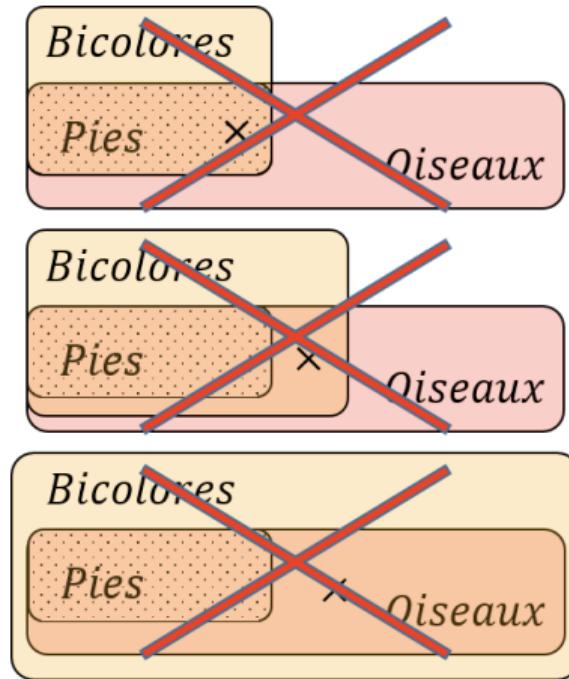
- (P_1) Les pies sont des oiseaux
 $\text{pies} \subseteq \text{oiseaux}$
- (P_2) Les pies sont bicolores
 $\text{pies} \subseteq \text{bicolores}$
- (C) Donc quelque oiseau est bicolore
 $\text{oiseaux} \cap \text{bicolores} \neq \{\}$
➡ mais ni (P_1) ni (P_2) imposent que ces ensembles aient au moins un élément





Les diagrammes de Venn (exemple 2)

- (P_1) Les pies sont des oiseaux
 $\text{pies} \subseteq \text{oiseaux}$
- (P_2) Les pies sont bicolores
 $\text{pies} \subseteq \text{bicolores}$
- (C) Donc quelque oiseau est bicolore
 $\text{oiseaux} \cap \text{bicolores} \neq \{\}$
 - ➡ mais ni (P_1) ni (P_2) imposent que ces ensembles aient au moins un élément
 - ➡ **raisonnement non valide**





Les diagrammes de Venn (exemple 2 : contre-exemple)

- Si $p_{ies} = \{\}$, alors il est inclus dans tous les ensembles ;
- y compris quand ces ensembles sont disjoints ;



Les diagrammes de Venn (exemple 2 : contre-exemple)

- Si $Pies = \{\}$, alors il est inclus dans **tous** les ensembles ;
- y compris quand ces ensembles sont disjoints ;
- autrement dit, le diagramme ci-contre :
 - satisfait (P_1) et (P_2) ;
 - mais falsifie (C) ;
- ▶ et est appelé **contre-exemple** (à la validité du raisonnement énoncé).

Bicolores

$Pies = \{ \}$

Oiseaux



Les types de règles de raisonnement

- Les règles de type **déductif** (ou **règles déductives**) n'utilisent :
 - que la règle de déduction,
 - exemple : $\frac{A, A \rightarrow B}{B}$ (« de A et A implique B , on infère B »);
- les règles (de type) **hypothétique(s)** utilisent :
 - au moins une règle d'abduction,
 - exemple : $\frac{B, A \rightarrow B}{A}$ (« de B et A implique B , on infère A »);
 - ou au moins une règle d'induction,
 - exemple : $\frac{A, B}{A \rightarrow B}$ (« de A et B , on infère A implique B »).



Les types de règles de raisonnement (remarques)

- Les règles de raisonnement ne sont pas nécessairement fondées sur l'implication ;
 - par exemple, la règle de **résolution** : $\frac{A, \neg A \vee B}{B}$ est une autre règle de **déduction** ;
- seules les règles **déductives** sont **valides** (voir Slide 13) ;
- par exemple : la valuation v telle que $I_v(A) = 0$ et $I_v(B) = 1$:
 - satisfait les hypothèses de la règle **abductive** $\frac{B, A \rightarrow B}{A}$,
 - mais falsifie sa conclusion.



Introduction

Focus sur la logique des propositions



Les origines

- Les stoïciens ont remarqué l'existence de paradoxes logiques.
Par exemple :
 - « Cette phrase est fausse » ➔ vraie ou fausse ?
 - « Un crétois dit que tous les crétois sont des menteurs »
➔ vraie ou fausse ?
 - ...
- Ils choisissent de ne pas analyser les propositions, qui sont considérées comme formant un bloc :
➔ on parle de **logique des propositions**.



Représentation d'un énoncé

- Les différentes **propositions** (non-analysées) d'un énoncé seront représentées :
 - par différents symboles (appelées **variables propositionnelles**),
 - reliés entre eux par des **connecteurs logiques** : ou, et, si ... alors, non, etc. ;
- constituant ce qu'on appellera des **formules**.



Quelques règles de raisonnement

- Les règles suivantes sont toutes des **règles de déduction** (voir Slide 34) :
 - Si A alors B . Or A . Donc B . (Modus Ponens)
 - Si A alors B . Or non B . Donc non A . (Modus Tollens)
 - Si A alors B . Donc Si non B alors non A . (contraposition)
 - A ou B . Or non A . Donc B . (règle disjonctive)
 - A et B . Donc A . (règle conjonctive)
 - ...
- qui sont considérées à ce titre comme **valides** (voir Slide 13), ce qu'on ne démontrera pas.



Exemples de jugements

- Dans les jugements suivants, quelle **règle de raisonnement** est utilisée ?
 - Si j'ai de la fièvre alors je suis malade. Or j'ai de la fièvre. Donc je suis malade ;



Exemples de jugements

- Dans les jugements suivants, quelle **règle de raisonnement** est utilisée ?
 - Si j'ai de la fièvre alors je suis malade. Or j'ai de la fièvre.
Donc je suis malade ; (➡ modus ponens)
 - S'il pleut alors le sol est mouillé. Or le sol n'est pas mouillé.
Donc il ne pleut pas ;



Exemples de jugements

- Dans les jugements suivants, quelle **règle de raisonnement** est utilisée ?
 - Si j'ai de la fièvre alors je suis malade. Or j'ai de la fièvre.
Donc je suis malade ; (➡ modus ponens)
 - S'il pleut alors le sol est mouillé. Or le sol n'est pas mouillé.
Donc il ne pleut pas ; (➡ modus tollens)
 - Tu as l'as de pique ou l'as de carreau. Or tu n'as pas l'as de carreau.
Donc tu as l'as de pique.



Exemples de jugements

- Dans les jugements suivants, quelle **règle de raisonnement** est utilisée ?
 - Si j'ai de la fièvre alors je suis malade. Or j'ai de la fièvre.
Donc je suis malade ; (➡ modus ponens)
 - S'il pleut alors le sol est mouillé. Or le sol n'est pas mouillé.
Donc il ne pleut pas ; (➡ modus tollens)
 - Tu as l'as de pique ou l'as de carreau. Or tu n'as pas l'as de carreau.
Donc tu as l'as de pique. (➡ règle disjonctive)



Exercice sur des jugements complexes

- Un **jugement complexe** est une séquence de jugements ;



Exercice sur des jugements complexes

- Un **jugement complexe** est une séquence de jugements ;
- il est **valide** ssi chacun des jugements utilisés est valide ;



Exercice sur des jugements complexes

- Un **jugement complexe** est une séquence de jugements ;
- il est **valide** ssi chacun des jugements utilisés est valide ;
- lesquels des jugements complexes ci-dessous sont valides ?
Pourquoi ? (indiquer les règles de raisonnement utilisées)
 - 1 S'il y a du verglas, la route est glissante.
Si la route est glissante, les camions ne roulent pas.
Or les camions roulent.
Donc il n'y a pas de verglas.



Exercice sur des jugements complexes

- Un **jugement complexe** est une séquence de jugements ;
- il est **valide** ssi chacun des jugements utilisés est valide ;
- lesquels des jugements complexes ci-dessous sont valides ?
Pourquoi ? (indiquer les règles de raisonnement utilisées)
 - 1 S'il y a du verglas, la route est glissante.
Si la route est glissante, les camions ne roulent pas.
Or les camions roulent.
Donc il n'y a pas de verglas.
 - 2 Quand il ne pleut pas, le sol est sec.
Quand il n'y a pas d'humidité dans l'air, il ne pleut pas.
Or l'air est humide.
Donc le sol n'est pas sec.



Remarque sur « Si . . . alors . . . »

- Cet opérateur ne marque pas nécessairement une **relation de cause à effet**
 - il peut aussi représenter une simple **simultanéité**
 - Dans « Si j'ai de la fièvre alors je suis malade », la fièvre ne cause pas la maladie (c'est même l'inverse) ;
- mais plutôt que, quand on a de la fièvre, alors **nécessairement** on est (en même temps) malade ;
- ou encore, qu'il **suffit** d'avoir de la fièvre pour être déclaré malade.



Conditions nécessaire, suffisante

- Dans « Si j'ai de la fièvre alors je suis malade », on dit que :
 - « être malade » est une condition nécessaire de « avoir de la fièvre » ;
 - « avoir de la fièvre » est une condition suffisante de « être malade » ;
- De manière générale, dans « Si A alors B » :
 - B est une condition nécessaire de A ;
 - A est une condition suffisante de B .
- « Pas de fumée sans feu et pas de feu sans fumée » : statut de « fumée » vs « feu » ?



Conditions nécessaire, suffisante (exemples)

- Quelle est le rôle de « J'ai un bon tarif » dans :
 - 1 « Je prends l'avion *si* j'ai un bon tarif »
 - 2 « Je prends l'avion *seulement si* j'ai un bon tarif »
 - 3 « Je prends l'avion *si et seulement si* j'ai un bon tarif »
- Attention : le langage courant ne fait pas toujours ces distinctions. C'est source de sous-entendus et donc d'ambiguités.



Conditions nécessaire, suffisante (exemples)

- Quelle est le rôle de « J'ai un bon tarif » dans :
 - 1 « Je prends l'avion *si* j'ai un bon tarif »
 - ➡ condition suffisante ;
 - 2 « Je prends l'avion *seulement si* j'ai un bon tarif »
 - 3 « Je prends l'avion *si et seulement si* j'ai un bon tarif »
- Attention : le langage courant ne fait pas toujours ces distinctions. C'est source de sous-entendus et donc d'ambiguités.



Conditions nécessaire, suffisante (exemples)

- Quelle est le rôle de « J'ai un bon tarif » dans :
 - 1 « Je prends l'avion *si* j'ai un bon tarif »
 - ➡ condition suffisante ;
 - 2 « Je prends l'avion *seulement si* j'ai un bon tarif »
 - ➡ condition nécessaire ;
 - 3 « Je prends l'avion *si et seulement si* j'ai un bon tarif »
- Attention : le langage courant ne fait pas toujours ces distinctions. C'est source de sous-entendus et donc d'ambiguités.



Conditions nécessaire, suffisante (exemples)

- Quelle est le rôle de « J'ai un bon tarif » dans :
 - 1 « Je prends l'avion *si* j'ai un bon tarif »
 - ➡ condition suffisante ;
 - 2 « Je prends l'avion *seulement si* j'ai un bon tarif »
 - ➡ condition nécessaire ;
 - 3 « Je prends l'avion *si et seulement si* j'ai un bon tarif »
 - ➡ condition nécessaire et suffisante.
- Attention : le langage courant ne fait pas toujours ces distinctions. C'est source de sous-entendus et donc d'ambiguités.



Introduction

Quelques personnages emblématiques



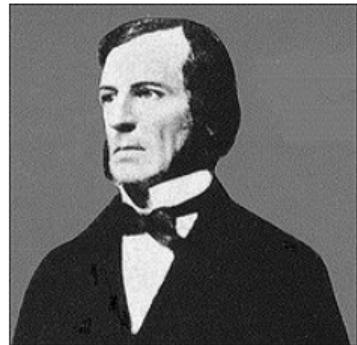
Leibniz (1646 - 1716)



- Codage binaire des nombres ;
- Construction de l'un des premiers calculateurs (mécanique, décimal) ;
- Développement du calcul infinitésimal (en concurrence avec Newton) ;
- *Ars combinatoria* : l'art de dériver des vérités de manière *calculatoire*, basé sur :
 - un langage mathématique non-ambigu ;
 - un calcul manipulant la *characteristica*.



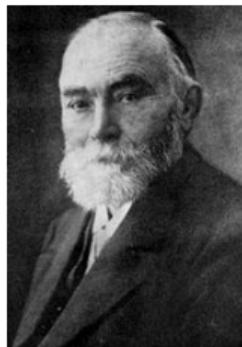
Boole (1815 - 1864)



- Livre : *An Investigation of the Laws of Thought*
 - Traitement algébrique de la logique propositionnelle
 - Procédure de décision pour la logique propositionnelle



Frege (1848 - 1925)



- Fondateur de la logique **moderne** :
 - les connecteurs « essentiels » de la logique des propositions : \rightarrow et \neg ;
 - les quantificateurs de la logique des prédictats ;
 - un calcul formel ;
- **Begriffsschrift** (1879) :
 - Distinction entre **formule** (proposition qui peut être vraie ou fausse), et **jugement** (noté \vdash , formule dont on constate la vérité dans un calcul donné).



Russell (1872 - 1970)



- Livre *Principia Mathematica* (1910 - 1913) : formalisation des mathématiques à partir de quelques notions élémentaires ;
- Découverte d'un paradoxe (1903) dans la théorie des ensembles de Cantor
 - ➡ crise des fondements de la mathématique : consistance des axiomes ?
- (Prix Nobel de littérature, emprisonné suite à des campagnes contre l'armement nucléaire...)



Gödel (1906 - 1978)



Le cataclysme de *Sur des propositions indécidables de Principia Mathematica* (1931) :

- Toute théorie mathématique *suffisamment expressive* est **incomplète** (il existe des énoncés vrais non démontrables) ou **contradictoire** ;
- surtout, elle ne permet pas de démontrer sa propre cohérence ;
 - ➡ échec du programme rationaliste dans la tradition Leibniz - Hilbert.



Résumé

- Quelques repères dans un paysage vaste :
 - Syllogisme d'Aristote : possibilité de raisonner en s'appuyant sur la **syntaxe**, sans connaissance de la **sémantique** ;
 - Leibniz : raisonnement exécutable par une machine (**calcul**) ;
 - Boole, Frege : forme moderne de la logique ;
 - Russell, Gödel : mise à mal du programme rationaliste
- Pour plus d'information sur le développement de la logique au tournant du XX^e siècle voir l'ouvrage de Jean Van Heijenoort **From Frege to Gödel**.



Introduction

Applications en informatique



Applications de la logique

- Importantes en informatique :
 - intelligence artificielle ;
 - sûreté de fonctionnement ;
 - sécurité.
- Objectifs de cette section :
 - Comprendre le rôle de la logique dans l'informatique ;
 - Gagner une intuition des méthodes mises en œuvre.



Intelligence artificielle (1/2)

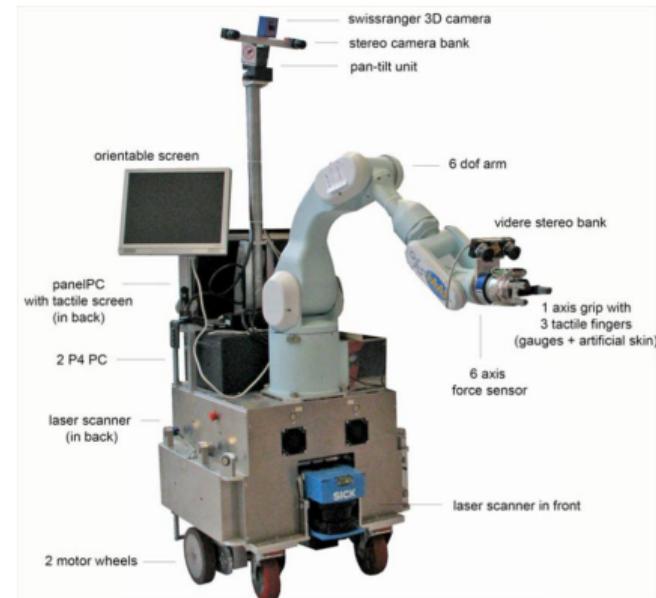
- But :
 - mieux comprendre le fonctionnement de l'intelligence humaine ;
 - découvrir les facettes d'un comportement rationnel.
- Applications :
 - simuler le comportement humain : économie, écologie, médecine, etc. ;
 - guider l'action humaine : aide au diagnostic (médical, informatique, etc.) ;
 - imiter le comportement humain : robotique, jeux (de plateaux, sérieux, vidéos...), agent conversationnels animés (ACA), etc.



Intelligence artificielle (2/2)

■ Robotique cognitive :

- contrôle des actions (comment déplacer un livre d'une chaise sur une table ?);
- contrôle des mouvements (comment se déplacer d'une salle à une autre ?);
- interaction humain-machine (comment interpréter une commande en langage naturel ?)



Robot Jido (LAAS)



Sûreté de fonctionnement (1/3)

- But :
 - S'assurer du bon fonctionnement du matériel et logiciel de systèmes critiques (transports, nucléaire, systèmes médicaux, etc.).
- Contexte :
 - systèmes embarqués de plus en plus sophistiqués ;
 - des vies soumises au bon fonctionnement du système (voir *fly by wire* de l'A330).





Sûreté de fonctionnement (2/3)



- Explosion de la fusée Ariane 5 (juin 1996) :
 - 40s après son lancement ;
 - à bord, 4 satellites de recherche ;
 - dispersion de 745 tonnes de débris (toxiques) ;
 - coût des dégâts : 290 M €.
- Contexte :
 - 1^{er} lancement de la fusée ;
 - réutilisation du logiciel d'Ariane 4 (inadapté)
 - ➡ dépassement arithmétique.



Sûreté de fonctionnement (3/3)

- Exemple du Therac-25 (1985 – 1987) :
 - lors d'une radiothérapie, 6 personnes reçoivent une surdose massive (facteur 100) ;
 - conséquence : 3 décès ;
 - causes multiples, surtout 1 problème de synchronisation de deux tâches.
- Mai 2007, à Toulouse :
 - 145 personnes irradiées au CHU de Rangueil ;
 - « c'est encore une fois une déficience de l'informatique qui serait en cause (...) l'étalonnage était mal fait », *Le Parisien*.



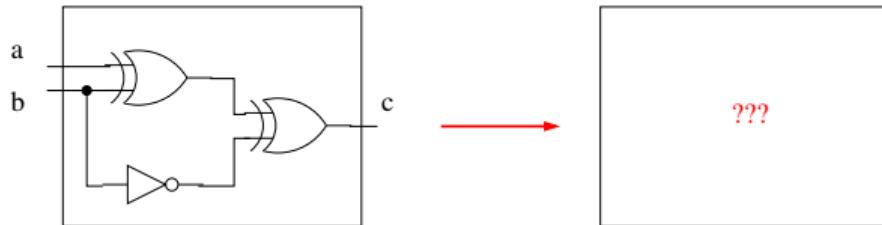
Sécurité

- Problème des virus et vers informatiques infectant des systèmes connectés par Internet ;
- exemple du **ver Slammer** (2003) infectant 75 000 machines en 30 mn :
 - conséquence : blocage du système de surveillance de la centrale nucléaire Davis-Besse (Ohio) pendant 5h ;
 - causes de la vulnérabilité :
 - langages de programmation de bas niveau (assembleur, C) ;
 - incompréhension du fonctionnement des protocoles de communication ;
 - cryptage insuffisant.



Architecture

- But :
 - simplification de circuits électroniques ;
- Problème :
 - Utiliser moins de composants.





Les systèmes à états

- Systèmes aussi appelés « systèmes de transitions (d'états) » ;
- ils sont composés :
 - d'états en nombre fini ;
 - de transitions entre ces états, étiquetées ou non ;
- mathématiquement, ce sont des graphes orientés ;
- ils permettent la modélisation de systèmes ou de calculs selon des règles définies ➔ implémentation machine ;
- la logique est bien adaptée pour les décrire et démontrer leurs propriétés.



Exemple de système à états (cadre)

- Un **état** est caractérisé par des cubes A, B, C, D empilés arbitrairement sur trois positions P_1, P_2, P_3 ;
- **opérations** : le robot peut déplacer un seul cube au sommet d'une pile vers le sommet d'une autre pile;
- **but** : état où A, B, C, D sont empilés dans l'ordre sur P_1 ;
- **solution** : séquence d'opérations conduisant à l'état but.

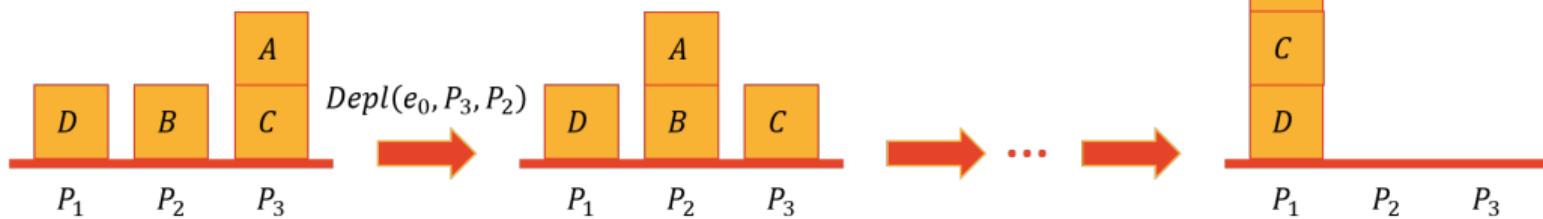


Exemple de système à états (langage de modélisation)

- $\mathcal{C} = \{A, B, C, D\}$ est l'ensemble des cubes ;
- $\mathcal{P} = \{P_1, P_2, P_3\}$ est l'ensemble des positions ;
- $\mathcal{E} = \{e_0, e_1, \dots, e_N : N \in \mathbb{N}\}$ est l'ensemble des $N + 1$ états possibles ;
- *Sur(x, y, ε)* est vrai quand le cube x est sur un autre cube ou un bloc $y \in \mathcal{C} \cup \mathcal{P}$ dans l'état $ε$;
- *Depl(e, p, p')* convertit l'état e en un état e' en déplaçant le sommet de la position p vers celui d'une autre position p' .



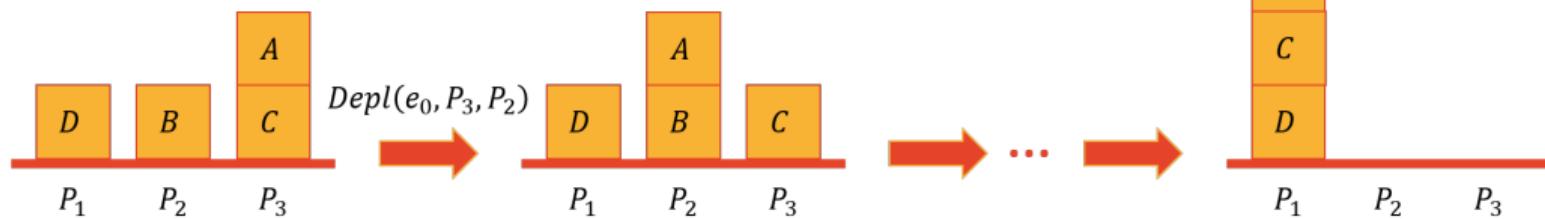
Exemple de système à états (description du système)



- Initialement, la situation courante (dans l'état e_0) est :



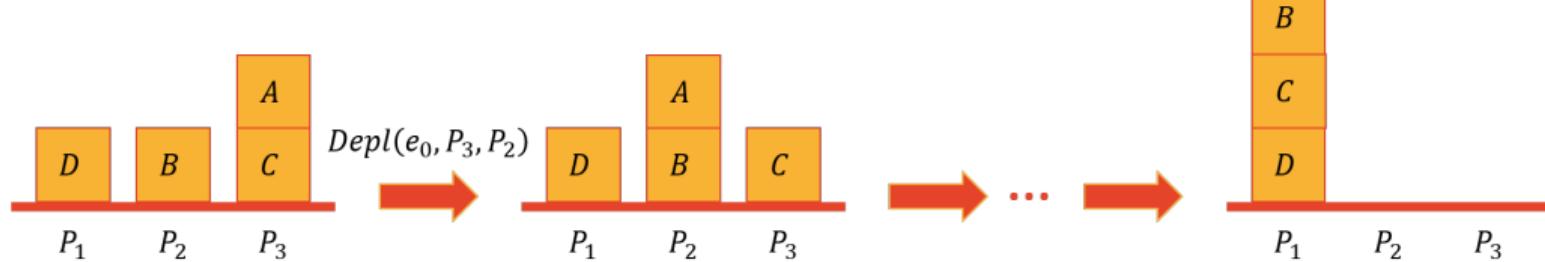
Exemple de système à états (description du système)



- Initialement, la situation courante (dans l'état e_0) est :
 $Sur(A, C, e_0)$, $Sur(B, P_2, e_0)$, $Sur(C, P_3, e_0)$, $Sur(D, P_1, e_0)$;
- état $e_1 =$



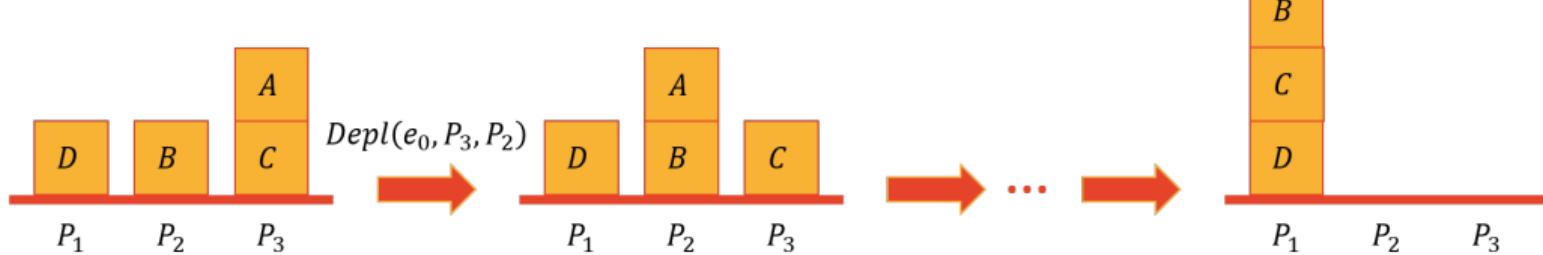
Exemple de système à états (description du système)



- Initialement, la **situation courante** (dans l'état e_0) est :
 $Sur(A, C, e_0)$, $Sur(B, P_2, e_0)$, $Sur(C, P_3, e_0)$, $Sur(D, P_1, e_0)$;
- état $e_1 = Depl(e_0, P_3, P_2)$;
- séquence d'opérations menant au but ?



Exemple de système à états (description du système)



- Initialement, la **situation courante** (dans l'état e_0) est :
 $Sur(A, C, e_0)$, $Sur(B, P_2, e_0)$, $Sur(C, P_3, e_0)$, $Sur(D, P_1, e_0)$;
- état $e_1 = Depl(e_0, P_3, P_2)$;
- séquence d'opérations menant au but ? $Depl(e_0, P_3, P_2)$,
 $Depl(e_1, P_3, P_1)$, $Depl(e_2, P_2, P_3)$, $Depl(e_3, P_2, P_1)$, $Depl(e_4, P_3, P_1)$.



La logique et les systèmes à états

- Dans le cadre d'une recherche de solution, on peut utiliser la logique pour :
 - décrire le but : « **Pour tout** état initial, **il existe** un état final ayant la propriété ... » ;
 - faire une preuve constructive \rightsquigarrow séquence d'opérations.
- Répondre à des questions plus générales :
 - comment trouver une solution pour n'importe quel état initial ?
 - ▶ stratégie de preuve ;
 - est-ce qu'on peut **assurer** qu'une solution existe toujours ?
 - ▶ complétude de la stratégie ;
 - et avec seulement deux positions ?



Résumé

- La logique est essentielle dans des domaines tels que :
 - l'intelligence artificielle ;
 - la sûreté de fonctionnement et la sécurité.
- La logique permet de décrire :
 - l'équivalence fonctionnelle et comportementale (par exemple, les circuits) ;
 - les propriétés des systèmes à états (recherche d'une séquence d'actions conduisant, à partir d'un état initial, à l'état but souhaité ➡ planification).



Introduction

Preuve par induction



Définition intuitive

- Voici une définition **inductive** de poupée gigogne :
 - Ceci est une poupée gigogne...



- Étant donnée une poupée gigogne, si on l'enferme dans une figurine creuse de poupée alors on obtient une poupée gigogne





Induction et métathéorie

- L'induction est un concept très général et fondamental dans les mathématiques, l'informatique et la logique ;
- Dans ce cours, quand nous étudions l'induction, nous faisons de la **métathéorie** :
 - induction = théorie ayant la théorie logique pour objet d'étude.



Les 3 exemples qui seront étudiés

- 1 L'ensemble \mathbb{N} des entiers naturels en base 10 (0, 1, 7, 42, ...);



Les 3 exemples qui seront étudiés

- 1 L'ensemble \mathbb{N} des entiers naturels en base 10 (0, 1, 7, 42, ...);
- 2 les expressions arithmétiques (+, −, ×) parenthésées sur les entiers avec variables. Ex. : $((2 + x) \times y)$;



Les 3 exemples qui seront étudiés

- 1 L'ensemble \mathbb{N} des entiers naturels en base 10 (0, 1, 7, 42, ...);
- 2 les expressions arithmétiques (+, −, ×) parenthésées sur les entiers avec variables. Ex. : $((2 + x) \times y)$;
- 3 les entiers naturels représentés comme 0, $(S0)$, $(S(S0))$, ...;



Les 3 exemples qui seront étudiés

- 1 L'ensemble \mathbb{N} des entiers naturels en base 10 (0, 1, 7, 42, ...);
 - 2 les expressions arithmétiques (+, −, ×) parenthésées sur les entiers avec variables. Ex. : $((2 + x) \times y)$;
 - 3 les entiers naturels représentés comme 0, $(S0)$, $(S(S0))$, ...;
- ▶ nous appliquerons ces notions à la logique dans le chapitre suivant pour définir les formules de la logique propositionnelle.



Les 3 aspects de l'induction étudiés

- Pour ce cours, 3 aspects seront étudiés :
 - la **définition inductive** d'un ensemble ;
 - les **fonctions récursives** ;
 - les **preuves par induction**.



Ex. 1 : représenter \mathbb{N}

- **But** : trouver des règles décrivant toutes les constructions permises, et seulement celles-ci.
- **Proposition** :



Ex. 1 : représenter \mathbb{N}

- **But** : trouver des règles décrivant toutes les constructions permises, et seulement celles-ci.
- **Proposition** :
 - soit $\mathcal{N} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ tel que $\mathcal{N} \subseteq \mathbb{N}$;
 - si $x \in \mathbb{N}$ et $a \in \mathcal{N}$ alors l'assemblage $xa \in \mathbb{N}$;
 - ▶ x et a ne sont pas des symboles autorisés dans un nombre ! Ces **méta-variables** représentent un nombre (x) et un chiffre (a).



Ex. 1 : représenter \mathbb{N}

- **But** : trouver des règles décrivant toutes les constructions permises, et seulement celles-ci.
- **Proposition** :
 - soit $\mathcal{N} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ tel que $\mathcal{N} \subseteq \mathbb{N}$;
 - si $x \in \mathbb{N}$ et $a \in \mathcal{N}$ alors l'assemblage $xa \in \mathbb{N}$;
 - ▶ x et a ne sont pas des symboles autorisés dans un nombre ! Ces **méta-variables** représentent un nombre (x) et un chiffre (a).
- une **définition inductive** (ou **par induction**) comprend :
 - des éléments de base (ici les chiffres de \mathcal{N});
 - un ensemble de règles d'assemblage.



Ex. 2 : expressions arithmétiques

- Comment représenter l'ensemble XP des expressions algébriques (par ex. : $((2 + x) \times y)$ avec $x, y, \dots \in \mathcal{V}$, l'ensemble des variables entières) ?
- XP est le **plus petit** ensemble tel que :
 - 1 si $x \in \mathcal{V}$, $x \in XP$ (Variable)
 - 2 si $c \in \mathbb{N}$, $c \in XP$ (Constante)
 - 3 si $A \in XP$, $(-A) \in XP$ (Opposé)
 - 4 si $A, B \in XP$: $(A + B) \in XP$, $(A - B) \in XP$, $(A \times B) \in XP$ (Opération)
- Cette définition utilise des **variables** représentant des entiers, mais aussi des **méta-variables** représentant des **expressions**.



Principe d'une définition inductive

- Une **définition inductive** rend précise l'idée que le tout est composé de parties, elles-mêmes composées de sous-parties, etc. jusqu'aux sous-parties indivisibles ;



Principe d'une définition inductive

- Une **définition inductive** rend précise l'idée que le tout est composé de parties, elles-mêmes composées de sous-parties, etc. jusqu'aux sous-parties indivisibles ;
- dans l'exemple précédent, le fait que XP soit le **plus petit** ensemble tel que (...) interdit tout élément non généré à partir des règles :
 - $(1 \bowtie x) \notin XP$, parce qu'aucune règle ne génère $(1 \bowtie x)$;



Principe d'une définition inductive

- Une **définition inductive** rend précise l'idée que le tout est composé de parties, elles-mêmes composées de sous-parties, etc. jusqu'aux sous-parties indivisibles ;
- dans l'exemple précédent, le fait que XP soit le **plus petit** ensemble tel que (...) interdit tout élément non généré à partir des règles :
 - $(1 \bowtie x) \notin XP$, parce qu'aucune règle ne génère $(1 \bowtie x)$;
 - pour les mêmes raisons : $\notin XP$,



Principe d'une définition inductive

- Une **définition inductive** rend précise l'idée que le tout est composé de parties, elles-mêmes composées de sous-parties, etc. jusqu'aux sous-parties indivisibles ;
- dans l'exemple précédent, le fait que XP soit le **plus petit** ensemble tel que (...) interdit tout élément non généré à partir des règles :
 - $(1 \bowtie x) \notin XP$, parce qu'aucune règle ne génère $(1 \bowtie x)$;
 - pour les mêmes raisons : $\notin XP$, $\notin XP$,



Principe d'une définition inductive

- Une **définition inductive** rend précise l'idée que le tout est composé de parties, elles-mêmes composées de sous-parties, etc. jusqu'aux sous-parties indivisibles ;
- dans l'exemple précédent, le fait que XP soit le **plus petit** ensemble tel que (...) interdit tout élément non généré à partir des règles :
 - $(1 \bowtie x) \notin XP$, parce qu'aucune règle ne génère $(1 \bowtie x)$;
 - pour les mêmes raisons : $\notin XP$, $\notin XP$, $\notin XP$, ...



Principe d'une définition inductive

- Une **définition inductive** rend précise l'idée que le tout est composé de parties, elles-mêmes composées de sous-parties, etc. jusqu'aux sous-parties indivisibles ;
- dans l'exemple précédent, le fait que XP soit le **plus petit** ensemble tel que (...) interdit tout élément non généré à partir des règles :
 - $(1 \bowtie x) \notin XP$, parce qu'aucune règle ne génère $(1 \bowtie x)$;
 - pour les mêmes raisons : $\notin XP$, $\notin XP$, $\notin XP$, ...
 - $((1 + (x - 3)) + (- 61)) \notin XP$, parce qu' il manque) à la fin.



Ex. 3 : définition inductive des nombres naturels (principe)

- On souhaite représenter les entiers à l'aide du successeur d'un entier :



Représentation décimale	0	1	2	3
-------------------------	---	---	---	---

Représentation unaire	0	$(S0)$	$(S(S0))$	$(S(S(S0)))$
-----------------------	---	--------	-----------	--------------



Ex. 3 : définition inductive des nombres naturels (conditions)

- L'ensemble \mathbb{N} est **le plus petit ensemble** qui respecte les conditions suivantes :

- 1 $0 \in \mathbb{N}$ (Zéro)
- 2 Si la métavariable $n \in \mathbb{N}$, alors $(Sn) \in \mathbb{N}$ (Successseur)



Ex. 3 : définition inductive des nombres naturels (conditions)

- L'ensemble \mathbb{N} est **le plus petit ensemble** qui respecte les conditions suivantes :
 - 1 $0 \in \mathbb{N}$ (Zéro)
 - 2 Si la métavariable $n \in \mathbb{N}$, alors $(Sn) \in \mathbb{N}$ (Successseur)
- Là encore, « plus petit ensemble » sert à éliminer les ensembles qui :
 - respecteraient les conditions 1) et 2) donc contiendraient $0, (S0), (S(S0)), \dots$
 - mais contiendraient aussi des éléments indésirables.



Définition récursive d'une fonction

- Fonction récursive : fonction définie récursivement (*i.e.* dont la définition fait appel à elle-même).
- exemple de la fonction d'addition $n + m$ où n et m sont des entiers définis à la manière de l'exemple 3 :
 - $(0 + m) = m$ (Zéro +)
 - $((Sn) + m) = (S(n + m))$ (Successeur +)



Preuve par induction (principe)

- Principe d'induction sur les nombres naturels :
 - c'est le cas pour 0 (Base)
 - quand c'est le cas pour un nombre,
c'est aussi le cas pour son successeur (Héritéité)
- Les nombres naturels ainsi définis partagent une caractéristique, une propriété donnée.



Preuves par induction (schéma)

- Soit \mathcal{P} une propriété.
- Schéma d'induction sur les nombres naturels :
 - si $\mathcal{P}(0)$ (Base : Zéro)
 - et si pour n , $\mathcal{P}(n)$ implique $\mathcal{P}((Sn))$ (Hérédité : Successeur)
 - alors on peut conclure que : $\forall n. \mathcal{P}(n)$.



Preuves par induction (exemple)

- On veut montrer que $\forall n. (n + 0 = n)$.
- Pour cela, il faut :
 - 1 identifier la propriété P :



Preuves par induction (exemple)

- On veut montrer que $\forall n. (n + 0 = n)$.
- Pour cela, il faut :
 - 1 identifier la propriété \mathcal{P} : ici, $\mathcal{P}(n)$ est $(n + 0 = n)$ pour tout n ;



Preuves par induction (exemple)

- On veut montrer que $\forall n. (n + 0 = n)$.
- Pour cela, il faut :
 - 1 identifier la propriété P : ici, $P(n)$ est $(n + 0 = n)$ pour tout n ;
 - 2 prouver le cas de base :



Preuves par induction (exemple)

- On veut montrer que $\forall n. (n + 0 = n)$.
- Pour cela, il faut :
 - 1 identifier la propriété P : ici, $P(n)$ est $(n + 0 = n)$ pour tout n ;
 - 2 prouver le cas de base : ici, $P(0)$ est $0 + 0 = 0$ (Base : Zéro)



Preuves par induction (exemple)

- On veut montrer que $\forall n. (n + 0 = n)$.
- Pour cela, il faut :
 - 1 identifier la propriété P : ici, $P(n)$ est $(n + 0 = n)$ pour tout n ;
 - 2 prouver le cas de base : ici, $P(0)$ est $0 + 0 = 0$ (Base : Zéro)
 - 3 prouver l'hérédité :



Preuves par induction (exemple)

- On veut montrer que $\forall n. (n + 0 = n)$.
- Pour cela, il faut :
 - 1 identifier la propriété P : ici, $P(n)$ est $(n + 0 = n)$ pour tout n ;
 - 2 prouver le cas de base : ici, $P(0)$ est $0 + 0 = 0$ (Base : Zéro)
 - 3 prouver l'hérédité : ici : si $P(n)$, alors $P((Sn))$



Preuves par induction (exemple)

- On veut montrer que $\forall n. (n + 0 = n)$.
- Pour cela, il faut :
 - 1 identifier la propriété P : ici, $P(n)$ est $(n + 0 = n)$ pour tout n ;
 - 2 prouver le cas de base : ici, $P(0)$ est $0 + 0 = 0$ (Base : Zéro)
 - 3 prouver l'hérédité : ici : si $P(n)$, alors $P((Sn))$
 - on suppose : $P(n)$ est $n + 0 = n$ (Hypothèse d'induction)



Preuves par induction (exemple)

- On veut montrer que $\forall n. (n + 0 = n)$.
- Pour cela, il faut :
 - 1 identifier la propriété P : ici, $P(n)$ est $(n + 0 = n)$ pour tout n ;
 - 2 prouver le cas de base : ici, $P(0)$ est $0 + 0 = 0$ (Base : Zéro)
 - 3 prouver l'hérédité : ici : si $P(n)$, alors $P((Sn))$
 - on suppose : $P(n)$ est $n + 0 = n$ (Hypothèse d'induction)
 - on veut déduire : $P((Sn))$ est $(Sn) + 0 = (Sn)$



Preuves par induction (exemple)

- On veut montrer que $\forall n. (n + 0 = n)$.
- Pour cela, il faut :
 - 1 identifier la propriété P : ici, $P(n)$ est $(n + 0 = n)$ pour tout n ;
 - 2 prouver le cas de base : ici, $P(0)$ est $0 + 0 = 0$ (Base : Zéro)
 - 3 prouver l'hérédité : ici : si $P(n)$, alors $P((Sn))$
 - on suppose : $P(n)$ est $n + 0 = n$ (Hypothèse d'induction)
 - on veut déduire : $P((Sn))$ est $(Sn) + 0 = (Sn)$

$$\begin{aligned}(Sn) + 0 &= (S(n + 0)) && \text{(d'après (Successeur +))} \\ &= (Sn) && \text{(d'après hyp. d'induction)}\end{aligned}$$

donc $P((Sn))$



Résumé sur l'induction

- Les **ensembles inductifs**
 - sont générés à partir de *cas de base* ;
 - en appliquant des *règles de construction*.
- Les **fonctions récursives**
 - décomposent une structure inductive composée ;
 - s'arrêtent sur les cas de base.
- Les **preuves par induction** permettent de conclure qu'une propriété est satisfaite pour *tout* élément d'un ensemble inductif
 - si elle est satisfaite pour les cas de base
 - si elle est héréditaire pour les éléments composés



Conclusions

- Au travers de 3 exemples, on sait maintenant donner :
 - la définition inductive d'un ensemble ;
 - la définition récursive d'une fonction ;
 - faire une preuve par induction.
- Dans le prochain chapitre :
 - nous reviendrons sur l'induction en l'appliquant à la logique ;
 - on définira le langage de la logique propositionnelle de manière inductive.



Plan

1 Introduction

2 Logique des propositions

- Syntaxe (langage)
- Sémantique (théorie des modèles)
- Résolution
- Quantification sur ensembles finis
- *Toulouse Integrated Satisfiability Tool (TouIST)*

3 Logique des prédictats



Logique des propositions

Syntaxe (langage)



Énoncé vs proposition (rappel diapo 10)

- L'énoncé
 - « Je suis au bureau ou dans le jardin et je lis un livre » ;
- est composé de 3 propositions :
 - « Je suis au bureau » ;
 - « Je suis dans le jardin » ;
 - « je lis un livre » ;
- reliées par 2 connecteurs :
 - « ou » ;
 - « et ».



La logique des propositions

- C'est une logique très simple où :
 - une **proposition** constituant un énoncé est représentée par une entité non analysable appelée **variable propositionnelle** ;



La logique des propositions

- C'est une logique très simple où :
 - une **proposition** constituant un énoncé est représentée par une entité non analysable appelée **variable propositionnelle** ;
 - un **énoncé** est représenté par une entité constituée de **propositions reliées par des connecteurs** et appelée **formule** ;



La logique des propositions

- C'est une logique très simple où :
 - une **proposition** constituant un énoncé est représentée par une entité non analysable appelée **variable propositionnelle** ;
 - un **énoncé** est représenté par une entité constituée de **propositions reliées par des connecteurs** et appelée **formule** ;
- dans toute la suite, on note :
 - *LProp*, la logique propositionnelle en général ;
 - *PROP*, l'ensemble des variables propositionnelles ;
 - *FORM*, l'ensemble des formules.



Principe de formalisation avec *LProp*

- Cela consiste à associer des **variables propositionnelles** à des **propositions**



Principe de formalisation avec *LProp*

- Cela consiste à associer des **variables propositionnelles** à des **propositions** (par exemple) :
 - n est associé à « il fait nuit » ;
 - l est associé à « la lumière est allumée » ;



Principe de formalisation avec *LProp*

- Cela consiste à associer des **variables propositionnelles** à des **propositions** (par exemple) :
 - n est associé à « il fait nuit » ;
 - l est associé à « la lumière est allumée » ;
- puis les relier par des **connecteurs logiques**



Principe de formalisation avec *LProp*

- Cela consiste à associer des **variables propositionnelles** à des **propositions** (par exemple) :
 - n est associé à « il fait nuit » ;
 - l est associé à « la lumière est allumée » ;
- puis les relier par des **connecteurs logiques** (par exemple) :
 - « Il fait nuit **et** la lumière est allumée » : $n \wedge l$;



Principe de formalisation avec *LProp*

- Cela consiste à associer des **variables propositionnelles** à des **propositions** (par exemple) :
 - n est associé à « il fait nuit » ;
 - l est associé à « la lumière est allumée » ;
- puis les relier par des **connecteurs logiques** (par exemple) :
 - « Il fait nuit **et** la lumière est allumée » : $n \wedge l$;
 - « Il fait nuit **ou** la lumière est allumée » : $n \vee l$;



Principe de formalisation avec *LProp*

- Cela consiste à associer des **variables propositionnelles** à des **propositions** (par exemple) :
 - n est associé à « il fait nuit » ;
 - l est associé à « la lumière est allumée » ;
- puis les relier par des **connecteurs logiques** (par exemple) :
 - « Il fait nuit **et** la lumière est allumée » : $n \wedge l$;
 - « Il fait nuit **ou** la lumière est allumée » : $n \vee l$;
 - « **S'il** fait nuit, **alors** la lumière est allumée » : $n \rightarrow l$;



Principe de formalisation avec *LProp*

- Cela consiste à associer des **variables propositionnelles** à des **propositions** (par exemple) :
 - n est associé à « il fait nuit » ;
 - l est associé à « la lumière est allumée » ;
- puis les relier par des **connecteurs logiques** (par exemple) :
 - « Il fait nuit **et** la lumière est allumée » : $n \wedge l$;
 - « Il fait nuit **ou** la lumière est allumée » : $n \vee l$;
 - « **S'il** fait nuit, **alors** la lumière est allumée » : $n \rightarrow l$;
 - « Il **ne** fait **pas** nuit » : $\neg n$;



Principe de formalisation avec *LProp*

- Cela consiste à associer des **variables propositionnelles** à des **propositions** (par exemple) :
 - n est associé à « il fait nuit » ;
 - l est associé à « la lumière est allumée » ;
- puis les relier par des **connecteurs logiques** (par exemple) :
 - « Il fait nuit **et** la lumière est allumée » : $n \wedge l$;
 - « Il fait nuit **ou** la lumière est allumée » : $n \vee l$;
 - « **S'il** fait nuit, **alors** la lumière est allumée » : $n \rightarrow l$;
 - « Il **ne** fait **pas** nuit » : $\neg n$;
- pour constituer des **formules** associées à des **énoncés**.



Limite d'expressivité de *LProp*

- *LProp* permet **difficilement** d'exprimer :
 - des rapports entre entités en nombre **infini** ou **non borné** :
« Entre 2 nombres, il y en a un 3^e » ➔ logique des prédictats ;



Limite d'expressivité de *LProp*

- *LProp* permet **difficilement** d'exprimer :
 - des rapports entre entités en nombre **infini** ou **non borné** :
« Entre 2 nombres, il y en a un 3^e » ➔ logique des prédictats ;
 - des rapports *temporels* :
« S'il fait nuit, la lumière va s'allumer » ➔ logiques temporelles ;



Limite d'expressivité de *LProp*

- *LProp* permet **difficilement** d'exprimer :
 - des rapports entre entités en nombre **infini** ou **non borné** :
« Entre 2 nombres, il y en a un 3^e » ➔ logique des prédictats ;
 - des rapports *temporels* :
« S'il fait nuit, la lumière va s'allumer » ➔ logiques temporelles ;
 - des souhaits, obligations, possibilités, ... :
« Je pense que s'il fait nuit, la lumière devrait être allumée » ➔ logiques modales.



Limite d'expressivité de *LProp*

- *LProp* permet **difficilement** d'exprimer :
 - des rapports entre entités en nombre **infini** ou **non borné** :
« Entre 2 nombres, il y en a un 3^e » ➔ logique des prédictats ;
 - des rapports *temporels* :
« S'il fait nuit, la lumière va s'allumer » ➔ logiques temporelles ;
 - des souhaits, obligations, possibilités, ... :
« Je pense que s'il fait nuit, la lumière devrait être allumée »
➔ logiques modales.
- et ne permet **pas** d'exprimer certaines nuances :
 - « Il fait nuit **mais** la lumière est allumée » est aussi représenté par $n \wedge l$ (« Il fait nuit **et** la lumière est allumée »).



Non ambiguïté des langages formels

- *LProp*, comme tous les langages formels, est **non ambigu** :
 - « Je suis au bureau **ou** dans le jardin **et** je lis un livre » ;
 - peut être représenté par $(b \vee j) \wedge l$ ou par $b \vee (j \wedge l)$
- ➡ il faut choisir !



Non ambiguïté des langages formels

- *LProp*, comme tous les langages formels, est **non ambigu** :
 - « Je suis au bureau **ou** dans le jardin **et** je lis un livre » ;
 - peut être représenté par $(b \vee j) \wedge l$ ou par $b \vee (j \wedge l)$
► il faut choisir !
- Inversement, une formule peut correspondre à **plusieurs énoncés** :
 - « Je vais à la plage s'il fait beau » et « S'il fait beau, je vais à la plage » ;
 - correspondent tous les 2 à la formule $b \rightarrow p$
où b associé à « il fait beau » et p à « je vais à la plage ».



Vocabulaire de *LProp*

- Des **variables propositionnelles** : $\{p, q, r, \dots\}$ ou des mnémoniques ;



Vocabulaire de *LProp*

- Des **variables propositionnelles** : $\{p, q, r, \dots\}$ ou des mnémoniques ;
- des **constantes propositionnelles** :
 - \perp (« bottom ») associée à « toujours faux » ;
 - \top (« top ») associée à « toujours vrai » ;



Vocabulaire de *LProp*

- Des **variables propositionnelles** : $\{p, q, r, \dots\}$ ou des mnémoniques ;
- des **constantes propositionnelles** :
 - \perp (« bottom ») associée à « toujours faux » ;
 - \top (« top ») associée à « toujours vrai » ;
- des **connecteurs logiques** : $\wedge, \vee, \rightarrow, \neg, \dots$;



Vocabulaire de *LProp*

- Des **variables propositionnelles** : $\{p, q, r, \dots\}$ ou des mnémoniques ;
- des **constantes propositionnelles** :
 - \perp (« bottom ») associée à « toujours faux » ;
 - \top (« top ») associée à « toujours vrai » ;
- des **connecteurs logiques** : $\wedge, \vee, \rightarrow, \neg, \dots$;
- des **parenthèses** (« sucre syntaxique ») : (,).



Vocabulaire de *LProp*

- Des **variables propositionnelles** : $\{p, q, r, \dots\}$ ou des mnémoniques ;
 - des **constantes propositionnelles** :
 - \perp (« bottom ») associée à « toujours faux » ;
 - \top (« top ») associée à « toujours vrai » ;
 - des **connecteurs logiques** : $\wedge, \vee, \rightarrow, \neg, \dots$;
 - des **parenthèses** (« sucre syntaxique ») : $(,)$.
- ! *Les constantes ou les connecteurs ne sont pas toujours définis de manière primitive dans le langage.*



Liste des connecteurs utilisés

- \wedge (**conjonction**) se lit « et » ;
- \vee (**disjonction inclusive**) se lit « ou » ;
- \oplus (**disjonction exclusive**) se lit « ou-exclusif » ou « xor » ;
- \rightarrow (**implication**) se lit « si ... alors » ;
- \leftrightarrow (**équivalence**) se lit « si et seulement si » ou « équivaut » ;
- \neg (**négation**) se lit « non ».



Définition inductive des formules

- $FORM$ est le **plus petit ensemble** satisfaisant les conditions :
 - 1 pour tout $p \in PROP$, $p \in FORM$ (Variable propositionnelle)



Définition inductive des formules

- $FORM$ est le **plus petit ensemble** satisfaisant les conditions :
 - 1 pour tout $p \in PROP$, $p \in FORM$ (Variable propositionnelle)
 - 2 $\perp \in FORM$ (Constante)



Définition inductive des formules

- $FORM$ est le **plus petit ensemble** satisfaisant les conditions :
 - 1 pour tout $p \in PROP$, $p \in FORM$ (Variable propositionnelle)
 - 2 $\perp \in FORM$ (Constante)
 - 3 si $A \in FORM$ alors $(\neg A) \in FORM$ (Négation)



Définition inductive des formules

- $FORM$ est le **plus petit ensemble** satisfaisant les conditions :
 - 1 pour tout $p \in PROP$, $p \in FORM$ (Variable propositionnelle)
 - 2 $\perp \in FORM$ (Constante)
 - 3 si $A \in FORM$ alors $(\neg A) \in FORM$ (Négation)
 - 4 si $A \in FORM$ et $B \in FORM$ alors $(A \wedge B) \in FORM$ (Conjonction)



Définition inductive des formules

- $FORM$ est le **plus petit ensemble** satisfaisant les conditions :
 - 1 pour tout $p \in PROP$, $p \in FORM$ (Variable propositionnelle)
 - 2 $\perp \in FORM$ (Constante)
 - 3 si $A \in FORM$ alors $(\neg A) \in FORM$ (Négation)
 - 4 si $A \in FORM$ et $B \in FORM$ alors $(A \wedge B) \in FORM$ (Conjonction)
 - 5 si $A \in FORM$ et $B \in FORM$ alors $(A \vee B) \in FORM$ (Disjonction)



Définition inductive des formules

- $FORM$ est le **plus petit ensemble** satisfaisant les conditions :
 - 1 pour tout $p \in PROP$, $p \in FORM$ (Variable propositionnelle)
 - 2 $\perp \in FORM$ (Constante)
 - 3 si $A \in FORM$ alors $(\neg A) \in FORM$ (Négation)
 - 4 si $A \in FORM$ et $B \in FORM$ alors $(A \wedge B) \in FORM$ (Conjonction)
 - 5 si $A \in FORM$ et $B \in FORM$ alors $(A \vee B) \in FORM$ (Disjonction)
 - 6 si $A \in FORM$ et $B \in FORM$ alors $(A \rightarrow B) \in FORM$ (Implication)



Définition inductive des formules

- $FORM$ est le **plus petit ensemble** satisfaisant les conditions :
 - 1 pour tout $p \in PROP$, $p \in FORM$ (Variable propositionnelle)
 - 2 $\perp \in FORM$ (Constante)
 - 3 si $A \in FORM$ alors $(\neg A) \in FORM$ (Négation)
 - 4 si $A \in FORM$ et $B \in FORM$ alors $(A \wedge B) \in FORM$ (Conjonction)
 - 5 si $A \in FORM$ et $B \in FORM$ alors $(A \vee B) \in FORM$ (Disjonction)
 - 6 si $A \in FORM$ et $B \in FORM$ alors $(A \rightarrow B) \in FORM$ (Implication)

! Les **variables propositionnelles** (p) sont des propositions de $LProp$.
Mais les **méta-variables** (A, B) **représentent** des formules de $LProp$ mais **n'appartiennent pas** au langage du $LProp$. **Ne pas confondre !**



Notion d'arbre syntaxique

- Les **arbres syntaxiques** servent à représenter des structures syntaxiques (énoncés en langage naturel, formules, etc.);
- ils sont constitués de **nœuds** et de **sous-arbres**, constitués eux-mêmes d'un nœud et de sous-arbres (déf. récursive);
- les **nœuds terminaux** (sans sous-arbre) sont appelés **feuilles**;



Notion d'arbre syntaxique

- Les **arbres syntaxiques** servent à représenter des structures syntaxiques (énoncés en langage naturel, formules, etc.);
- ils sont constitués de **nœuds** et de **sous-arbres**, constitués eux-mêmes d'un nœud et de sous-arbres (déf. récursive);
- les **nœuds terminaux** (sans sous-arbre) sont appelés **feuilles**;
- ici, nous allons utiliser des arbres :
 - dont chaque nœud a au plus 2 sous-arbres;
 - dont les **nœuds** sont des **connecteurs**;
 - dont les **feuilles** sont des **variables propositionnelles**.



Représentation d'une formule avec un arbre syntaxique

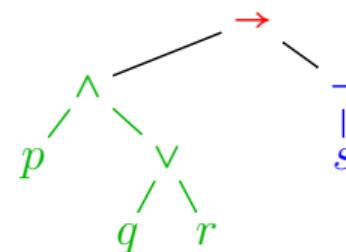
Formule :

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

➡ Passage

« formule \Leftrightarrow arbre » :
voir p. 112 et suiv.

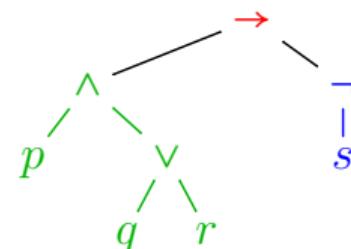
Arbre syntaxique :





Comprendre la définition inductive des formules propositionnelles

- $s \in PROP$, donc $s \in FORM$
 - donc $(\neg s) \in FORM$
- $q \in PROP$, donc $q \in FORM$
- $r \in PROP$, donc $r \in FORM$
 - donc $(q \vee r) \in FORM$
- $p \in PROP$, donc $p \in FORM$
 - donc $(p \wedge (q \vee r)) \in FORM$
- donc $((p \wedge (q \vee r)) \rightarrow (\neg s)) \in FORM$





Exemple de fonction récursive

- Définition de $\text{ndp}(A)$ (nombre de parenthèses de $A \in FORM$) ?



Exemple de fonction récursive

- Définition de $\text{ndp}(A)$ (nombre de parenthèses de $A \in FORM$) ?
 - Variable : $\text{ndp}(p) = 0$ [eq. V]
 - Constante : $\text{ndp}(\perp) = 0$ [eq. Ct]



Exemple de fonction récursive

- Définition de $\text{ndp}(A)$ (nombre de parenthèses de $A \in FORM$) ?
 - *Variable* : $\text{ndp}(p) = 0$ [eq. V]
 - *Constante* : $\text{ndp}(\perp) = 0$ [eq. Ct]
 - *Négation* : $\text{ndp}((\neg A)) = \text{ndp}(A) + 2$ [eq. N]



Exemple de fonction récursive

- Définition de $\text{ndp}(A)$ (nombre de parenthèses de $A \in FORM$) ?
 - *Variable* : $\text{ndp}(p) = 0$ [eq. V]
 - *Constante* : $\text{ndp}(\perp) = 0$ [eq. Ct]
 - *Négation* : $\text{ndp}((\neg A)) = \text{ndp}(A) + 2$ [eq. N]
 - *Conjonction* : $\text{ndp}((A \wedge B)) = \text{ndp}(A) + \text{ndp}(B) + 2$ [eq. C]
 - *Disjonction* : $\text{ndp}((A \vee B)) = \text{ndp}(A) + \text{ndp}(B) + 2$ [eq. D]
 - *Implication* : $\text{ndp}((A \rightarrow B)) = \text{ndp}(A) + \text{ndp}(B) + 2$ [eq. I]



Exemple de fonction récursive

- Définition de $\text{ndp}(A)$ (nombre de parenthèses de $A \in \text{FORM}$) ?
 - *Variable* : $\text{ndp}(p) = 0$ [eq. V]
 - *Constante* : $\text{ndp}(\perp) = 0$ [eq. Ct]
 - *Négation* : $\text{ndp}((\neg A)) = \text{ndp}(A) + 2$ [eq. N]
 - *Conjonction* : $\text{ndp}((A \wedge B)) = \text{ndp}(A) + \text{ndp}(B) + 2$ [eq. C]
 - *Disjonction* : $\text{ndp}((A \vee B)) = \text{ndp}(A) + \text{ndp}(B) + 2$ [eq. D]
 - *Implication* : $\text{ndp}((A \rightarrow B)) = \text{ndp}(A) + \text{ndp}(B) + 2$ [eq. I]
- Exemples :
 - $\text{ndp}(((p \vee q) \wedge r)) = 4$
 - $\text{ndp}(((p \wedge q) \vee (\neg(r \vee \perp)))) = 8$



Schéma de preuve par induction sur les formules (base)

- Base :
 - *Variable* : si pour tout $p \in PROP$, $\mathcal{P}(p)$
 - *Constante* : et si $\mathcal{P}(\perp)$



Schéma de preuve par induction sur les formules (héritéité)

■ Héritéité :

- *Négation* : et si pour tout $A \in FORM$, $\mathcal{P}(A)$ implique $\mathcal{P}((\neg A))$
- *Conjonction* : et si pour tout $A, B \in FORM$,
 $\mathcal{P}(A)$ et $\mathcal{P}(B)$ implique $\mathcal{P}((A \wedge B))$
- *Disjonction* : et si pour tout $A, B \in FORM$,
 $\mathcal{P}(A)$ et $\mathcal{P}(B)$ implique $\mathcal{P}((A \vee B))$
- *Implication* : et si pour tout $A, B \in FORM$,
 $\mathcal{P}(A)$ et $\mathcal{P}(B)$ implique $\mathcal{P}((A \rightarrow B))$



Schéma de preuve par induction sur les formules (conclusion)

- Conclusion :
 - alors : $\forall A \in FORM. \mathcal{P}(A)$



Ex. de preuve par induction sur les formules (propriété)

- On cherche à démontrer que le nombre de parenthèses d'une formule de $LProp$ est toujours pair ;
- soit : $\forall f. \text{ndp}(f) \bmod 2 = 0$;
- ici, la propriété $\mathcal{P}(A)$ est fonction des formules et est $(\text{ndp}(A) \bmod 2 = 0)$.



Ex. de preuve par induction sur les formules (base)

■ Base :

- *Variable* : il faut montrer que $\mathcal{P}(p)$;
or, $\text{ndp}(p) \bmod 2 = 0 \bmod 2 = 0$ (par [eq. V])
- *Constante* : il faut montrer que $\mathcal{P}(\perp)$;
or, $\text{ndp}(\perp) \bmod 2 = 0 \bmod 2 = 0$ (par [eq. C])



Ex. de preuve par induction sur les formules (hérité Négation)

■ Hérité (Négation) :

- il faut montrer que si $\mathcal{P}(A)$, alors $\mathcal{P}((\neg A))$
- hypothèse d'induction : $\text{ndp}(A) \bmod 2 = 0$ [eq. H]
- on veut montrer que $\text{ndp}((\neg A)) \bmod 2 = 0$:

$$\begin{aligned}\text{ndp}((\neg A)) \bmod 2 &= (\text{ndp}(A) + 2) \bmod 2 && \text{(par [eq. N])} \\ &= \text{ndp}(A) \bmod 2 && \text{(arithmétique)} \\ &= 0 && \text{(par [eq. H])}\end{aligned}$$



Ex. de preuve par induction sur les formules (hérité Conjonction)

- Hérité (Conjonction) :
 - il faut montrer que si $\mathcal{P}(A)$ et $\mathcal{P}(B)$ alors $\mathcal{P}((A \wedge B))$
 - Hypothèses d'induction :
 - $\text{ndp}(A) \bmod 2 = 0$ [eq. Ha]
 - $\text{ndp}(B) \bmod 2 = 0$ [eq. Hb]



Ex. de preuve par induction sur les formules (hérité Conjonction)

- Hérité (Conjonction) :

- il faut montrer que si $\mathcal{P}(A)$ et $\mathcal{P}(B)$ alors $\mathcal{P}((A \wedge B))$

- Hypothèses d'induction :

- $\text{ndp}(A) \bmod 2 = 0$ [eq. Ha]

- $\text{ndp}(B) \bmod 2 = 0$ [eq. Hb]

- on veut montrer que $\text{ndp}((A \wedge B)) \bmod 2 = 0$:



Ex. de preuve par induction sur les formules (hérité Conjonction)

- Hérité (Conjonction) :

- il faut montrer que si $\mathcal{P}(A)$ et $\mathcal{P}(B)$ alors $\mathcal{P}((A \wedge B))$

- Hypothèses d'induction :

- $\text{ndp}(A) \bmod 2 = 0$ [eq. Ha]

- $\text{ndp}(B) \bmod 2 = 0$ [eq. Hb]

- on veut montrer que $\text{ndp}((A \wedge B)) \bmod 2 = 0$:

$$\text{ndp}((A \wedge B)) \bmod 2 = (\text{ndp}(A) + \text{ndp}(B) + 2) \bmod 2 \quad (\text{par [eq. C]})$$

$$= (\text{ndp}(A) + \text{ndp}(B)) \bmod 2 \quad (\text{arithmétique})$$

$$= (0 + \text{ndp}(B)) \bmod 2 \quad (\text{par [eq. Ha]})$$

$$= (0 + 0) \bmod 2 \quad (\text{par [eq. Hb]})$$

$$= 0 \quad (\text{arithmétique})$$



Ex. de preuve par induction sur les formules (héritéité autres)

- Héritéité (*Disjonction et Implication*) :
 - Similaire à Héritéité (Conjonction).



Sous-formules d'une formule (définition récursive)

- Soit $p \in PROP$, $F \in FORM$;
- on définit la fonction récursive qui renvoie l'ensemble $\text{sf}(F)$ des sous-formules de F de la manière suivante :
 - $\text{sf}(p) = \{p\}$ [sf.prop]



Sous-formules d'une formule (définition récursive)

- Soit $p \in PROP$, $F \in FORM$;
- on définit la fonction récursive qui renvoie l'ensemble $\text{sf}(F)$ des sous-formules de F de la manière suivante :

- $\text{sf}(p) = \{p\}$ [sf.prop]

- $\text{sf}(\perp) = \{\perp\}$ [sf.bot]



Sous-formules d'une formule (définition récursive)

- Soit $p \in PROP$, $F \in FORM$;
- on définit la fonction récursive qui renvoie l'ensemble $\text{sf}(F)$ des sous-formules de F de la manière suivante :

- $\text{sf}(p) = \{p\}$ [sf.prop]

- $\text{sf}(\perp) = \{\perp\}$ [sf.bot]

- $\text{sf}((\neg A)) = \text{sf}(A) \cup \{(\neg A)\}$ [sf.not]



Sous-formules d'une formule (définition récursive)

- Soit $p \in PROP$, $F \in FORM$;
- on définit la fonction récursive qui renvoie l'ensemble $\text{sf}(F)$ des sous-formules de F de la manière suivante :

- $\text{sf}(p) = \{p\}$ [sf.prop]

- $\text{sf}(\perp) = \{\perp\}$ [sf.bot]

- $\text{sf}(\neg A) = \text{sf}(A) \cup \{\neg A\}$ [sf.not]

- $\text{sf}(A \wedge B) = \text{sf}(A) \cup \text{sf}(B) \cup \{A \wedge B\}$ [sf.and]



Sous-formules d'une formule (définition récursive)

- Soit $p \in PROP$, $F \in FORM$;
- on définit la fonction récursive qui renvoie l'ensemble $\text{sf}(F)$ des sous-formules de F de la manière suivante :

$$\blacksquare \quad \text{sf}(p) = \{p\} \qquad \qquad \qquad [\text{sf.prop}]$$

$$\blacksquare \quad \text{sf}(\perp) = \{\perp\} \qquad \qquad \qquad [\text{sf.bot}]$$

$$\blacksquare \quad \text{sf}(\neg A) = \text{sf}(A) \cup \{\neg A\} \qquad \qquad \qquad [\text{sf.not}]$$

$$\blacksquare \quad \text{sf}(A \wedge B) = \text{sf}(A) \cup \text{sf}(B) \cup \{A \wedge B\} \qquad \qquad \qquad [\text{sf.and}]$$

$$\blacksquare \quad \text{sf}(A \vee B) = \text{sf}(A) \cup \text{sf}(B) \cup \{A \vee B\} \qquad \qquad \qquad [\text{sf.or}]$$



Sous-formules d'une formule (définition récursive)

- Soit $p \in PROP$, $F \in FORM$;
- on définit la fonction récursive qui renvoie l'ensemble $\text{sf}(F)$ des sous-formules de F de la manière suivante :

■ $\text{sf}(p) = \{p\}$ [sf.prop]

■ $\text{sf}(\perp) = \{\perp\}$ [sf.bot]

■ $\text{sf}(\neg A) = \text{sf}(A) \cup \{\neg A\}$ [sf.not]

■ $\text{sf}(A \wedge B) = \text{sf}(A) \cup \text{sf}(B) \cup \{A \wedge B\}$ [sf.and]

■ $\text{sf}(A \vee B) = \text{sf}(A) \cup \text{sf}(B) \cup \{A \vee B\}$ [sf.or]

■ $\text{sf}(A \rightarrow B) = \text{sf}(A) \cup \text{sf}(B) \cup \{A \rightarrow B\}$ [sf.imp]



Sous-formules d'une formule (exemple - 1^{er} niveau)

- Par application de [sf.imp] on a :

$$\begin{aligned} \text{sf}\left(\left((p \wedge (q \vee r)) \rightarrow (\neg s)\right)\right) = & \text{sf}\left((p \wedge (q \vee r))\right) \cup \\ & \text{sf}\left((\neg s)\right) \cup \\ & \left\{\left((p \wedge (q \vee r)) \rightarrow (\neg s)\right)\right\} \end{aligned}$$

- $(p \wedge (q \vee r))$ et $(\neg s)$ sont appelées « *sous-formules immédiates* de $((p \wedge (q \vee r)) \rightarrow (\neg s))$ ».



Sous-formules d'une formule (exemple - 2^e niveau)

- Par application de [sf.and] on a :

$$\text{sf}((p \wedge (q \vee r))) = \text{sf}(p) \cup \text{sf}((q \vee r)) \cup \{(p \wedge (q \vee r))\}$$

- Par application de [sf.not] on a :

$$\text{sf}((\neg s)) = \text{sf}(s) \cup \{(\neg s)\}$$

- Donc :

$$\begin{aligned} \text{sf}(((p \wedge (q \vee r)) \rightarrow (\neg s))) = & \text{sf}(p) \cup \text{sf}((q \vee r)) \cup \{(p \wedge (q \vee r))\} \cup \\ & \text{sf}(s) \cup \{(\neg s)\} \cup \\ & \left\{ ((p \wedge (q \vee r)) \rightarrow (\neg s)) \right\} \end{aligned}$$



Sous-formules d'une formule (exemple - 3^e niveau)

- Par application de [sf.prop] on a :

$$\text{sf}(p) = \{p\} \quad \text{et} \quad \text{sf}(s) = \{s\}$$

- Par application de [sf.or] on a :

$$\text{sf}((q \vee r)) = \text{sf}(q) \cup \text{sf}(r) \cup \{(q \vee r)\}$$

- Donc :

$$\begin{aligned} \text{sf}(((p \wedge (q \vee r)) \rightarrow (\neg s))) &= \{p\} \cup \text{sf}(q) \cup \text{sf}(r) \cup \{(q \vee r), (p \wedge (q \vee r))\} \cup \\ &\quad \{s, (\neg s)\} \cup \\ &\quad \left\{ ((p \wedge (q \vee r)) \rightarrow (\neg s)) \right\} \end{aligned}$$



Sous-formules d'une formule (exemple - 4^e niveau)

- Par application de [sf.prop] on a :

$$\text{sf}(\textcolor{green}{q}) = \{q\} \quad \text{et} \quad \text{sf}(\textcolor{blue}{r}) = \{r\}$$

- Donc :

$$\begin{aligned}\text{sf}\left(\left((p \wedge (q \vee r)) \rightarrow (\neg s)\right)\right) = & \{p, q, r, (q \vee r), (p \wedge (q \vee r))\} \cup \\ & \{s, (\neg s)\} \cup \\ & \left\{\left((p \wedge (q \vee r)) \rightarrow (\neg s)\right)\right\}\end{aligned}$$



Formules et parenthèses

- En principe, chaque connecteur introduit une paire de parenthèses permettant d'identifier les sous-formules sur lesquelles il s'applique.
- Cela permet de distinguer entre $(b \rightarrow (c \vee p))$ et $((b \rightarrow c) \vee p)$.
- On peut omettre certaines d'entre elles en jouant sur la **priorité des connecteurs** de façon non-ambiguë : on doit pouvoir les restituer.



Conventions sur les parenthèses

- Par convention :
 - quand tous les opérateurs sont identiques, ils **associent à droite** :
 - $A \wedge B \wedge C$ est la même chose que $(A \wedge (B \wedge C))$;
 - $A \rightarrow B \rightarrow C$ est la même chose que $(A \rightarrow (B \rightarrow C))$;



Conventions sur les parenthèses

- Par convention :
 - quand tous les opérateurs sont identiques, ils **associent à droite** :
 - $A \wedge B \wedge C$ est la même chose que $(A \wedge (B \wedge C))$;
 - $A \rightarrow B \rightarrow C$ est la même chose que $(A \rightarrow (B \rightarrow C))$;
 - sinon, on définit une **priorité entre opérateurs** : $\neg > \wedge > \vee > \rightarrow$ ($o > o'$ signifie « l'opérateur o est plus prioritaire que l'opérateur o' ») :
 - $A \wedge B \vee C$ est la même chose que $((A \wedge B) \vee C)$
 - $\neg A \vee B$ est la même chose que $((\neg A) \vee B)$
 - $A \wedge B \rightarrow A \vee B$ est la même chose que $((A \wedge B) \rightarrow (A \vee B))$



Parenthésage maximal d'une formule et arbre syntaxique (principe)

- Parcours sous-arbre gauche, Racine, sous-arbre droit de l'arbre de l'arbre ;
- on commence par la feuille la plus à gauche ;
- à chaque étape, on entoure systématiquement la structure lue de parenthèses ;
- les feuilles ne sont pas entourées de parenthèses.

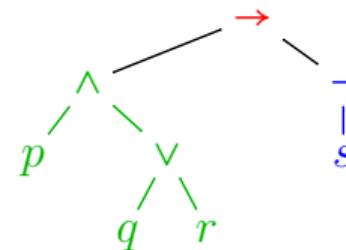


Parenthésage maximal d'une formule et arbre syntaxique (exemple)

Formule :

■ p

Arbre syntaxique :



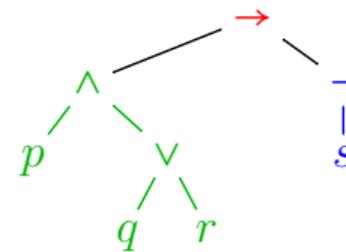


Parenthésage maximal d'une formule et arbre syntaxique (exemple)

Formule :

■ $(p \wedge \dots)$

Arbre syntaxique :



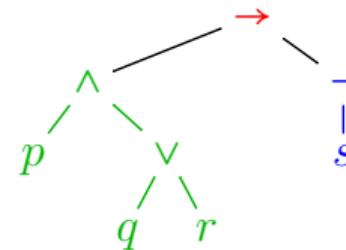


Parenthésage maximal d'une formule et arbre syntaxique (exemple)

Formule :

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

Arbre syntaxique :



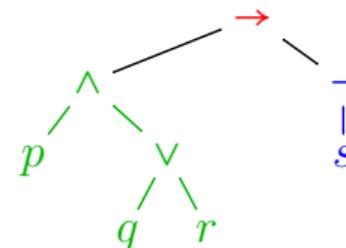


Parenthésage maximal d'une formule et arbre syntaxique (exemple)

Formule :

$$\blacksquare ((p \wedge (q \vee r)) \rightarrow \dots)$$

Arbre syntaxique :



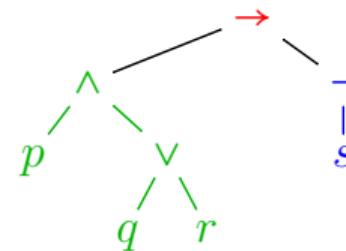


Parenthésage maximal d'une formule et arbre syntaxique (exemple)

Formule :

$$\blacksquare ((p \wedge (q \vee r)) \rightarrow (\neg \dots))$$

Arbre syntaxique :

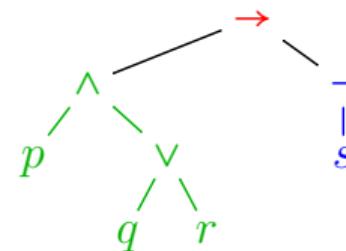




Parenthésage maximal d'une formule et arbre syntaxique (exemple)

Formule :

Arbre syntaxique :



$$\blacksquare ((p \wedge (q \vee r)) \rightarrow (\neg s))$$



Parenthésage minimal d'une formule et arbre syntaxique (principe)

- Parcours sous-arbre gauche, Racine, sous-arbre droit de l'arbre de l'arbre ;
- on commence par la feuille la plus à gauche ;
- à chaque étape, on entoure la structure lue de parenthèses uniquement si son opérateur est moins prioritaire que celui de la structure supérieure ;
- les feuilles ne sont pas entourées de parenthèses.

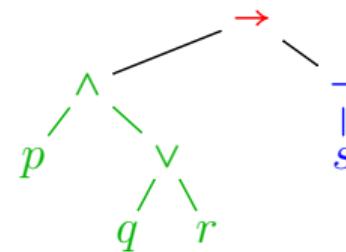


Parenthésage minimal d'une formule et arbre syntaxique (exemple)

Formule :

- p (car p est une feuille)

Arbre syntaxique :



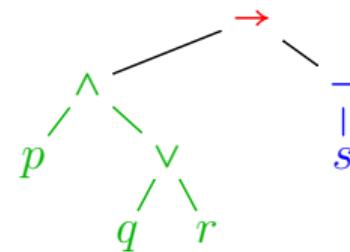


Parenthésage minimal d'une formule et arbre syntaxique (exemple)

Formule :

■ $p \wedge \dots (\text{car} \wedge > \rightarrow)$

Arbre syntaxique :



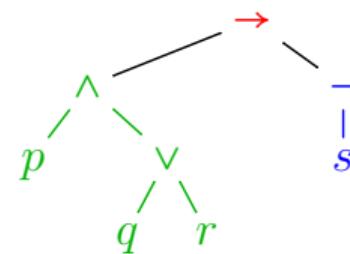


Parenthésage minimal d'une formule et arbre syntaxique (exemple)

Formule :

$$\blacksquare p \wedge (q \vee r) \text{ (car } \vee < \wedge)$$

Arbre syntaxique :



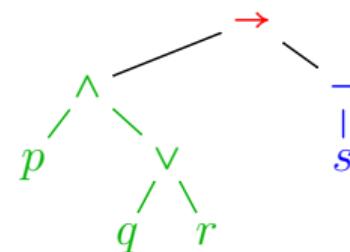


Parenthésage minimal d'une formule et arbre syntaxique (exemple)

Formule :

■ $p \wedge (q \vee r) \rightarrow \dots$ (car $\rightarrow > \emptyset$)

Arbre syntaxique :



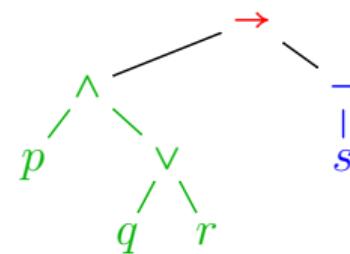


Parenthésage minimal d'une formule et arbre syntaxique (exemple)

Formule :

$$\blacksquare p \wedge (q \vee r) \rightarrow \neg \dots \text{ (car } \neg > \rightarrow)$$

Arbre syntaxique :



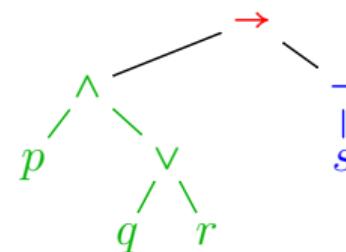


Parenthésage minimal d'une formule et arbre syntaxique (exemple)

Formule :

$$\blacksquare p \wedge (q \vee r) \rightarrow \neg s \text{ (feuille)}$$

Arbre syntaxique :





Logique des propositions

Sémantique (théorie des
modèles)



Sémantique vs syntaxe

- La section précédente : **syntaxe**
 - la **structure** du langage logique ;
 - comment former des formules correctement (règles de construction) à partir d'éléments de base (vocabulaire) ;
 - grec : **syn-taxis**, la « composition » ;



Sémantique vs syntaxe

- La section précédente : **syntaxe**
 - la **structure** du langage logique ;
 - comment former des formules correctement (règles de construction) à partir d'éléments de base (vocabulaire) ;
 - grec : **syn-taxis**, la « composition » ;
- présente section : **sémantique**
 - **signification** du langage logique ;
 - grec : **sema**, le « signe ».



Sémantique vs syntaxe

- La section précédente : **syntaxe**
 - la **structure** du langage logique ;
 - comment former des formules correctement (règles de construction) à partir d'éléments de base (vocabulaire) ;
 - grec : **syn-taxis**, la « composition » ;
- présente section : **sémantique**
 - **signification** du langage logique ;
 - grec : **sema**, le « signe ».
- ▶ cette dichotomie existe dans tout langage (formel ou naturel).



Sémantique et logique

- Donner une sémantique à une formule
- c'est répondre à la question :
 - quand est-ce qu'une formule est **vraie** ou **fausse** ?



Sémantique et logique

- Donner une sémantique à une formule
- c'est répondre à la question :
 - quand est-ce qu'une formule est **vraie** ou **fausse** ?
- on représente souvent :
 - le **vrai** par la valeur **1** ;
 - le **faux** par la valeur **0**.



La vérité, une valeur toute relative

- En général, la vérité d'un **énoncé** est relative au **contexte d'énonciation** :
 - la vérité de « Il est grand pour son âge » ;
 - dépend de l'individu désigné par « il » et de la valeur de « son âge ».



La vérité, une valeur toute relative

- En général, la vérité d'un énoncé est relative au contexte d'énonciation :
 - la vérité de « Il est grand pour son âge » ;
 - dépend de l'individu désigné par « il » et de la valeur de « son âge ».
- De même, la vérité de $x^2 + 1 = y$:
 - dépend des valeurs attribuées à x et à y .



La vérité, une valeur toute relative

- En général, la vérité d'un **énoncé** est relative au **contexte d'énonciation** :
 - la vérité de « Il est grand pour son âge » ;
 - dépend de l'individu désigné par « il » et de la valeur de « son âge ».
- De même, la vérité de $x^2 + 1 = y$:
 - dépend des valeurs attribuées à x et à y .
- Pareillement, la vérité d'une **formule logique** :
 - dépend des valeurs de vérité attribuées à ses **variables propositionnelles**.



Valuation

- Décrit quelle **valeur de vérité** est associée à chaque **variable propositionnelle**.



Valuation

- Décrit quelle **valeur de vérité** est associée à chaque **variable propositionnelle**.
- Formellement :
 - c'est une fonction $v : PROP \rightarrow \{0, 1\}$;
 - avec $v(p) = 0$ pour « p faux » ;
 - et avec $v(p) = 1$ pour « p est vrai ».

(Rappel : $PROP$ est l'ensemble de propositions.)



Extension aux formules

- Étant donnée une valuation,
- on peut calculer la valeur de vérité de la combinaison entre une ou deux variables propositionnelles et un opérateur grâce à **la table de vérité** de cet opérateur,
- puis la nouvelle valeur de vérité obtenue peut à son tour être combinée à d'autres valeurs et opérateurs, etc.
- pour obtenir au final la valeur de vérité d'un formule.



Présentation d'une table de vérité

- chaque ligne correspond à une valuation ;

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

pour $p, q \in PROP$



Présentation d'une table de vérité

- chaque ligne correspond à une valuation ;
- la table de vérité recense toutes les valuations possibles ;

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

pour $p, q \in PROP$



Présentation d'une table de vérité

- chaque ligne correspond à une valuation ;
- la table de vérité recense toutes les valuations possibles ;
- pour n variables apparaissant dans une formule, il y a 2^n valuations possibles (ici, $n = 2$ donc 4 valuations possibles) ;

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

pour $p, q \in PROP$



Présentation d'une table de vérité

- chaque ligne correspond à une valuation ;
- la table de vérité recense toutes les valuations possibles ;
- pour n variables apparaissant dans une formule, il y a 2^n valuations possibles (ici, $n = 2$ donc 4 valuations possibles) ;
- la 3^e ligne se lit : « pour $v(p) = 1$ et $v(q) = 0$ la formule $p \wedge q$ est fausse ».

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

pour $p, q \in PROP$



Tables de vérité des principaux connecteurs (version 1)

p	q	$p \wedge q$	$p \vee q$	$p \leftrightarrow q$	$p \oplus q$	$p \rightarrow q$
0	0	0	0	1	0	?
0	1	0	1	0	1	?
1	0	0	1	0	1	?
1	1	1	1	1	0	?

Discussion

Quelle table de vérité pour \rightarrow ?

► étude de deux exemples pour tenter de répondre à cette question.



Tables de vérité : exemple du bar (problème)

- Dans un bar :
 - il est interdit de servir de l'alcool à un mineur ;
 - autrement dit : si le barman sert de l'alcool à un client, alors celui-ci est nécessairement majeur ;
 - ce qui se formalise par *alcool → majeur*.



Tables de vérité : exemple du bar (problème)

- Dans un bar :
 - il est interdit de servir de l'alcool à un mineur ;
 - autrement dit : si le barman sert de l'alcool à un client, alors celui-ci est nécessairement majeur ;
 - ce qui se formalise par *alcool → majeur*.
- Vous voyez les commandes suivantes : sur une face il y a la boisson commandée et sur l'autre l'âge du client :

Bière	Café	21 ans	13 ans
-------	------	--------	--------

- Lesquelles faut-il retourner pour vérifier que ce sont des commandes autorisées ?



Tables de vérité : exemple du bar (analyse)

- Quels sont les contextes qui enfreignent la loi ?
- autrement dit, pour quelles valuations l'implication $alcool \rightarrow majeur$ est-elle fausse ?

$alcool$	$majeur$	$alcool \rightarrow majeur$
0	0	?
0	1	?
1	0	?
1	1	?



Tables de vérité : exemple du bar (discussion)

- Pour quelle(s) valuation(s) $alcool \rightarrow majeur$ est-elle fausse ?

$alcool$	$majeur$	$alcool \rightarrow majeur$
0	0	?
0	1	?
1	0	0
1	1	?

- On voit que :
 - si $v(alcool) = 0$, la règle ne peut pas avoir été enfreinte ;
 - si $v(majeur) = 1$, non plus ;
 - le seul cas où la règle est falsifiée est quand :
 $v(alcool) = 1$ ET que $v(majeur) = 0$ (sinon, elle est vérifiée)



Tables de vérité : exemple des cartes (problème)

- 4 cartes ont une **lettre** sur une face et un **nombre** sur l'autre ;
- on cherche à vérifier la règle « si une carte a une voyelle sur une face, l'autre doit contenir un nombre pair » ;
- Vous voyez les cartes suivantes : lesquelles faut-il retourner pour vérifier la règle ?

A Z 4 3



Tables de vérité : exemple des cartes (analyse)

- La règle est de la forme : $voyelle \rightarrow pair$;
- les cartes qu'on a besoin de retourner sont celles susceptibles de falsifier la règle ;
- puisque A est une voyelle :
 - il faut vérifier que derrière il y a un nombre pair ;
 - donc, on ne veut pas du cas où $v(voyelle) = 1$ et $v(pair) = 0$;
- donc il faut également vérifier que derrière la carte 3 ($v(pair) = 0$) il n'y a **pas** une voyelle ($v(voyelle) = 0$).

A Z 4 3



Tables de vérité pour \rightarrow et \neg

p	q	$p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1



Tables de vérité pour \rightarrow et \neg

p	q	$p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

p	$\neg p$
0	1
1	0



Interprétation vs valuation

- On a dit (Diapo. 122) que, par exemple :
 - pour $v(p) = 1$ et $v(q) = 0$ « la formule $p \wedge q$ est fausse » ;
- la valeur de vérité de $p \wedge q$:
 - est donné par la table de vérité de la conjonction ;
 - et s'appelle l'**interprétation** de la formule $p \wedge q$;
 - qu'on note à l'aide de la fonction $I_v : FORM \rightarrow \{0, 1\}$;
 - où l'indice v de la fonction indique que celle-ci dépend de la valuation v .



Interprétation d'une formule (définition informelle)

- Une (fonction d')interprétation $I_v : FORM \rightarrow \{0, 1\}$
 - est une généralisation de la valuation $v : PROP \rightarrow \{0, 1\}$;
 - qui affecte une valeur de vérité à toute formule (et pas seulement aux variables propositionnelles).
- MAIS :
 - toute variable propositionnelle étant une formule (voir Diapo. 92) ;
 - on doit imposer que $I_v(p) = v(p)$ pour tout $p \in PROP$.



Tables de vérité des principaux connecteurs (version finale)

A	B	$A \wedge B$	$A \vee B$	$A \leftrightarrow B$	$A \oplus B$	$A \rightarrow B$	$\neg A$
0	0	0	0	1	0	1	1
0	1	0	1	0	1	1	
1	0	0	1	0	1	0	0
1	1	1	1	1	0	1	

- Les tables peuvent donc être généralisées à toute formule ;
- par exemple pour la 3^e ligne :
 - si $I_v(A) = 1$ et $I_v(B) = 0$ alors $I_v(A \vee B) = 1$;
 - exemple : si $I_v(p \leftrightarrow q \wedge r) = 1$ et $I_v(p \wedge \neg r) = 0$ alors $I_v((p \leftrightarrow q \wedge r) \vee (p \wedge \neg r)) = 1$;



Exemple d'interprétation d'une formule à l'aide des tables de vérité

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1						
v_2						
v_3						
v_4						
v_5						
v_6						
v_7						
v_8						



Exemple d'interprétation d'une formule à l'aide des tables de vérité

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1	0					
v_2	0					
v_3	0					
v_4	0					
v_5	1					
v_6	1					
v_7	1					
v_8	1					

- Ex. : pour $v_5(p) = 1$



Exemple d'interprétation d'une formule à l'aide des tables de vérité

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1	0	0				
v_2	0	0				
v_3	0	1				
v_4	0	1				
v_5	1	0				
v_6	1	0				
v_7	1	1				
v_8	1	1				

- Ex. : pour $v_5(p) = 1$, $v_5(q) = 0$



Exemple d'interprétation d'une formule à l'aide des tables de vérité

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1	0	0	0			
v_2	0	0	1			
v_3	0	1	0			
v_4	0	1	1			
v_5	1	0	0			
v_6	1	0	1			
v_7	1	1	0			
v_8	1	1	1			

- Ex. : pour $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$



Exemple d'interprétation d'une formule à l'aide des tables de vérité

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1	0	0	0	0		
v_2	0	0	1	0		
v_3	0	1	0	0		
v_4	0	1	1	0		
v_5	1	0	0	0		
v_6	1	0	1	0		
v_7	1	1	0	1		
v_8	1	1	1	1		

- Ex. : pour $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$, $I_{v_5}(p \wedge q) = 0$



Exemple d'interprétation d'une formule à l'aide des tables de vérité

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1	0	0	0	0	1	
v_2	0	0	1	0	0	
v_3	0	1	0	0	1	
v_4	0	1	1	0	0	
v_5	1	0	0	0	1	
v_6	1	0	1	0	0	
v_7	1	1	0	1	1	
v_8	1	1	1	1	0	

- Ex. : pour $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$, $I_{v_5}(p \wedge q) = 0$ et $I_{v_5}(\neg r) = 1$



Exemple d'interprétation d'une formule à l'aide des tables de vérité

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1	0	0	0	0	1	1
v_2	0	0	1	0	0	0
v_3	0	1	0	0	1	1
v_4	0	1	1	0	0	0
v_5	1	0	0	0	1	1
v_6	1	0	1	0	0	0
v_7	1	1	0	1	1	1
v_8	1	1	1	1	0	1

- Ex. : pour $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$, $I_{v_5}(p \wedge q) = 0$ et $I_{v_5}(\neg r) = 1$, donc $I_{v_5}(p \wedge q \vee \neg r) = 1$.



Interprétation d'une formule : définition formelle

- L'interprétation I_v d'une formule se définit récursivement à partir d'une valuation v comme suit :

- $I_v(p) = v(p);$

- $I_v(\perp) = 0;$

- $I_v(\top) = 1;$



Interprétation d'une formule : définition formelle

- L'interprétation I_v d'une formule se définit récursivement à partir d'une valuation v comme suit :
 - $I_v(p) = v(p)$;
 - $I_v(\perp) = 0$;
 - $I_v(\top) = 1$;
 - $I_v(\neg A) = 1 - I_v(A)$;



Interprétation d'une formule : définition formelle

- L'interprétation I_v d'une formule se définit récursivement à partir d'une valuation v comme suit :
 - $I_v(p) = v(p)$;
 - $I_v(\perp) = 0$;
 - $I_v(\top) = 1$;
 - $I_v(\neg A) = 1 - I_v(A)$;
 - $I_v(A \wedge B) = \min(I_v(A), I_v(B))$;



Interprétation d'une formule : définition formelle

- L'interprétation I_v d'une formule se définit récursivement à partir d'une valuation v comme suit :
 - $I_v(p) = v(p)$;
 - $I_v(\perp) = 0$;
 - $I_v(\top) = 1$;
 - $I_v(\neg A) = 1 - I_v(A)$;
 - $I_v(A \wedge B) = \min(I_v(A), I_v(B))$;
 - $I_v(A \vee B) = \max(I_v(A), I_v(B))$;



Interprétation d'une formule : définition formelle

- L'interprétation I_v d'une formule se définit récursivement à partir d'une valuation v comme suit :

- $I_v(p) = v(p)$;
- $I_v(\perp) = 0$;
- $I_v(\top) = 1$;
- $I_v(\neg A) = 1 - I_v(A)$;
- $I_v(A \wedge B) = \min(I_v(A), I_v(B))$;
- $I_v(A \vee B) = \max(I_v(A), I_v(B))$;
- $I_v(A \rightarrow B) = \max(1 - I_v(A), I_v(B))$. (on y reviendra plus tard)



Exemple d'interprétation d'une formule à l'aide d'une fonction d'interprétation

- Exemple Diapo. 133 avec $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$:

$$I_{v_5}(p \wedge q \vee \neg r) = \max(I_{v_5}(p \wedge q), I_{v_5}(\neg r))$$



Exemple d'interprétation d'une formule à l'aide d'une fonction d'interprétation

- Exemple Diapo. 133 avec $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$:

$$\begin{aligned} I_{v_5}(p \wedge q \vee \neg r) &= \max(I_{v_5}(p \wedge q), I_{v_5}(\neg r)) \\ &= \max(\min(I_{v_5}(p), I_{v_5}(q)), 1 - I_{v_5}(r)) \end{aligned}$$



Exemple d'interprétation d'une formule à l'aide d'une fonction d'interprétation

- Exemple Diapo. 133 avec $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$:

$$\begin{aligned} I_{v_5}(p \wedge q \vee \neg r) &= \max(I_{v_5}(p \wedge q), I_{v_5}(\neg r)) \\ &= \max(\min(I_{v_5}(p), I_{v_5}(q)), 1 - I_{v_5}(r)) \\ &= \max(\min(v_5(p), v_5(q)), 1 - v_5(r)) \end{aligned}$$



Exemple d'interprétation d'une formule à l'aide d'une fonction d'interprétation

- Exemple Diapo. 133 avec $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$:

$$\begin{aligned}I_{v_5}(p \wedge q \vee \neg r) &= \max(I_{v_5}(p \wedge q), I_{v_5}(\neg r)) \\&= \max(\min(I_{v_5}(p), I_{v_5}(q)), 1 - I_{v_5}(r)) \\&= \max(\min(v_5(p), v_5(q)), 1 - v_5(r)) \\&= \max(\min(1, 0), 1 - 0)\end{aligned}$$



Exemple d'interprétation d'une formule à l'aide d'une fonction d'interprétation

- Exemple Diapo. 133 avec $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$:

$$\begin{aligned}I_{v_5}(p \wedge q \vee \neg r) &= \max(I_{v_5}(p \wedge q), I_{v_5}(\neg r)) \\&= \max(\min(I_{v_5}(p), I_{v_5}(q)), 1 - I_{v_5}(r)) \\&= \max(\min(v_5(p), v_5(q)), 1 - v_5(r)) \\&= \max(\min(1, 0), 1 - 0) \\&= \max(0, 1)\end{aligned}$$



Exemple d'interprétation d'une formule à l'aide d'une fonction d'interprétation

- Exemple Diapo. 133 avec $v_5(p) = 1$, $v_5(q) = 0$ et $v_5(r) = 0$:

$$\begin{aligned}I_{v_5}(p \wedge q \vee \neg r) &= \max(I_{v_5}(p \wedge q), I_{v_5}(\neg r)) \\&= \max(\min(I_{v_5}(p), I_{v_5}(q)), 1 - I_{v_5}(r)) \\&= \max(\min(v_5(p), v_5(q)), 1 - v_5(r)) \\&= \max(\min(1, 0), 1 - 0) \\&= \max(0, 1) \\&= 1\end{aligned}$$



Modèle et contre-modèle d'une formule

- Soient v une valuation et I_v l'interprétation associée ;
- On dit que v est :
 - un **modèle** d'une formule A ssi $I_v(A) = 1$;
On dit alors que « v satisfait A » ;



Modèle et contre-modèle d'une formule

- Soient v une valuation et I_v l'interprétation associée ;
- On dit que v est :
 - un **modèle** d'une formule A ssi $I_v(A) = 1$;
On dit alors que « v satisfait A » ;
 - un **contre-modèle** d'une formule A ssi $I_v(A) = 0$;
On dit alors que « v falsifie A » ;



Modèle et contre-modèle d'une formule

- Soient v une valuation et I_v l'interprétation associée ;
- On dit que v est :
 - un **modèle** d'une formule A ssi $I_v(A) = 1$;
On dit alors que « v satisfait A » ;
 - un **contre-modèle** d'une formule A ssi $I_v(A) = 0$;
On dit alors que « v falsifie A » ;
- Exemple : la valuation v telle que $v(p) = 0$ et $v(q) = 1$ est
 - un modèle de $p \vee q$ (car $I_v(p \vee q) = 1$) ;
 - un contre-modèle de $p \wedge q$ (car $I_v(p \wedge q) = 0$).



Validité et (in)satisfiabilité (définition)

- Une formule A est :
 - valide ssi **pour toute** valuation v , $I_v(A) = 1$
($\Rightarrow A$ n'a **aucun** contre-modèle);



Validité et (in)satisfiabilité (définition)

- Une formule A est :
 - valide ssi **pour toute** valuation v , $I_v(A) = 1$
($\Rightarrow A$ n'a **aucun** contre-modèle);
 - insatisfiable ssi **pour toute** valuation v , $I_v(A) = 0$
($\Rightarrow A$ n'a **aucun** modèle);

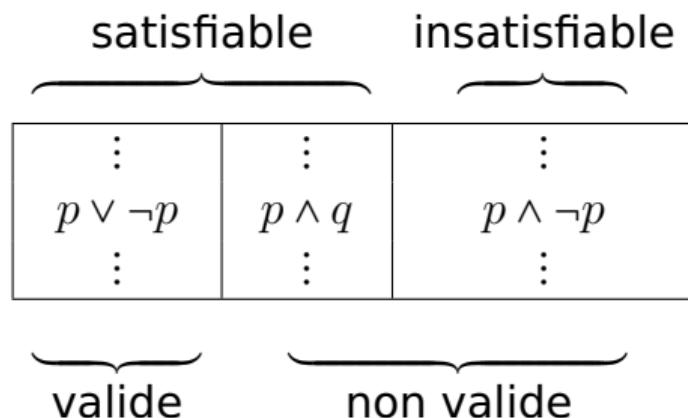


Validité et (in)satisfiabilité (définition)

- Une formule A est :
 - valide ssi **pour toute** valuation v , $I_v(A) = 1$
($\Rightarrow A$ n'a **aucun** contre-modèle);
 - insatisfiable ssi **pour toute** valuation v , $I_v(A) = 0$
($\Rightarrow A$ n'a **aucun** modèle);
 - satisfiable ssi **il existe (au moins) une** valuation v , $I_v(A) = 1$
($\Rightarrow A$ a **au moins un** modèle).



Validité et (in)satisfiabilité (lien entre les notions)





Validité et (in)satisfiabilité (exemple)

p	q	r	$p \wedge q \rightarrow q \vee r$	$(p \vee q) \wedge \neg p \wedge \neg q$	$(p \wedge q) \vee (\neg r)$
0	0	0	1	0	
0	0	1	1	0	
0	1	0	1	0	
0	1	1	1	0	
1	0	0	1	0	
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	0	

- Toutes les valuations de p , q et r sont des modèles de la 1^{re} formule et des contre-modèles de la 2^{nde}.



Validité et (in)satisfiabilité (exemple)

p	q	r	$p \wedge q \rightarrow q \vee r$	$(p \vee q) \wedge \neg p \wedge \neg q$	$(p \wedge q) \vee (\neg r)$
0	0	0	1	0	1
0	0	1	1	0	
0	1	0	1	0	1
0	1	1	1	0	
1	0	0	1	0	1
1	0	1	1	0	
1	1	0	1	0	1
1	1	1	1	0	1

- Toutes les valuation de p , q et r en orange sont des modèles de la 3^e formule.



Validité et (in)satisfiabilité (exemple)

p	q	r	$p \wedge q \rightarrow q \vee r$	$(p \vee q) \wedge \neg p \wedge \neg q$	$(p \wedge q) \vee (\neg r)$
0	0	0	1	0	1
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	0	1

- Toutes les valuation de p , q et r en rouge sont des contre-modèles de la 3^e formule.



Validité et (in)satisfiabilité (remarque 1)

- si A est valide, que sait-on de $\neg A$?



Validité et (in)satisfiabilité (remarque 1)

- si A est valide, que sait-on de $\neg A$?
 - si A est valide, toute valuation est un modèle de A ;



Validité et (in)satisfiabilité (remarque 1)

- si A est valide, que sait-on de $\neg A$?
 - si A est valide, toute valuation est un modèle de A ;
 - donc toute valuation est un contre-modèle de $\neg A$;



Validité et (in)satisfiabilité (remarque 1)

- si A est valide, que sait-on de $\neg A$?
 - si A est valide, toute valuation est un modèle de A ;
 - donc toute valuation est un contre-modèle de $\neg A$;
 - donc $\neg A$ est insatisfiable;



Validité et (in)satisfiabilité (remarque 2)

- si A est satisfiable, $\neg A$ est-il insatisfiable ?



Validité et (in)satisfiabilité (remarque 2)

- si A est satisfiable, $\neg A$ est-il insatisfiable ?
 - A est satisfiable ssi il existe au moins une valuation v modèle de A ;



Validité et (in)satisfiabilité (remarque 2)

- si A est satisfiable, $\neg A$ est-il insatisfiable ?
 - A est satisfiable ssi il existe au moins une valuation v modèle de A ;
 - donc il est possible (mais on n'en sait rien !) qu'il existe une valuation v' qui soit un contre-modèle de A ;



Validité et (in)satisfiabilité (remarque 2)

- si A est satisfiable, $\neg A$ est-il insatisfiable ?
 - A est satisfiable ssi il existe au moins une valuation v modèle de A ;
 - donc il est possible (mais on n'en sait rien !) qu'il existe une valuation v' qui soit un contre-modèle de A ;
 - donc que v' soit un modèle de $\neg A$, auquel cas A n'est pas insatisfiable ;



Validité et (in)satisfiabilité (remarque 2)

- si A est satisfiable, $\neg A$ est-il insatisfiable ?
 - A est satisfiable ssi il existe au moins une valuation v modèle de A ;
 - donc il est possible (mais on n'en sait rien !) qu'il existe une valuation v' qui soit un contre-modèle de A ;
 - donc que v' soit un modèle de $\neg A$, auquel cas A n'est pas insatisfiable ;
 - mais comme on ne sait pas si v' existe ou non, on ne peut rien conclure sur l'insatisfiabilité de $\neg A$.



Modèle et contre-modèle d'un ensemble de formules

■ Soit :

- \mathcal{H} un *ensemble* de formules ($\mathcal{H} \subseteq FORM$) ;
- v une valuation et I_v l'interprétation associée.



Modèle et contre-modèle d'un ensemble de formules

- Soit :
 - \mathcal{H} un *ensemble* de formules ($\mathcal{H} \subseteq FORM$) ;
 - v une valuation et I_v l'interprétation associée.
- v est un **modèle** de \mathcal{H} ssi $I_v(A) = 1$ **pour toute** formule A de \mathcal{H}
 - et on note : $I_v(\mathcal{H}) = 1$
(extension de la fonction I_v aux ensembles de formules) ;



Modèle et contre-modèle d'un ensemble de formules

- Soit :
 - \mathcal{H} un *ensemble* de formules ($\mathcal{H} \subseteq FORM$) ;
 - v une valuation et I_v l'interprétation associée.
- v est un **modèle** de \mathcal{H} ssi $I_v(A) = 1$ **pour toute** formule A de \mathcal{H}
 - et on note : $I_v(\mathcal{H}) = 1$
(extension de la fonction I_v aux ensembles de formules) ;
- v est un **contre-modèle** de \mathcal{H} ssi $I_v(A) = 0$ **pour au moins une** formule A de \mathcal{H}
 - et on note : $I_v(\mathcal{H}) = 0$.



Modèle et contre-modèle d'un ensemble de formules (exemples)

- $v(p) = 0, v(q) = 1$ est un modèle de $\mathcal{H} = \{p \vee q, q\}$
 - car $I_v(p \vee q) = I_v(q) = 1$
(v est un modèle de toutes les formules de \mathcal{H});
 - donc on écrit aussi que $I_v(\mathcal{H}) = 1$;



Modèle et contre-modèle d'un ensemble de formules (exemples)

- $v(p) = 0, v(q) = 1$ est un modèle de $\mathcal{H} = \{p \vee q, q\}$
 - car $I_v(p \vee q) = I_v(q) = 1$
(v est un modèle de toutes les formules de \mathcal{H});
 - donc on écrit aussi que $I_v(\mathcal{H}) = 1$;
- $v(p) = 0, v(q) = 1$ est un contre-modèle de $\mathcal{H}' = \{p \vee q, p\}$
 - car $I_v(p) = 0$
(v est un contre-modèle d'au moins une formule de \mathcal{H}');
 - donc on écrit aussi que $I_v(\mathcal{H}') = 0$.



Satisfiabilité d'un ensemble de formules (exemple)

- L'ensemble $\mathcal{H} = \{\neg p \rightarrow m, m \rightarrow g, \neg g\}$ est-il satisfiable ?
- autrement dit, existe-t-il $v : I_v(A) = 1$ pour tout A de \mathcal{H} ?



Satisfiabilité d'un ensemble de formules (exemple)

- L'ensemble $\mathcal{H} = \{\neg p \rightarrow m, m \rightarrow g, \neg g\}$ est-il satisfiable ?
- autrement dit, existe-t-il $v : I_v(A) = 1$ pour tout A de \mathcal{H} ?
 - il faut $I_v(\neg g) = 1$, donc $v(g) = 0$;



Satisfiabilité d'un ensemble de formules (exemple)

- L'ensemble $\mathcal{H} = \{\neg p \rightarrow m, m \rightarrow g, \neg g\}$ est-il satisfiable ?
- autrement dit, existe-t-il $v : I_v(A) = 1$ pour tout A de \mathcal{H} ?
 - il faut $I_v(\neg g) = 1$, donc $v(g) = 0$;
 - il faut $I_v(m \rightarrow g) = 1$, donc $v(m) = 0$;



Satisfiabilité d'un ensemble de formules (exemple)

- L'ensemble $\mathcal{H} = \{\neg p \rightarrow m, m \rightarrow g, \neg g\}$ est-il satisfiable ?
- autrement dit, existe-t-il $v : I_v(A) = 1$ pour tout A de \mathcal{H} ?
 - il faut $I_v(\neg g) = 1$, donc $v(g) = 0$;
 - il faut $I_v(m \rightarrow g) = 1$, donc $v(m) = 0$;
 - il faut $I_v(\neg p \rightarrow m) = 1$, donc $I_v(\neg p) = 0$, donc $v(p) = 1$.



Satisfiabilité d'un ensemble de formules (exemple)

- L'ensemble $\mathcal{H} = \{\neg p \rightarrow m, m \rightarrow g, \neg g\}$ est-il satisfiable ?
- autrement dit, existe-t-il $v : I_v(A) = 1$ pour tout A de \mathcal{H} ?
 - il faut $I_v(\neg g) = 1$, donc $v(g) = 0$;
 - il faut $I_v(m \rightarrow g) = 1$, donc $v(m) = 0$;
 - il faut $I_v(\neg p \rightarrow m) = 1$, donc $I_v(\neg p) = 0$, donc $v(p) = 1$.
- il existe une valuation v modèle de \mathcal{H} , donc \mathcal{H} est satisfiable.



La conséquence logique (représentation, signification)

- La **conséquence logique** définit précisément ce que signifie : « tirer une **conclusion** à partir d'un ensemble **d'hypothèses** » ;



La conséquence logique (représentation, signification)

- La **conséquence logique** définit précisément ce que signifie : « tirer une **conclusion** à partir d'un ensemble **d'hypothèses** » ;
- elle est représentée par le (méta-)symbole \models ;
- pour $\mathcal{H} \subseteq FORM$ et $C \in FORM$, $\mathcal{H} \models C$ se lit :
 - « C est conséquence (logique) de \mathcal{H} » ;
 - ou encore : « \mathcal{H} a pour conséquence (logique) C » ;



La conséquence logique (représentation, signification)

- La **conséquence logique** définit précisément ce que signifie : « tirer une **conclusion** à partir d'un ensemble **d'hypothèses** » ;
- elle est représentée par le (méta-)symbole \models ;
- pour $\mathcal{H} \subseteq FORM$ et $C \in FORM$, $\mathcal{H} \models C$ se lit :
 - « C est conséquence (logique) de \mathcal{H} » ;
 - ou encore : « \mathcal{H} a pour conséquence (logique) C » ;
- \mathcal{H} s'appelle l'**ensemble des hypothèses** ;
- et C s'appelle la **conclusion**.



La conséquence logique (signification sur un exemple)

- Soit :
 - f représentant « La fonction f est monotone » ;
 - g représentant « La fonction g est monotone » ;
 - m représente « La fonction $f \circ g$ est monotone » ;
- $\{f, g\} \models m$ signifie que :
 - chaque fois que f et g sont des fonctions monotones
 - leur composition est une fonction monotone.



La conséquence logique (sémantique)

- Pour $\mathcal{H} \subseteq FORM$, $C \in FORM$,

$\mathcal{H} \models C$ ssi **tout modèle** de \mathcal{H} est un **modèle** de C .



La conséquence logique (sémantique)

- Pour $\mathcal{H} \subseteq FORM$, $C \in FORM$,

$\mathcal{H} \models C$ ssi tout modèle de \mathcal{H} est un modèle de C .

- Autrement dit, on ne s'intéresse qu'aux valuations qui sont des modèles de \mathcal{H} (cf. Diapo 142) :
 - si ces valuations sont toutes également des modèles de C , alors C est bien conséquence de \mathcal{H} ;



La conséquence logique (sémantique)

- Pour $\mathcal{H} \subseteq FORM$, $C \in FORM$,

$\mathcal{H} \models C$ ssi tout modèle de \mathcal{H} est un modèle de C .

- Autrement dit, on ne s'intéresse qu'aux valuations qui sont des modèles de \mathcal{H} (cf. Diapo 142) :
 - si ces valuations sont toutes également des modèles de C , alors C est bien conséquence de \mathcal{H} ;
 - si au moins un valuation qui est un modèle de \mathcal{H} n'est pas un modèle de C , alors C n'est pas conséquence de \mathcal{H}
 - ▶ et on note : $\mathcal{H} \not\models C$.



La conséquence logique (cas particuliers)

- Dans $\mathcal{H} \models C$, si \mathcal{H} est insatisfiable :
 - alors il n'y a (trivialement) aucun modèle de \mathcal{H} qui n'est pas modèle de C ;
 - donc $\mathcal{H} \models C$ est vrai.



La conséquence logique (cas particuliers)

- Dans $\mathcal{H} \models C$, si \mathcal{H} est insatisfiable :
 - alors il n'y a (trivialement) aucun modèle de \mathcal{H} qui n'est pas modèle de C ;
 - donc $\mathcal{H} \models C$ est vrai.
- Dans $\{\} \models C$:
 - il n'y a (trivialement) aucun modèle de \mathcal{H} qui n'est pas modèle de C ;
 - donc $\{\} \models C$ est vrai ;
 - ce qu'on note $\models C$ et se lit « C est valide ».



La conséquence logique (exemples)

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1	0	0	0	0	1	1
v_2	0	0	1	0	0	0
v_3	0	1	0	0	1	1
v_4	0	1	1	0	0	0
v_5	1	0	0	0	1	1
v_6	1	0	1	0	0	0
v_7	1	1	0	1	1	1
v_8	1	1	1	1	0	1

- $\{p, \neg r\} \vDash p \wedge q \vee \neg r$ car chaque fois que $I_v(p) = I_v(\neg r) = 1$ on a $I_v(p \wedge q \vee \neg r) = 1$ (lignes en orange).



La conséquence logique (exemples)

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1	0	0	0	0	1	1
v_2	0	0	1	0	0	0
v_3	0	1	0	0	1	1
v_4	0	1	1	0	0	0
v_5	1	0	0	0	1	1
v_6	1	0	1	0	0	0
v_7	1	1	0	1	1	1
v_8	1	1	1	1	0	1

- $\{p \wedge q, \neg r\} \models p \wedge q \vee \neg r$ car chaque fois que $I_v(p \wedge q) = I_v(\neg r) = 1$ on a $I_v(p \wedge q \vee \neg r) = 1$ (lignes en orange).



La conséquence logique (exemples)

	p	q	r	$p \wedge q$	$\neg r$	$p \wedge q \vee \neg r$
v_1	0	0	0	0	1	1
v_2	0	0	1	0	0	0
v_3	0	1	0	0	1	1
v_4	0	1	1	0	0	0
v_5	1	0	0	0	1	1
v_6	1	0	1	0	0	0
v_7	1	1	0	1	1	1
v_8	1	1	1	1	0	1

- $\{q, r\} \not\models p \wedge q \vee \neg r$ car :

pour $I_{v_8}(q) = I_{v_8}(r) = 1$ on a $I_{v_8}(p \wedge q \vee \neg r) = 1$ (en orange),

mais pour $I_{v_4}(q) = I_{v_4}(r) = 1$ on a $I_{v_4}(p \wedge q \vee \neg r) = 0$ (en rouge).



Lien entre conséquence et insatisfiabilité (théorème)

Théorème

$\mathcal{H} \models C$ si et seulement si $\mathcal{H} \cup \{\neg C\}$ est insatisfiable.



Lien entre conséquence et insatisfiabilité (théorème)

Théorème

$\mathcal{H} \models C$ si et seulement si $\mathcal{H} \cup \{\neg C\}$ est insatisfiable.

■ Il s'agit donc de démontrer que :

- 1 si $\mathcal{H} \models C$ alors $\mathcal{H} \cup \{\neg C\}$ est insatisfiable; (\Rightarrow)
- 2 si $\mathcal{H} \cup \{\neg C\}$ est insatisfiable alors $\mathcal{H} \models C$. (\Leftarrow)



Lien entre conséquence et insatisfiabilité (preuve (\Rightarrow))

- Raisonnement par l'absurde : on suppose que

$$\mathcal{H} \models C \quad (\text{Hyp})$$

et que $\mathcal{H} \cup \{\neg C\}$ est satisfiable (Abs)



Lien entre conséquence et insatisfiabilité (preuve (\Rightarrow))

- Raisonnement par l'absurde : on suppose que

$$\mathcal{H} \models C \quad (\text{Hyp})$$

et que $\mathcal{H} \cup \{\neg C\}$ est satisfiable (Abs)

- d'après (Abs), il existe v telle que $I_v(\mathcal{H}) = 1$ et $I_v(\neg C) = 1$, i.e. $I_v(C) = 0$;



Lien entre conséquence et insatisfiabilité (preuve (\Rightarrow))

- Raisonnement par l'absurde : on suppose que

$$\mathcal{H} \models C \quad (\text{Hyp})$$

et que $\mathcal{H} \cup \{\neg C\}$ est satisfiable (Abs)

- d'après (Abs), il existe v telle que $I_v(\mathcal{H}) = 1$ et $I_v(\neg C) = 1$, i.e. $I_v(C) = 0$;
- or d'après (Hyp), tout modèle de \mathcal{H} est un modèle de C : contradiction !

□



Lien entre conséquence et insatisfiabilité (preuve (\Leftarrow))

- Raisonnement par l'absurde : on suppose que

$\mathcal{H} \cup \{\neg C\}$ est insatisfiable (Hyp)

et que $\mathcal{H} \not\models C$ (Abs)



Lien entre conséquence et insatisfiabilité (preuve (\Leftarrow))

- Raisonnement par l'absurde : on suppose que

$\mathcal{H} \cup \{\neg C\}$ est insatisfiable (Hyp)

et que $\mathcal{H} \not\models C$ (Abs)

- d'après (Abs), il existe v_i telle que $I_{v_i}(\mathcal{H}) = 1$ et $I_{v_i}(C) = 0$; (*)



Lien entre conséquence et insatisfiabilité (preuve (\Leftarrow))

- Raisonnement par l'absurde : on suppose que

$\mathcal{H} \cup \{\neg C\}$ est insatisfiable (Hyp)

et que $\mathcal{H} \not\models C$ (Abs)

- d'après (Abs), il existe v_i telle que $I_{v_i}(\mathcal{H}) = 1$ et $I_{v_i}(C) = 0$; (*)
- or d'après (Hyp), pour tout v on a $I_v(\mathcal{H}) = 0$ ou $I_v(\neg C) = 0$, donc en particulier $I_{v_i}(\mathcal{H}) = 0$ ou $I_{v_i}(\neg C) = 0$;



Lien entre conséquence et insatisfiabilité (preuve (\Leftarrow))

- Raisonnement par l'absurde : on suppose que

$\mathcal{H} \cup \{\neg C\}$ est insatisfiable (Hyp)

et que $\mathcal{H} \not\models C$ (Abs)

- d'après (Abs), il existe v_i telle que $I_{v_i}(\mathcal{H}) = 1$ et $I_{v_i}(C) = 0$; (*)
- or d'après (Hyp), pour tout v on a $I_v(\mathcal{H}) = 0$ ou $I_v(\neg C) = 0$, donc en particulier $I_{v_i}(\mathcal{H}) = 0$ ou $I_{v_i}(\neg C) = 0$;
- comme d'après (*) $I_{v_i}(\mathcal{H}) = 1$, alors c'est que $I_{v_i}(\neg C) = 0$, i.e. $I_{v_i}(C) = 1$: contradiction avec (*)!

□



Lien entre conséquence et insatisfiabilité (exemple 1)

- On a vu précédemment (cf. Diapo 149) que :

$$\{p, \neg r\} \vDash p \wedge q \vee \neg r$$



Lien entre conséquence et insatisfiabilité (exemple 1)

- On a vu précédemment (cf. Diapo 149) que :

$$\{p, \neg r\} \models p \wedge q \vee \neg r$$

- d'après le théorème précédent (cf. Diapo 150) :

$$\{p, \neg r\} \models p \wedge q \vee \neg r$$

ssi

$\mathcal{H}' = \{p, \neg r\} \cup \{\neg(p \wedge q \vee \neg r)\}$ est **insatisfiable**
(i.e., il n'y a **aucun** modèle de \mathcal{H}');



Lien entre conséquence et insatisfiabilité (exemple 1)

- On a vu précédemment (cf. Diapo 149) que :

$$\{p, \neg r\} \models p \wedge q \vee \neg r$$

- d'après le théorème précédent (cf. Diapo 150) :

$$\{p, \neg r\} \models p \wedge q \vee \neg r$$

ssi

$\mathcal{H}' = \{p, \neg r\} \cup \{\neg(p \wedge q \vee \neg r)\}$ est **insatisfiable**

(i.e., il n'y a **aucun** modèle de \mathcal{H}');

- autrement dit, il faut montrer que **toute valuation falsifie au moins une formule** de $\mathcal{H}' = \{p, \neg r, \neg(p \wedge q \vee \neg r)\}$.



Lien entre conséquence et insatisfiabilité (exemple 1 - fin)

p	q	r	$\neg r$	$p \wedge q$	$p \wedge q \vee \neg r$	$\neg(p \wedge q \vee \neg r)$	\mathcal{H}'
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	0
1	0	0	1	0	1	0	0
1	0	1	0	0	0	1	0
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	0

- ▶ $\mathcal{H}' = \{p, \neg r, \neg(p \wedge q \vee \neg r)\}$ est insatisfiable
- ▶ $\{p, \neg r\} \vDash p \wedge q \vee \neg r$



Lien entre conséquence et insatisfiabilité (exemple 2)

- On a vu précédemment (cf. Diapo 149) que :

$$\{q, r\} \not\models p \wedge q \vee \neg r$$



Lien entre conséquence et insatisfiabilité (exemple 2)

- On a vu précédemment (cf. Diapo 149) que :

$$\{q, r\} \not\models p \wedge q \vee \neg r$$

- d'après le théorème précédent (cf. Diapo 150) :

$$\{q, r\} \not\models p \wedge q \vee \neg r$$

ssi

$\mathcal{H}' = \{q, r\} \cup \{\neg(p \wedge q \vee \neg r)\}$ est satisfiable
(i.e., il existe au moins un modèle de \mathcal{H}');



Lien entre conséquence et insatisfiabilité (exemple 2)

- On a vu précédemment (cf. Diapo 149) que :

$$\{q, r\} \not\models p \wedge q \vee \neg r$$

- d'après le théorème précédent (cf. Diapo 150) :

$$\{q, r\} \not\models p \wedge q \vee \neg r$$

ssi

$\mathcal{H}' = \{q, r\} \cup \{\neg(p \wedge q \vee \neg r)\}$ est satisfiable
(i.e., il existe au moins un modèle de \mathcal{H}');

- autrement dit, il faut montrer qu'**au moins une valuation est modèle de toutes les formules** de $\mathcal{H}' = \{q, r, \neg(p \wedge q \vee \neg r)\}$.



Lien entre conséquence et insatisfiabilité (exemple 2 - fin)

- Il suffit de trouver une valuation v telle que :
 - $I_v(q) = I_v(r) = I_v(\neg(p \wedge q \vee \neg r)) = 1$;



Lien entre conséquence et insatisfiabilité (exemple 2 - fin)

- Il suffit de trouver une valuation v telle que :
 - $I_v(q) = I_v(r) = I_v(\neg(p \wedge q \vee \neg r)) = 1$;
 - ssi $I_v(q) = I_v(r) = 1$ et $I_v(p \wedge q \vee \neg r) = 0$;



Lien entre conséquence et insatisfiabilité (exemple 2 - fin)

- Il suffit de trouver une valuation v telle que :
 - $I_v(q) = I_v(r) = I_v(\neg(p \wedge q \vee \neg r)) = 1$;
 - ssi $I_v(q) = I_v(r) = 1$ et $I_v(p \wedge q \vee \neg r) = 0$;
 - ssi $I_v(q) = I_v(r) = 1$ et $I_v(p \wedge q) = 0$ et $I_v(\neg r) = 0$;



Lien entre conséquence et insatisfiabilité (exemple 2 - fin)

- Il suffit de trouver une valuation v telle que :
 - $I_v(q) = I_v(r) = I_v(\neg(p \wedge q \vee \neg r)) = 1$;
 - ssi $I_v(q) = I_v(r) = 1$ et $I_v(p \wedge q \vee \neg r) = 0$;
 - ssi $I_v(q) = I_v(r) = 1$ et $I_v(p \wedge q) = 0$ et $I_v(\neg r) = 0$;
 - ssi $I_v(q) = I_v(r) = 1$ et $I_v(p) = 0$ puisque $I_v(q) = 1$;



Lien entre conséquence et insatisfiabilité (exemple 2 - fin)

- Il suffit de trouver une valuation v telle que :
 - $I_v(q) = I_v(r) = I_v(\neg(p \wedge q \vee \neg r)) = 1$;
 - ssi $I_v(q) = I_v(r) = 1$ et $I_v(p \wedge q \vee \neg r) = 0$;
 - ssi $I_v(q) = I_v(r) = 1$ et $I_v(p \wedge q) = 0$ et $I_v(\neg r) = 0$;
 - ssi $I_v(q) = I_v(r) = 1$ et $I_v(p) = 0$ puisque $I_v(q) = 1$;
- ainsi, la valuation v telle que : $v(p) = 0$, $v(q) = v(r) = 1$ est une modèle de $\mathcal{H}' = \{q, r, \neg(p \wedge q \vee \neg r)\}$, donc $\{q, r\} \not\models p \wedge q \vee \neg r$.



Équivalence de formules (définition et propriétés)

- Deux formules A et B sont **équivalentes** ssi elles ont les mêmes modèles ;
- et on note alors $A \equiv B$;
- propriétés : $A \equiv B$ ssi
 - $B \equiv A$;
 - A et B ont les mêmes tables de vérité ;
 - $\models A \leftrightarrow B$;
 - $\{A\} \models B$ et $\{B\} \models A$
(A et B sont conséquences logiques l'une de l'autre).



Équivalence de formules (à quoi ça sert?)

- L'équivalence logique permet :
 - de parler de l'équivalence logique (égalité des modèles) de deux formules en dehors de la logique ;
 - de définir des **abréviations** pour des formules complexes et rendre le langage logique manipulé plus compact ;
 - de trouver des formules équivalentes dont certaines propriétés sont déjà connues (validité, satisfiabilité...) ;
 - d'éliminer certains opérateurs en les exprimant par d'autres opérateurs (formes normales)...



Équivalence de formules (exemple et contre-exemple)

$$A \wedge B \stackrel{?}{\equiv} B \wedge A$$



Équivalence de formules (exemple et contre-exemple)

$$A \wedge B \stackrel{?}{=} B \wedge A$$

0

0

1

1



Équivalence de formules (exemple et contre-exemple)

$$A \wedge B \stackrel{?}{=} B \wedge A$$

$$\begin{array}{cc} 0 & 0 \end{array}$$

$$\begin{array}{cc} 0 & 1 \end{array}$$

$$\begin{array}{c} 1 \\ \end{array}$$

$$\begin{array}{c} 1 \\ \end{array}$$



Équivalence de formules (exemple et contre-exemple)

$$A \quad \wedge \quad B \quad \stackrel{?}{\equiv} \quad B \quad \wedge \quad A$$

$$\begin{array}{cc} 0 & 0 \end{array}$$

$$\begin{array}{cc} 0 & 1 \end{array}$$

$$\begin{array}{cc} 1 & 0 \end{array}$$

$$\begin{array}{cc} 1 & 1 \end{array}$$



Équivalence de formules (exemple et contre-exemple)

$$A \quad \wedge \quad B \quad \stackrel{?}{\equiv} \quad B \quad \wedge \quad A$$

$$0 \quad 0 \quad 0$$

$$0 \quad 0 \quad 1$$

$$1 \quad 0 \quad 0$$

$$1 \quad 1 \quad 1$$



Équivalence de formules (exemple et contre-exemple)

A	\wedge	B	$\stackrel{?}{\equiv}$	B	\wedge	A
0	0	0		0		0
0	0	1		1		0
1	0	0		0		1
1	1	1		1		1



Équivalence de formules (exemple et contre-exemple)

A	\wedge	B	$\stackrel{?}{\equiv}$	B	\wedge	A
0	0	0		0	0	0
0	0	1		1	0	0
1	0	0		0	0	1
1	1	1		1	1	1



Équivalence de formules (exemple et contre-exemple)

A	\wedge	B	\equiv	B	\wedge	A
0	0	0	1	0	0	0
0	0	1	1	1	0	0
1	0	0	1	0	0	1
1	1	1	1	1	1	1

Donc les 2 formules sont
équivalentes



Équivalence de formules (exemple et contre-exemple)

$$\begin{array}{llllll} A & \wedge & B & \equiv & B & \wedge & A \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

$$A \rightarrow B \stackrel{?}{\equiv} B \rightarrow A$$

Donc les 2 formules sont équivalentes



Équivalence de formules (exemple et contre-exemple)

$$\begin{array}{llllll} A & \wedge & B & \equiv & B & \wedge & A \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

$$\begin{array}{llll} A & \rightarrow & B & \stackrel{?}{\equiv} & B & \rightarrow & A \\ \hline 0 & & 1 & & 1 & & 0 \end{array}$$

Donc les 2 formules sont équivalentes



Équivalence de formules (exemple et contre-exemple)

$$\begin{array}{llllll} A & \wedge & B & \equiv & B & \wedge & A \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

$$\begin{array}{llllll} A & \rightarrow & B & \stackrel{?}{\equiv} & B & \rightarrow & A \\ \hline 0 & 1 & 1 & & 1 & 0 & 0 \end{array}$$

Donc les 2 formules sont équivalentes



Équivalence de formules (exemple et contre-exemple)

A	\wedge	B	\equiv	B	\wedge	A
0	0	0	1	0	0	0
0	0	1	1	1	0	0
1	0	0	1	0	0	1
1	1	1	1	1	1	1

A	\rightarrow	B	$\not\equiv$	B	\rightarrow	A
0	1	1	0	1	0	0

Donc les 2 formules **sont**
équivalentes

Donc les 2 formules **ne sont**
pas équivalentes



Équivalence de formules (équivalences remarquables)

$\neg(A \vee B) \equiv \neg A \wedge \neg B$	$A \vee \top \equiv \top$	$A \vee \perp \equiv A$	$A \vee A \equiv A$
$\neg(A \wedge B) \equiv \neg A \vee \neg B$	$A \wedge \perp \equiv \perp$	$A \wedge \top \equiv A$	$A \wedge A \equiv A$
$A \vee B \equiv B \vee A$	$A \rightarrow \top \equiv \top$	$\perp \rightarrow A \equiv \top$	$A \vee \neg A \equiv \top$
$A \wedge B \equiv B \wedge A$	$A \rightarrow \perp \equiv \neg A$	$\top \rightarrow A \equiv A$	$A \wedge \neg A \equiv \perp$
$A \rightarrow B \equiv \neg A \vee B$ (*)	$\neg \top \equiv \perp$	$\neg \perp \equiv \top$	$\neg \neg A \equiv A$
$A \oplus B \equiv (A \wedge \neg B) \vee (\neg A \wedge B)$	$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$		

$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$	$A \wedge (B \vee C) \equiv A \wedge B \vee A \wedge C$
$(A \vee B) \vee C \equiv A \vee (B \vee C)$	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

(*) À relier avec la définition de l'interprétation de l'implication (cf. Diapo 134).



Équivalence de formules (exemple 1 d'utilisation)

- À l'aide des équivalences remarquables précédentes, transformer $\neg p \rightarrow q \wedge \neg r$ de manière à trouver une expression équivalente n'utilisant que des opérateurs \neg et \vee :



Équivalence de formules (exemple 1 d'utilisation)

- À l'aide des équivalences remarquables précédentes, transformer $\neg p \rightarrow q \wedge \neg r$ de manière à trouver une expression équivalente n'utilisant que des opérateurs \neg et \vee :

$$\neg p \rightarrow q \wedge \neg r \equiv \neg\neg p \vee q \wedge \neg r \quad (\text{car } A \rightarrow B \equiv \neg A \vee B)$$



Équivalence de formules (exemple 1 d'utilisation)

- À l'aide des équivalences remarquables précédentes, transformer $\neg p \rightarrow q \wedge \neg r$ de manière à trouver une expression équivalente n'utilisant que des opérateurs \neg et \vee :

$$\neg p \rightarrow q \wedge \neg r \equiv \neg\neg p \vee q \wedge \neg r$$

(car $A \rightarrow B \equiv \neg A \vee B$)

$$\equiv p \vee q \wedge \neg r$$

(car $\neg\neg A \equiv A$)



Équivalence de formules (exemple 1 d'utilisation)

- À l'aide des équivalences remarquables précédentes, transformer $\neg p \rightarrow q \wedge \neg r$ de manière à trouver une expression équivalente n'utilisant que des opérateurs \neg et \vee :

$$\neg p \rightarrow q \wedge \neg r \equiv \neg\neg p \vee q \wedge \neg r$$

(car $A \rightarrow B \equiv \neg A \vee B$)

$$\equiv p \vee q \wedge \neg r$$

(car $\neg\neg A \equiv A$)

$$\equiv p \vee \neg\neg q \wedge \neg r$$

(car $\neg\neg A \equiv A$)



Équivalence de formules (exemple 1 d'utilisation)

- À l'aide des équivalences remarquables précédentes, transformer $\neg p \rightarrow q \wedge \neg r$ de manière à trouver une expression équivalente n'utilisant que des opérateurs \neg et \vee :

$$\begin{aligned}\neg p \rightarrow q \wedge \neg r &\equiv \neg\neg p \vee q \wedge \neg r && (\text{car } A \rightarrow B \equiv \neg A \vee B) \\ &\equiv p \vee q \wedge \neg r && (\text{car } \neg\neg A \equiv A) \\ &\equiv p \vee \neg\neg q \wedge \neg r && (\text{car } \neg\neg A \equiv A) \\ &\equiv p \vee \neg(\neg q \vee r) && (\text{car } \neg A \wedge \neg B \equiv \neg(A \vee B))\end{aligned}$$



Équivalence de formules (exemple 2 d'utilisation)

- Montrer que $\neg A$ et $A \rightarrow \perp$ sont des formules équivalentes :



Équivalence de formules (exemple 2 d'utilisation)

- Montrer que $\neg A$ et $A \rightarrow \perp$ sont des formules équivalentes :
- cela revient à montrer que ces formules ont les mêmes tables de vérité :



Équivalence de formules (exemple 2 d'utilisation)

- Montrer que $\neg A$ et $A \rightarrow \perp$ sont des formules équivalentes :
- cela revient à montrer que ces formules ont les mêmes tables de vérité :

A	\perp	$\neg A$	$A \rightarrow \perp$
0	0	1	1
1	0	0	0

- ce qui est bien le cas (donc elles sont équivalentes).



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A$	\wedge	$B)$	\leftrightarrow	\neg	A	\vee	\neg	B



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A \wedge B)$	\leftrightarrow	$\neg A \vee \neg B$
	0		0
	0		0
	1		1
	1		1



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A \wedge B)$	\leftrightarrow	$\neg A \vee \neg B$
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A \wedge B)$	\leftrightarrow	$\neg A \vee \neg B$
	0	0	0
	0	0	1
	1	0	0
	1	1	1



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A$	\wedge	$B)$	\leftrightarrow	\neg	A	\vee	\neg	B
1	0	0	0			0			0
1	0	0	1			0			1
1	1	0	0			1			0
0	1	1	1			1			1



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A$	\wedge	$B)$	\leftrightarrow	\neg	A	\vee	\neg	B
1	0	0	0		1	0			0
1	0	0	1		1	0			1
1	1	0	0		0	1			0
0	1	1	1		0	1			1



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A$	\wedge	$B)$	\leftrightarrow	\neg	A	\vee	\neg	B
1	0	0	0		1	0		1	0
1	0	0	1		1	0		0	1
1	1	0	0		0	1		1	0
0	1	1	1		0	1		0	1



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A$	\wedge	$B)$	\leftrightarrow	\neg	A	\vee	\neg	B
1	0	0	0		1	0	1	1	0
1	0	0	1		1	0	1	0	1
1	1	0	0		0	1	1	1	0
0	1	1	1		0	1	0	0	1



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A$	\wedge	$B)$	\leftrightarrow	\neg	A	\vee	\neg	B
1	0	0	0	1	1	0	1	1	0
1	0	0	1	1	1	0	1	0	1
1	1	0	0	1	0	1	1	1	0
0	1	1	1	1	0	1	0	0	1



Équivalence de formules (exemple 3 d'utilisation)

- De même, montrer que $\neg(A \wedge B) \equiv \neg A \vee \neg B$:
- ce qui peut se faire en montrant que $\models \neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$:

\neg	$(A \wedge B)$	\leftrightarrow	$\neg A \vee \neg B$
1	0	0	0
1	0	0	1
1	1	0	1
0	1	1	0
1	0	1	1
1	1	1	0
0	1	0	0
0	0	0	1

- ce qui est bien le cas (donc elles sont équivalentes).



Résumé

- Différence syntaxe / sémantique ;
- on définit la **sémantique** d'une formule à l'aide d'une **valuation** étendue à une fonction d'**interprétation** ;
- correspondance entre interprétations et tables de vérité ;
- modèles d'une formule ;
- validité et satisfiabilité d'une formule ;
- conséquence logique, équivalence logique, base pour le calcul booléen.



Logique des propositions

Résolution



Conséquence logique vs déduction

- La **conséquence logique** est basée sur la **théorie des modèles** :
 - $\mathcal{H} \models C$ ssi tout modèle de l'ensemble \mathcal{H} est modèle de la formule C ;



Conséquence logique vs déduction

- La **conséquence logique** est basée sur la **théorie des modèles** :
 - $\mathcal{H} \models C$ ssi tout modèle de l'ensemble \mathcal{H} est modèle de la formule C ;
- la **déduction** (de nouvelles formules) est basée sur la **théorie de la preuve** :
 - notion purement syntaxique notée \vdash ;
 - par application de règles : les **règles d'inférence**;



Conséquence logique vs déduction

- La **conséquence logique** est basée sur la **théorie des modèles** :
 - $\mathcal{H} \models C$ ssi tout modèle de l'ensemble \mathcal{H} est modèle de la formule C ;
- la **déduction** (de nouvelles formules) est basée sur la **théorie de la preuve** :
 - notion purement syntaxique notée \vdash ;
 - par application de règles : les **règles d'inférence**;
 - **avantage** : implantation simple et efficace sur ordinateur;
 - ... mais sans ambition d'être compréhensible pour des humains.



Principe d'une méthode de déduction

- À partir de données :
 - des faits : p, r ;
 - des implications : $p \rightarrow q, r \rightarrow q \rightarrow s, q \rightarrow s \rightarrow u$;
 - un but : prouver u



Principe d'une méthode de déduction

- À partir de données :
 - des faits : p, r ;
 - des implications : $p \rightarrow q, r \rightarrow q \rightarrow s, q \rightarrow s \rightarrow u$;
 - un but : prouver u
- il s'agit de propager des faits dans les implications (*modus ponens*) afin de dériver le but :



Principe d'une méthode de déduction

- À partir de données :
 - des faits : p, r ;
 - des implications : $p \rightarrow q, r \rightarrow q \rightarrow s, q \rightarrow s \rightarrow u$;
 - un but : prouver u
- il s'agit de propager des faits dans les implications (*modus ponens*) afin de dériver le but :
 - 1 p et $p \rightarrow q$ donnent q



Principe d'une méthode de déduction

- À partir de données :
 - des faits : p, r ;
 - des implications : $p \rightarrow q, r \rightarrow q \rightarrow s, q \rightarrow s \rightarrow u$;
 - un but : prouver u
- il s'agit de propager des faits dans les implications (*modus ponens*) afin de dériver le but :
 - 1 p et $p \rightarrow q$ donnent q
 - 2 r et $r \rightarrow q \rightarrow s$ donnent $q \rightarrow s$



Principe d'une méthode de déduction

- À partir de données :
 - des faits : p, r ;
 - des implications : $p \rightarrow q, r \rightarrow q \rightarrow s, q \rightarrow s \rightarrow u$;
 - un but : prouver u
- il s'agit de propager des faits dans les implications (*modus ponens*) afin de dériver le but :
 - 1 p et $p \rightarrow q$ donnent q
 - 2 r et $r \rightarrow q \rightarrow s$ donnent $q \rightarrow s$
 - 3 q et $q \rightarrow s \rightarrow u$ donnent $s \rightarrow u$



Principe d'une méthode de déduction

- À partir de données :
 - des faits : p, r ;
 - des implications : $p \rightarrow q, r \rightarrow q \rightarrow s, q \rightarrow s \rightarrow u$;
 - un but : prouver u
- il s'agit de propager des faits dans les implications (*modus ponens*) afin de dériver le but :
 - 1 p et $p \rightarrow q$ donnent q
 - 2 r et $r \rightarrow q \rightarrow s$ donnent $q \rightarrow s$
 - 3 q et $q \rightarrow s \rightarrow u$ donnent $s \rightarrow u$
 - 4 q et $q \rightarrow s$ donnent s



Principe d'une méthode de déduction

- À partir de données :
 - des faits : p, r ;
 - des implications : $p \rightarrow q, r \rightarrow q \rightarrow s, q \rightarrow s \rightarrow u$;
 - un but : prouver u
- il s'agit de propager des faits dans les implications (*modus ponens*) afin de dériver le but :

<p>1 p et $p \rightarrow q$ donnent q</p> <p>2 r et $r \rightarrow q \rightarrow s$ donnent $q \rightarrow s$</p> <p>3 q et $q \rightarrow s \rightarrow u$ donnent $s \rightarrow u$</p>	<p>4 q et $q \rightarrow s$ donnent s</p> <p>5 s et $s \rightarrow u$ donnent u</p>
--	---



Principe d'une méthode par résolution

- Méthode de déduction particulière où :
 - les implications (du type $p \rightarrow q$) sont remplacées par des disjonctions (du type $\neg p \vee q$) ;
 - le *modus ponens* est remplacé par la règle de résolution :

p et $\neg p \vee q$ donnent q .



Principe d'une méthode par résolution

- Méthode de déduction particulière où :
 - les **implications** (du type $p \rightarrow q$) sont **remplacées** par des **disjonctions** (du type $\neg p \vee q$);
 - le *modus ponens* est remplacé par la **règle de résolution** :

p et $\neg p \vee q$ donnent q .

- **Plus généralement** la règle de résolution est (pour $p \in PROP$ et $A, B \in FORM$) :

$p \vee A$ et $\neg p \vee B$ donnent $A \vee B$.



La méthode par résolution (remarques)

- La résolution **est** :
 - une méthode de déduction particulière ;
 - assez récente (Alan Robinson, *A Machine-Oriented Logic Based on the Resolution Principle*, 1965) ;
 - dont les premières applications furent pour l'intelligence artificielle.



La méthode par résolution (remarques)

- La résolution *est* :
 - une méthode de déduction particulière ;
 - assez récente (Alan Robinson, *A Machine-Oriented Logic Based on the Resolution Principle*, 1965) ;
 - dont les premières applications furent pour l'intelligence artificielle.
- La résolution *utilise* :
 - une seule *règle d'inférence* ;
 - un ensemble de formules ayant subi un *pré-traitement syntaxique*.



Démarche de la méthode par résolution

- Pour une formule $A \in FORM$:
 - 1 construction d'une forme clausale de A :
 - 2 application de la résolution.



Démarche de la méthode par résolution

- Pour une formule $A \in FORM$:
 - 1 construction d'une forme clausale de A :
 - 1 mise sous forme normale conjonctive de A ;
 - 2 application de la résolution.



Démarche de la méthode par résolution

■ Pour une formule $A \in FORM$:

- 1 construction d'une forme clausale de A :
 - 1 mise sous forme normale conjonctive de A ;
 - 2 élimination des connecteurs \wedge et \vee afin de réécrire A sous la forme d'un ensemble de clauses (logiquement équivalent à A);
- 2 application de la résolution.



Définition d'un littéral

- Un **littéral** est :
 - une variable propositionnelle (**littéral positif**) ;
 - ou sa négation (**littéral négatif**).



Définition d'un littéral

- Un **littéral** est :
 - une variable propositionnelle (**littéral positif**) ;
 - ou sa négation (**littéral négatif**).
- Ainsi :
 - $p, \neg q$ sont des littéraux ;
 - mais pas : $p \vee \neg q$, ni \perp ;



Définition d'un littéral

- Un **littéral** est :
 - une variable propositionnelle (**littéral positif**) ;
 - ou sa négation (**littéral négatif**).
- Ainsi :
 - $p, \neg q$ sont des littéraux ;
 - mais pas : $p \vee \neg q$, ni \perp ;
- p est le **littéral complémentaire** de $\neg p$, et réciproquement ;



Définition d'un littéral

- Un **littéral** est :
 - une variable propositionnelle (**littéral positif**) ;
 - ou sa négation (**littéral négatif**).
- Ainsi :
 - $p, \neg q$ sont des littéraux ;
 - mais pas : $p \vee \neg q$, ni \perp ;
- p est le **littéral complémentaire** de $\neg p$, et réciproquement ;
- $LIT = \{p, \neg p : p \in PROP\}$ est l'ensemble de tous les littéraux possibles de $LProp$:
 - on note $l, l', \dots, l_1, l_2, \dots$ des éléments de LIT .



Définition d'une clause

- Une **clause** est une disjonction de littéraux ;



Définition d'une clause

- Une **clause** est une disjonction de littéraux ;
- définition inductive :
 - si $l \in LIT$, alors l est un clause ;
 - si A et B sont des clauses, alors $A \vee B$ est une clause ;



Définition d'une clause

- Une **clause** est une disjonction de littéraux ;
- définition inductive :
 - si $l \in LIT$, alors l est un clause ;
 - si A et B sont des clauses, alors $A \vee B$ est une clause ;
- conséquence : tout littéral est une clause ;
- ainsi :
 - p et $\neg q$ sont des clauses, ainsi que $p \vee \neg q$ et $p \vee \neg q \vee r$;
 - mais **pas** $\neg(p \vee \neg q)$ ni $\top \vee \neg q$.



Les formes normales (définition)

- une formule est en **forme normale conjonctive** (FNC) ssi c'est une **conjonction de disjonctions** de littéraux (ou conjonction de clauses) ;
 - exemple : $(\neg p \vee q) \wedge (r \vee s)$;



Les formes normales (définition)

- une formule est en **forme normale conjonctive** (FNC) ssi c'est une **conjonction de disjonctions** de littéraux (ou conjonction de clauses) ;
 - exemple : $(\neg p \vee q) \wedge (r \vee s)$;
- une formule est en **forme normale disjonctive** (FND) ssi c'est une **disjonction de conjonctions** de littéraux.
 - exemple : $(\neg p \wedge q) \vee (r \wedge s)$.



Les formes normales (cas particuliers)

- $p \vee \neg q$:



Les formes normales (cas particuliers)

- $p \vee \neg q$: c'est à la fois une FNC et une FND ;



Les formes normales (cas particuliers)

- $p \vee \neg q$: c'est à la fois une FNC et une FND ;
- $p \wedge \neg q$:



Les formes normales (cas particuliers)

- $p \vee \neg q$: c'est à la fois une FNC et une FND ;
- $p \wedge \neg q$: c'est à la fois une FNC et une FND ;



Les formes normales (cas particuliers)

- $p \vee \neg q$: c'est à la fois une FNC et une FND ;
- $p \wedge \neg q$: c'est à la fois une FNC et une FND ;
- p :



Les formes normales (cas particuliers)

- $p \vee \neg q$: c'est à la fois une FNC et une FND ;
- $p \wedge \neg q$: c'est à la fois une FNC et une FND ;
- p : c'est à la fois une FNC et une FND ;



Les formes normales (cas particuliers)

- $p \vee \neg q$: c'est à la fois une FNC et une FND ;
- $p \wedge \neg q$: c'est à la fois une FNC et une FND ;
- p : c'est à la fois une FNC et une FND ;
- $\neg(p \wedge q) \vee (r \wedge s)$:



Les formes normales (cas particuliers)

- $p \vee \neg q$: c'est à la fois une FNC et une FND ;
- $p \wedge \neg q$: c'est à la fois une FNC et une FND ;
- p : c'est à la fois une FNC et une FND ;
- $\neg(p \wedge q) \vee (r \wedge s)$: ce n'est ni une FNC ni une FND ;



Les formes normales (cas particuliers)

- $p \vee \neg q$: c'est à la fois une FNC et une FND ;
- $p \wedge \neg q$: c'est à la fois une FNC et une FND ;
- p : c'est à la fois une FNC et une FND ;
- $\neg(p \wedge q) \vee (r \wedge s)$: ce n'est ni une FNC ni une FND ;
- $(p \vee \perp) \wedge (r \wedge \neg s)$:



Les formes normales (cas particuliers)

- $p \vee \neg q$: c'est à la fois une FNC et une FND ;
- $p \wedge \neg q$: c'est à la fois une FNC et une FND ;
- p : c'est à la fois une FNC et une FND ;
- $\neg(p \wedge q) \vee (r \wedge s)$: ce n'est ni une FNC ni une FND ;
- $(p \vee \perp) \wedge (r \wedge \neg s)$: ce n'est ni une FNC ni une FND.



Définition inductive d'une FNC

- Si A est une clause, alors A est une FNC;
- si A et B sont des FNC, alors $A \wedge B$ est une FNC.
- exemples :
 - $p, \neg q, (p \vee q) \wedge r$ sont des FNC;
 - $p \vee q \wedge r$ n'est pas une FNC (d'après nos conventions sur le parenthésage).



Conversion d'une formule en FNC (étape 1/3)

- Cette étape consiste à éliminer les connecteurs autres que \neg, \wedge, \vee ;



Conversion d'une formule en FNC (étape 1/3)

- Cette étape consiste à éliminer les connecteurs autres que \neg, \wedge, \vee ;
- pour cela, on utilise les équivalences de formules (cf. Diapo 160) :
 - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$;
 - $p \rightarrow q \equiv \neg p \vee q$;
 - $p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$;
 - ...



Conversion d'une formule en FNC (étape 1/3)

- Cette étape consiste à éliminer les connecteurs autres que \neg, \wedge, \vee ;
- pour cela, on utilise les **équivalences de formules** (cf. Diapo 160) :
 - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$;
 - $p \rightarrow q \equiv \neg p \vee q$;
 - $p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$;
 - ...
- on obtient une formule équivalente à la formule initiale (cf. Diapo 157).



Conversion d'une formule en FNC (étape 2/3)

- Ici, il s'agit d'utiliser les **équivalences de formules** pour :
 - 1 distribuer les négations à l'intérieur des expressions entre parenthèses :
 - $\neg(p \wedge q) \equiv \neg p \vee \neg q$
 - $\neg(p \vee q) \equiv \neg p \wedge \neg q$



Conversion d'une formule en FNC (étape 2/3)

- Ici, il s'agit d'utiliser les **équivalences de formules** pour :
 - 1 distribuer les négations à l'intérieur des expressions entre parenthèses :
 - $\neg(p \wedge q) \equiv \neg p \vee \neg q$
 - $\neg(p \vee q) \equiv \neg p \wedge \neg q$
 - 2 éliminer les doubles négations :
 - $\neg\neg p \equiv p$.



Conversion d'une formule en FNC (étape 2/3)

- Ici, il s'agit d'utiliser les **équivalences de formules** pour :
 - 1 distribuer les négations à l'intérieur des expressions entre parenthèses :
 - $\neg(p \wedge q) \equiv \neg p \vee \neg q$
 - $\neg(p \vee q) \equiv \neg p \wedge \neg q$
 - 2 éliminer les doubles négations :
 - $\neg\neg p \equiv p$.
- et obtenir ainsi une formule équivalente à la formule initiale (cf. Diapo 157).



Conversion d'une formule en FNC (étape 3/3)

- Enfin, il s'agit d'utiliser les **équivalences** pour :

1 distribuer \vee sur \wedge :

- $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
- $(q \wedge r) \vee p \equiv (q \vee p) \wedge (r \vee p)$;



Conversion d'une formule en FNC (étape 3/3)

- Enfin, il s'agit d'utiliser les **équivalences** pour :

1 distribuer \vee sur \wedge :

- $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
- $(q \wedge r) \vee p \equiv (q \vee p) \wedge (r \vee p)$;

2 éliminer \perp et \top :

- $q \wedge (p \vee \perp) \equiv q \wedge p$;
- $q \wedge (p \vee \neg\perp) \equiv q \wedge (p \vee \top) \equiv q \wedge \top \equiv q$
- ...



Conversion d'une formule en FNC (étape 3/3)

- Enfin, il s'agit d'utiliser les **équivalences** pour :
 - 1 distribuer \vee sur \wedge :
 - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
 - $(q \wedge r) \vee p \equiv (q \vee p) \wedge (r \vee p)$;
 - 2 éliminer \perp et \top :
 - $q \wedge (p \vee \perp) \equiv q \wedge p$;
 - $q \wedge (p \vee \neg\perp) \equiv q \wedge (p \vee \top) \equiv q \wedge \top \equiv q$
 - ...
- et obtenir ainsi une formule équivalente à la formule initiale (cf. Diapo 157).



Conversion d'une formule en FNC (exemple 1 - étape 1/3)

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



Conversion d'une formule en FNC (exemple 1 - étape 1/3)

$$\begin{aligned} & \neg((p \wedge \neg q) \rightarrow (p \leftrightarrow \neg q)) \\ & \equiv \neg(\neg(p \wedge \neg q) \vee (p \leftrightarrow \neg q)) \end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 1/3)

$$\begin{aligned} & \neg((p \wedge \neg q) \rightarrow (p \leftrightarrow \neg q)) \\ & \equiv \neg(\neg(p \wedge \neg q) \vee (p \leftrightarrow \neg q)) \\ & \equiv \neg(\neg(p \wedge \neg q) \vee ((p \rightarrow \neg q) \wedge (\neg q \rightarrow p))) \end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 1/3)

$$\begin{aligned} & \neg((p \wedge \neg q) \rightarrow (p \leftrightarrow \neg q)) \\ & \equiv \neg(\neg(p \wedge \neg q) \vee (p \leftrightarrow \neg q)) \\ & \equiv \neg(\neg(p \wedge \neg q) \vee ((p \rightarrow \neg q) \wedge (\neg q \rightarrow p))) \\ & \equiv \neg(\neg(p \wedge \neg q) \vee ((\neg p \vee \neg q) \wedge (\neg \neg q \vee p))) \end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 2/3)

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



Conversion d'une formule en FNC (exemple 1 - étape 2/3)

$$\begin{aligned}& \neg(\neg(p \wedge \neg q) \vee ((\neg p \vee \neg q) \wedge (\neg \neg q \vee p))) \\& \equiv \neg\neg(p \wedge \neg q) \wedge \neg((\neg p \vee \neg q) \wedge (\neg \neg q \vee p))\end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 2/3)

$$\begin{aligned}& \neg(\neg(p \wedge \neg q) \vee ((\neg p \vee \neg q) \wedge (\neg \neg q \vee p))) \\& \equiv \neg\neg(p \wedge \neg q) \wedge \neg((\neg p \vee \neg q) \wedge (\neg \neg q \vee p)) \\& \equiv \neg\neg(p \wedge \neg q) \wedge (\neg(\neg p \vee \neg q) \vee \neg(\neg \neg q \vee p))\end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 2/3)

$$\begin{aligned}& \neg(\neg(p \wedge \neg q) \vee ((\neg p \vee \neg q) \wedge (\neg \neg q \vee p))) \\& \equiv \neg \neg(p \wedge \neg q) \wedge \neg((\neg p \vee \neg q) \wedge (\neg \neg q \vee p)) \\& \equiv \neg \neg(p \wedge \neg q) \wedge (\neg(\neg p \vee \neg q) \vee \neg(\neg \neg q \vee p)) \\& \equiv \neg \neg(p \wedge \neg q) \wedge ((\neg \neg p \wedge \neg \neg q) \vee (\neg \neg \neg q \wedge \neg p))\end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 2/3)

$$\begin{aligned}& \neg(\neg(p \wedge \neg q) \vee ((\neg p \vee \neg q) \wedge (\neg \neg q \vee p))) \\& \equiv \neg \neg(p \wedge \neg q) \wedge \neg((\neg p \vee \neg q) \wedge (\neg \neg q \vee p)) \\& \equiv \neg \neg(p \wedge \neg q) \wedge (\neg(\neg p \vee \neg q) \vee \neg(\neg \neg q \vee p)) \\& \equiv \neg \neg(p \wedge \neg q) \wedge ((\neg \neg p \wedge \neg \neg q) \vee (\neg \neg \neg q \wedge \neg p)) \\& \equiv (p \wedge \neg q) \wedge ((p \wedge q) \vee (\neg q \wedge \neg p))\end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 3/3)

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



Conversion d'une formule en FNC (exemple 1 - étape 3/3)

$$\begin{aligned}(p \wedge \neg q) \wedge ((p \wedge q) \vee (\neg q \wedge \neg p)) \\ \equiv (p \wedge \neg q) \wedge ((p \vee \neg q) \wedge (p \vee \neg p) \wedge (q \vee \neg q) \wedge (q \vee \neg p))\end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 3/3)

$$\begin{aligned}(p \wedge \neg q) \wedge ((p \wedge q) \vee (\neg q \wedge \neg p)) \\ \equiv (p \wedge \neg q) \wedge ((p \vee \neg q) \wedge (p \vee \neg p) \wedge (q \vee \neg q) \wedge (q \vee \neg p)) \\ \equiv (p \wedge \neg q) \wedge ((p \vee \neg q) \wedge \textcolor{red}{T} \wedge \textcolor{red}{T} \wedge (q \vee \neg p))\end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 3/3)

$$\begin{aligned}(p \wedge \neg q) \wedge ((p \wedge q) \vee (\neg q \wedge \neg p)) \\ \equiv (p \wedge \neg q) \wedge ((p \vee \neg q) \wedge (p \vee \neg p) \wedge (q \vee \neg q) \wedge (q \vee \neg p)) \\ \equiv (p \wedge \neg q) \wedge ((p \vee \neg q) \wedge \textcolor{red}{T} \wedge \textcolor{red}{T} \wedge (q \vee \neg p)) \\ \equiv (p \wedge \neg q) \wedge ((p \vee \neg q) \wedge (q \vee \neg p))\end{aligned}$$



Conversion d'une formule en FNC (exemple 1 - étape 3/3)

$$\begin{aligned}(p \wedge \neg q) \wedge ((p \wedge q) \vee (\neg q \wedge \neg p)) \\ \equiv (p \wedge \neg q) \wedge ((p \vee \neg q) \wedge (p \vee \neg p) \wedge (q \vee \neg q) \wedge (q \vee \neg p)) \\ \equiv (p \wedge \neg q) \wedge ((p \vee \neg q) \wedge \top \wedge \top \wedge (q \vee \neg p)) \\ \equiv (p \wedge \neg q) \wedge ((p \vee \neg q) \wedge (q \vee \neg p)) \\ \equiv p \wedge \neg q \wedge (p \vee \neg q) \wedge (q \vee \neg p)\end{aligned}$$



Conversion d'une formule en FNC (exemple 2)

$$\neg \left(((n \rightarrow f) \wedge (v \rightarrow f) \wedge (n \vee v) \wedge (e \rightarrow \neg f)) \rightarrow \neg f \right)$$



Conversion d'une formule en FNC (exemple 2)

$$\begin{aligned}& \neg \left(((n \rightarrow f) \wedge (v \rightarrow f) \wedge (n \vee v) \wedge (e \rightarrow \neg f)) \rightarrow \neg f \right) \\& \equiv \neg \left(\neg ((n \rightarrow f) \wedge (v \rightarrow f) \wedge (n \vee v) \wedge (e \rightarrow \neg f)) \vee \neg f \right)\end{aligned}$$



Conversion d'une formule en FNC (exemple 2)

$$\begin{aligned} & \neg\left(\left((n \rightarrow f) \wedge (v \rightarrow f) \wedge (n \vee v) \wedge (e \rightarrow \neg f)\right) \rightarrow \neg f\right) \\ & \equiv \neg\left(\neg\left((n \rightarrow f) \wedge (v \rightarrow f) \wedge (n \vee v) \wedge (e \rightarrow \neg f)\right) \vee \neg f\right) \\ & \equiv \left((n \rightarrow f) \wedge (v \rightarrow f) \wedge (n \vee v) \wedge (e \rightarrow \neg f)\right) \neg f \end{aligned}$$



Conversion d'une formule en FNC (exemple 2)

$$\begin{aligned} & \neg\left(\left((n \rightarrow f) \wedge (v \rightarrow f) \wedge (n \vee v) \wedge (e \rightarrow \neg f)\right) \rightarrow \neg f\right) \\ & \equiv \neg\left(\neg\left((n \rightarrow f) \wedge (v \rightarrow f) \wedge (n \vee v) \wedge (e \rightarrow \neg f)\right) \vee \neg f\right) \\ & \equiv \left((n \rightarrow f) \wedge (v \rightarrow f) \wedge (n \vee v) \wedge (e \rightarrow \neg f)\right) \wedge f \\ & \equiv (\neg n \vee f) \wedge (\neg v \vee f) \wedge (n \vee v) \wedge (\neg e \vee \neg f) \wedge f \end{aligned}$$



Vers la forme clausale d'une formule

- On a vu (cf. Diapo 170) que la résolution requiert que les formules soient sous **forme clausale** ;
- et la mise sous forme clausale d'une formule requiert la FNC de cette formule ;
- comment passer de la FNC à la forme clausale ?



Représentation des clauses

- Une **clause** (cf. Diapo 172) est représentée par l'**ensemble de ses littéraux** (cf. Diapo 171);
- par exemple :
 - p est représenté par l'ensemble $\{p\}$;
 - $\{p, \neg q\}$ représente la clause $p \vee \neg q$;



Représentation des clauses

- Une **clause** (cf. Diapo 172) est représentée par l'**ensemble de ses littéraux** (cf. Diapo 171);
- par exemple :
 - p est représenté par l'ensemble $\{p\}$;
 - $\{p, \neg q\}$ représente la clause $p \vee \neg q$, mais aussi :
 - $p \vee p \vee \neg q$;
 - $p \vee \neg q \vee \neg q$;
 - ...
 - ➡ toutes ces formules sont (**syntaxiquement**) différentes mais (**logiquement**) équivalentes.



Représentation de la clause vide

- La **clause vide** est la clause représentée par {} ;
- à quelle formule correspond logiquement la clause vide (qui est une clause... sans aucun littéral) ?



Représentation de la clause vide

- La **clause vide** est la clause représentée par {} ;
- à quelle formule correspond logiquement la clause vide (qui est une clause... sans aucun littéral) ?
 - une disjonction de littéraux est satisfaite ssi au moins l'un des littéraux est satisfait ;



Représentation de la clause vide

- La **clause vide** est la clause représentée par {} ;
- à quelle formule correspond logiquement la clause vide (qui est une clause... sans aucun littéral) ?
 - une disjonction de littéraux est satisfaite ssi au moins l'un des littéraux est satisfait ;
 - dans le cas d'un ensemble vide de littéraux, il est impossible de satisfaire la clause correspondante ;



Représentation de la clause vide

- La **clause vide** est la clause représentée par {} ;
- à quelle formule correspond logiquement la clause vide (qui est une clause... sans aucun littéral) ?
 - une disjonction de littéraux est satisfaite ssi au moins l'un des littéraux est satisfait ;
 - dans le cas d'un ensemble vide de littéraux, il est impossible de satisfaire la clause correspondante ;
 - celle-ci correspond donc à la formule \perp (qui par définition n'est pas une clause).



Représentation de la clause tautologique

- Une **clause tautologique** est une clause contenant à la fois **un littéral et son littéral complémentaire** (cf. Diapo 171) ;
 - $p \vee \neg p$, $p \vee \neg p \vee q$, ... sont des clauses tautologiques ;
 - qui correspondent aux ensembles de littéraux $\{p, \neg p\}$, $\{p, \neg p, q\}$... respectivement ;
- Comme cette clause est toujours vraie, les ensembles de littéraux correspondants sont représentés par **T**.



Forme clausale

- Une conjonction de clauses (FNC) peut être représentée par l'ensemble de ses clauses, donc un ensemble d'ensembles de littéraux ;
- on appelle **forme clausale d'une formule A** l'ensemble d'ensembles de littéraux associé à la FNC de A ;
- la forme clausale de A est logiquement équivalente à A ;
- exemples : $\{\{p, \neg q\}, \{r, \neg p\}\}$ représente $(p \vee \neg q) \wedge (r \vee \neg p)$.



Forme clausale (avec clause tautologique)

- $\{\{l_1, \dots, l_n, \top\}, \{l'_1, \dots, l'_m\}, \dots\}$ et $\{\{l'_1, \dots, l'_m\}, \dots\}$
sont des formes clausales logiquement équivalentes car :



Forme clausale (avec clause tautologique)

- $\{\{l_1, \dots, l_n, \top\}, \{l'_1, \dots, l'_m\}, \dots\}$ et $\{\{l'_1, \dots, l'_m\}, \dots\}$

sont des **formes clausales logiquement équivalentes** car :

- 1 $\{l_1, \dots, l_n, \top\}$ et $\{\top\}$

- sont des clauses logiquement équivalentes
- puisque $A \vee \top \equiv \top$ (cf. Diapo 160);



Forme clausale (avec clause tautologique)

- $\{\{l_1, \dots, l_n, \top\}, \{l'_1, \dots, l'_m\}, \dots\}$ et $\{\{l'_1, \dots, l'_m\}, \dots\}$

sont des **formes clausales logiquement équivalentes** car :

- 1 $\{l_1, \dots, l_n, \top\}$ et $\{\top\}$

- sont des clauses logiquement équivalentes
- puisque $A \vee \top \equiv \top$ (cf. Diapo 160);

- 2 et que $\{\{\top\}, \{l'_1, \dots, l'_m\}, \dots\}$ et $\{\{l'_1, \dots, l'_m\}, \dots\}$

- sont des formes clausales logiquement équivalentes
- puisque $A \wedge \top \equiv A$ (cf. Diapo 160).



Forme clausale (avec clause tautologique)

- $\{\{l_1, \dots, l_n, \top\}, \{l'_1, \dots, l'_m\}, \dots\}$ et $\{\{l'_1, \dots, l'_m\}, \dots\}$

sont des **formes clausales logiquement équivalentes** car :

- 1 $\{l_1, \dots, l_n, \top\}$ et $\{\top\}$

- sont des clauses logiquement équivalentes
 - puisque $A \vee \top \equiv \top$ (cf. Diapo 160);

- 2 et que $\{\{\top\}, \{l'_1, \dots, l'_m\}, \dots\}$ et $\{\{l'_1, \dots, l'_m\}, \dots\}$

- sont des formes clausales logiquement équivalentes
 - puisque $A \wedge \top \equiv A$ (cf. Diapo 160).

- Dans une forme clausale, tout ensemble de littéraux contenant la **clause tautologique** peut être supprimé.



Forme clausale (avec clause contenant \perp)

- $\{\{l_1, \dots, l_n, \perp\}, \{l'_1, \dots, l'_m\}, \dots\}$ et $\{\{l_1, \dots, l_n\}, \{l'_1, \dots, l'_m\}, \dots\}$
sont des formes clausales logiquement équivalentes car :



Forme clausale (avec clause contenant \perp)

- $\{\{l_1, \dots, l_n, \perp\}, \{l'_1, \dots, l'_m\}, \dots\}$ et $\{\{l_1, \dots, l_n\}, \{l'_1, \dots, l'_m\}, \dots\}$
sont des formes clausales logiquement équivalentes car :
 - $\{l_1, \dots, l_n, \perp\}$ et $\{l_1, \dots, l_n\}$
 - sont des clauses logiquement équivalentes
 - puisque $A \vee \perp \equiv A$ (cf. Diapo 160) ;



Forme clausale (avec clause contenant \perp)

- $\{\{l_1, \dots, l_n, \perp\}, \{l'_1, \dots, l'_m\}, \dots\}$ et $\{\{l_1, \dots, l_n\}, \{l'_1, \dots, l'_m\}, \dots\}$
sont des formes clausales logiquement équivalentes car :
 - $\{l_1, \dots, l_n, \perp\}$ et $\{l_1, \dots, l_n\}$
 - sont des clauses logiquement équivalentes
 - puisque $A \vee \perp \equiv A$ (cf. Diapo 160);
- Dans une forme clausale, toute occurrence de \perp dans un ensemble de littéraux peut être supprimée.



Forme clausale (avec clause vide)

- $\{\{\}, \{\dots\}, \dots\}$ et \perp
sont logiquement équivalentes car
 - $A \wedge \perp \equiv \perp$ (cf. Diapo 160).



Forme clausale (avec clause vide)

- $\{\{\}, \{\dots\}, \dots\}$ et \perp
sont logiquement équivalentes car
 - $A \wedge \perp \equiv \perp$ (cf. Diapo 160).
- Dans une forme clausale, toute occurrence de la clause vide signifie que la formule correspondante est insatisfiable.



Forme clausale (avec clause vide)

- $\{\{\}, \{\dots\}, \dots\}$ et \perp
sont logiquement équivalentes car
 - $A \wedge \perp \equiv \perp$ (cf. Diapo 160).
- Dans une forme clausale, toute occurrence de la clause vide signifie que la formule correspondante est insatisfiable.
- 💡 Ne pas confondre l'ensemble vide de clauses $\{\}$ et l'ensemble de clauses contenant uniquement la clause vide $\{\{\}\}$.



Forme clausale vide

- La forme clausale vide $\{\}$ (à ne pas confondre avec $\{\{\}\}$) est assimilée à \top
- car ne contenant aucun littéral, elle ne peut être falsifiée (donc elle est valide).



Résolution entre 2 ensembles de littéraux (résolvante)

- Par commodité, on assimile désormais une **clause** et l'**ensemble de littéraux** qui la représente.
- Soient $\mathcal{C} \subseteq LIT$ et $\mathcal{C}' \subseteq LIT$ deux clauses telles que $\mathcal{C} = \{l_1, \dots, l_m, p\}$ et $\mathcal{C}' = \{l'_1, \dots, l'_n, \neg p\}$:



Résolution entre 2 ensembles de littéraux (résolvante)

- Par commodité, on assimile désormais une **clause** et l'**ensemble de littéraux** qui la représente.
- Soient $\mathcal{C} \subseteq LIT$ et $\mathcal{C}' \subseteq LIT$ deux clauses telles que $\mathcal{C} = \{l_1, \dots, l_m, p\}$ et $\mathcal{C}' = \{l'_1, \dots, l'_n, \neg p\}$:
 - résoudre \mathcal{C} et \mathcal{C}' consiste à les fusionner en éliminant **une seule paire** de littéraux complémentaires ;



Résolution entre 2 ensembles de littéraux (résolvante)

- Par commodité, on assimile désormais une **clause** et l'**ensemble de littéraux** qui la représente.
- Soient $\mathcal{C} \subseteq LIT$ et $\mathcal{C}' \subseteq LIT$ deux clauses telles que $\mathcal{C} = \{l_1, \dots, l_m, p\}$ et $\mathcal{C}' = \{l'_1, \dots, l'_n, \neg p\}$:
 - résoudre \mathcal{C} et \mathcal{C}' consiste à les fusionner en éliminant **une seule paire** de littéraux complémentaires ;
 - la clause qui en résulte est appelée la **résolvante** de \mathcal{C} et \mathcal{C}' :

$$\{l_1, \dots, l_m, l'_1, \dots, l'_n\}.$$



Résolution entre 2 ensembles de littéraux (résolvante)

- Par commodité, on assimile désormais une **clause** et l'**ensemble de littéraux** qui la représente.
- Soient $\mathcal{C} \subseteq LIT$ et $\mathcal{C}' \subseteq LIT$ deux clauses telles que $\mathcal{C} = \{l_1, \dots, l_m, p\}$ et $\mathcal{C}' = \{l'_1, \dots, l'_n, \neg p\}$:
 - résoudre \mathcal{C} et \mathcal{C}' consiste à les fusionner en éliminant **une seule paire** de littéraux complémentaires ;
 - la clause qui en résulte est appelée la **résolvante** de \mathcal{C} et \mathcal{C}' :
$$\{l_1, \dots, l_m, l'_1, \dots, l'_n\}.$$
- Exemple : la résolvante des clauses $\{p, q\}$ et $\{\neg r, \neg q\}$ est $\{p, \neg r\}$.



Résolution entre 2 ensembles de littéraux (cas particuliers)

- La résolvante de :

- $C = \{p\}$ et $C' = \{\neg p\}$ est la **clause vide** $\{\}$,



Résolution entre 2 ensembles de littéraux (cas particuliers)

- La résolvante de :

- $C = \{p\}$ et $C' = \{\neg p\}$ est la clause vide $\{\}$,
- $C = \{\neg p, q, r\}$ et $C' = \{\neg q, p\}$ est soit $\{\neg p, p, r\}$, soit $\{q, \neg q, r\}$, mais pas les deux puisqu'une seule paire de littéraux complémentaires doit être éliminée ! (cf. diapo suivante) ;



Résolution entre 2 ensembles de littéraux (cas particuliers)

- La résolvante de :
 - $\mathcal{C} = \{p\}$ et $\mathcal{C}' = \{\neg p\}$ est la clause vide $\{\}$,
 - $\mathcal{C} = \{\neg p, q, r\}$ et $\mathcal{C}' = \{\neg q, p\}$ est soit $\{\neg p, p, r\}$, soit $\{q, \neg q, r\}$, mais pas les deux puisqu'une seule paire de littéraux complémentaires doit être éliminée ! (cf. diapo suivante) ;
- la résolvante $\text{Res}(\mathcal{C}, \mathcal{C}')$ des clauses \mathcal{C} et \mathcal{C}' étant unique à une équivalence près :
 - c'est une fonction $\text{Res} : 2^{\text{LIT}} \times 2^{\text{LIT}} \longrightarrow 2^{\text{LIT}}$,
 - où $\text{Res}(\mathcal{C}, \mathcal{C}') = \text{Res}(\mathcal{C}', \mathcal{C})$.



Résolution entre 2 ensembles de littéraux (résolvante tautologique)

- $\text{Res}(\{\neg p, q, r\}, \{\neg q, p\})$ est :
 - soit $\{\neg p, p, r\}$;
 - soit $\{q, \neg q, r\}$;
- Remarque :
 - les clauses représentées par $\{\neg p, p, r\}$ et $\{q, \neg q, r\}$ sont équivalentes (à \top) ;
 - résoudre 2 paires de littéraux simultanément conduirait à un résultat erroné (ici, $\{r\}$)
 - ne résoudre qu'**une seule** paire à la fois !



Généralisation aux ensembles de résolvantes

- Soit $\mathcal{E} = \{\mathcal{C}_1, \dots, \mathcal{C}_n\}$ un ensemble de n clauses ;
- $\text{EnsRes}(\mathcal{E})$ dénote l'ensemble de toutes résolvantes des clauses de \mathcal{E} et :

$$\begin{aligned}\text{EnsRes}(\mathcal{E}) &= \bigcup_{\substack{1 \leq i < n \\ i < j \leq n}} \{\text{Res}(\mathcal{C}_i, \mathcal{C}_j)\} \\ &= \{\text{Res}(\mathcal{C}_1, \mathcal{C}_2), \text{Res}(\mathcal{C}_1, \mathcal{C}_3), \dots, \text{Res}(\mathcal{C}_1, \mathcal{C}_n)\} \cup \\ &\quad \{\text{Res}(\mathcal{C}_2, \mathcal{C}_3), \dots, \text{Res}(\mathcal{C}_2, \mathcal{C}_n)\} \cup \dots \cup \\ &\quad \{\text{Res}(\mathcal{C}_{n-1}, \mathcal{C}_n)\}\end{aligned}$$



Généralisation aux ensembles de résolvantes (exemple)

- Soit $\mathcal{E} = \{\{r\}_0, \{p, \neg q, r\}_1, \{\neg p, s\}_2, \{q, r, \neg s\}_3, \{\neg r\}_4, \{p, \neg s\}_5\}$;



Généralisation aux ensembles de résolvantes (exemple)

- Soit $\mathcal{E} = \{\{r\}_0, \{p, \neg q, r\}_1, \{\neg p, s\}_2, \{q, r, \neg s\}_3, \{\neg r\}_4, \{p, \neg s\}_5\}$;
- Calculer $\text{EnsRes}(\mathcal{E})$, signaler les clauses **vides** ou **tautologiques**.

$$\text{EnsRes}(\mathcal{E}) = \{$$



Généralisation aux ensembles de résolvantes (exemple)

- Soit $\mathcal{E} = \{\{r\}_0, \{p, \neg q, r\}_1, \{\neg p, s\}_2, \{q, r, \neg s\}_3, \{\neg r\}_4, \{p, \neg s\}_5\}$;
- Calculer $\text{EnsRes}(\mathcal{E})$, signaler les clauses **vides** ou **tautologiques**.

$$\text{EnsRes}(\mathcal{E}) = \{\{\}, \quad (\text{Res}(\mathcal{C}_0, \mathcal{C}_4))$$



Généralisation aux ensembles de résolvantes (exemple)

- Soit $\mathcal{E} = \{\{r\}_0, \{p, \neg q, r\}_1, \{\neg p, s\}_2, \{q, r, \neg s\}_3, \{\neg r\}_4, \{p, \neg s\}_5\}$;
- Calculer $\text{EnsRes}(\mathcal{E})$, signaler les clauses **vides** ou **tautologiques**.

$$\begin{aligned}\text{EnsRes}(\mathcal{E}) &= \{\{\}, \\ &\quad \{\neg q, r, s\},\end{aligned}\begin{aligned}&\quad (\text{Res}(\mathcal{C}_0, \mathcal{C}_4)) \\ &\quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_2))\end{aligned}$$



Généralisation aux ensembles de résolvantes (exemple)

- Soit $\mathcal{E} = \{\{r\}_0, \{p, \neg q, r\}_1, \{\neg p, s\}_2, \{q, r, \neg s\}_3, \{\neg r\}_4, \{p, \neg s\}_5\}$;
- Calculer $\text{EnsRes}(\mathcal{E})$, signaler les clauses **vides** ou **tautologiques**.

$$\begin{aligned}\text{EnsRes}(\mathcal{E}) &= \left\{ \{\}, \quad (\text{Res}(\mathcal{C}_0, \mathcal{C}_4)) \right. \\ &\quad \{\neg q, r, s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_2)) \\ &\quad \left. \{p, r, \neg s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_3)) \right.\end{aligned}$$



Généralisation aux ensembles de résolvantes (exemple)

- Soit $\mathcal{E} = \{\{r\}_0, \{p, \neg q, r\}_1, \{\neg p, s\}_2, \{q, r, \neg s\}_3, \{\neg r\}_4, \{p, \neg s\}_5\}$;
- Calculer $\text{EnsRes}(\mathcal{E})$, signaler les clauses **vides** ou **tautologiques**.

$$\begin{aligned}\text{EnsRes}(\mathcal{E}) &= \left\{ \{\}, \quad (\text{Res}(\mathcal{C}_0, \mathcal{C}_4)) \right. \\ &\quad \{\neg q, r, s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_2)) \\ &\quad \{p, r, \neg s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_3)) \\ &\quad \left. \{p, \neg q\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_4)) \right.\end{aligned}$$



Généralisation aux ensembles de résolvantes (exemple)

- Soit $\mathcal{E} = \{\{r\}_0, \{p, \neg q, r\}_1, \{\neg p, s\}_2, \{q, r, \neg s\}_3, \{\neg r\}_4, \{p, \neg s\}_5\}$;
- Calculer $\text{EnsRes}(\mathcal{E})$, signaler les clauses **vides** ou **tautologiques**.

$$\begin{aligned}\text{EnsRes}(\mathcal{E}) &= \left\{ \{\}, \quad (\text{Res}(\mathcal{C}_0, \mathcal{C}_4)) \right. \\ &\quad \{\neg q, r, s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_2)) \\ &\quad \{p, r, \neg s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_3)) \\ &\quad \{p, \neg q\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_4)) \\ &\quad \left. \{\neg p, q, r\}, \quad (\text{Res}(\mathcal{C}_2, \mathcal{C}_3)) \right.\end{aligned}$$



Généralisation aux ensembles de résolvantes (exemple)

- Soit $\mathcal{E} = \{\{r\}_0, \{p, \neg q, r\}_1, \{\neg p, s\}_2, \{q, r, \neg s\}_3, \{\neg r\}_4, \{p, \neg s\}_5\}$;
- Calculer $\text{EnsRes}(\mathcal{E})$, signaler les clauses **vides** ou **tautologiques**.

$$\begin{aligned}\text{EnsRes}(\mathcal{E}) = & \left\{ \{\}, \quad (\text{Res}(\mathcal{C}_0, \mathcal{C}_4)) \right. \\ & \{\neg q, r, s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_2)) \\ & \{p, r, \neg s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_3)) \\ & \{p, \neg q\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_4)) \\ & \{\neg p, q, r\}, \quad (\text{Res}(\mathcal{C}_2, \mathcal{C}_3)) \\ & \left. \{\neg p, p\} \text{ ou } \{\neg s, s\}, \quad (\text{Res}(\mathcal{C}_2, \mathcal{C}_5)) \right.\end{aligned}$$



Généralisation aux ensembles de résolvantes (exemple)

- Soit $\mathcal{E} = \{\{r\}_0, \{p, \neg q, r\}_1, \{\neg p, s\}_2, \{q, r, \neg s\}_3, \{\neg r\}_4, \{p, \neg s\}_5\}$;
- Calculer $\text{EnsRes}(\mathcal{E})$, signaler les clauses **vides** ou **tautologiques**.

$$\begin{aligned}\text{EnsRes}(\mathcal{E}) = & \left\{ \{\}, \quad (\text{Res}(\mathcal{C}_0, \mathcal{C}_4)) \right. \\ & \{\neg q, r, s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_2)) \\ & \{p, r, \neg s\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_3)) \\ & \{p, \neg q\}, \quad (\text{Res}(\mathcal{C}_1, \mathcal{C}_4)) \\ & \{\neg p, q, r\}, \quad (\text{Res}(\mathcal{C}_2, \mathcal{C}_3)) \\ & \{\neg p, p\} \text{ ou } \{\neg s, s\}, \quad (\text{Res}(\mathcal{C}_2, \mathcal{C}_5)) \\ & \left. \{q, \neg s\} \right\} \quad (\text{Res}(\mathcal{C}_3, \mathcal{C}_4))\end{aligned}$$



Généralisation aux ensembles de résolvantes (propriété)

\mathcal{E} et $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ ont les mêmes (contre-)modèles



Généralisation aux ensembles de résolvantes (démonstration)

- La propriété « \mathcal{E} et $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ ont les mêmes (contre-)modèles » est vraie
 - ssi pour toute valuation v , $I_v(\mathcal{E}) = I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E}))$;



Généralisation aux ensembles de résolvantes (démonstration)

- La propriété « \mathcal{E} et $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ ont les mêmes (contre-)modèles » est vraie
 - ssi pour toute valuation v , $I_v(\mathcal{E}) = I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E}))$;
 - ssi pour toute valuation v :
 (\Leftarrow) si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors $I_v(\mathcal{E}) = 1$;
 (\Rightarrow) si $I_v(\mathcal{E}) = 1$ alors $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$.



Généralisation aux ensembles de résolvantes (démonstration)

- La propriété « \mathcal{E} et $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ ont les mêmes (contre-)modèles » est vraie
 - ssi pour toute valuation v , $I_v(\mathcal{E}) = I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E}))$;
 - ssi pour toute valuation v :
 - (\Leftarrow) si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors $I_v(\mathcal{E}) = 1$;
 - (\Rightarrow) si $I_v(\mathcal{E}) = 1$ alors $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$.
- ! Si $\text{EnsRes}(\mathcal{E}) = \{\}$ alors la propriété est trivialement vraie. Dans la suite, on suppose que $\text{EnsRes}(\mathcal{E})$ contient au moins une résolvante.



Généralisation aux ensembles de résolvantes (preuve propriété (\Leftarrow))

- Il faut montrer : si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors $I_v(\mathcal{E}) = 1$;
- puisque toute clause est une formule, et que $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ est un ensemble de clauses, alors c'est un ensemble de formules ;



Généralisation aux ensembles de résolvantes (preuve propriété (\Leftarrow))

- Il faut montrer : si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors $I_v(\mathcal{E}) = 1$;
- puisque toute clause est une formule, et que $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ est un ensemble de clauses, alors c'est un ensemble de formules ;
- si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors :



Généralisation aux ensembles de résolvantes (preuve propriété (\Leftarrow))

- Il faut montrer : si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors $I_v(\mathcal{E}) = 1$;
- puisque toute clause est une formule, et que $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ est un ensemble de clauses, alors c'est un ensemble de formules ;
- si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors :
 - par définition (cf. Diapo 142), I_v est un modèle de chaque clause contenue dans $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$;



Généralisation aux ensembles de résolvantes (preuve propriété (\Leftarrow))

- Il faut montrer : si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors $I_v(\mathcal{E}) = 1$;
- puisque toute clause est une formule, et que $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ est un ensemble de clauses, alors c'est un ensemble de formules ;
- si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors :
 - par définition (cf. Diapo 142), I_v est un modèle de chaque clause contenue dans $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$;
 - donc, en particulier, des clauses contenues dans \mathcal{E} ;



Généralisation aux ensembles de résolvantes (preuve propriété (\Leftarrow))

- Il faut montrer : si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors $I_v(\mathcal{E}) = 1$;
- puisque toute clause est une formule, et que $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ est un ensemble de clauses, alors c'est un ensemble de formules ;
- si $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$ alors :
 - par définition (cf. Diapo 142), I_v est un modèle de chaque clause contenue dans $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$;
 - donc, en particulier, des clauses contenues dans \mathcal{E} ;
 - donc, $I_v(\mathcal{E}) = 1$.

□



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Il faut montrer que : si $I_v(\mathcal{E}) = 1$ alors $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$.



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Il faut montrer que : si $I_v(\mathcal{E}) = 1$ alors $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$.
- Soient $\mathcal{C} = \{l_1, \dots, l_m, p\}$ et $\mathcal{C}' = \{l'_1, \dots, l'_n, \neg p\}$ telles que :
 - $\mathcal{C}, \mathcal{C}' \in \mathcal{E}$ et
 - $\text{Res}(\mathcal{C}, \mathcal{C}') = \{l_1, \dots, l_m, l'_1, \dots, l'_n\} \in \text{EnsRes}(\mathcal{E})$;



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Il faut montrer que : si $I_v(\mathcal{E}) = 1$ alors $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$.
- Soient $\mathcal{C} = \{l_1, \dots, l_m, p\}$ et $\mathcal{C}' = \{l'_1, \dots, l'_n, \neg p\}$ telles que :
 - $\mathcal{C}, \mathcal{C}' \in \mathcal{E}$ et
 - $\text{Res}(\mathcal{C}, \mathcal{C}') = \{l_1, \dots, l_m, l'_1, \dots, l'_n\} \in \text{EnsRes}(\mathcal{E})$;
- Remarque :

$$I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1 \text{ ssi } \begin{cases} \text{il existe } l_i \in \mathcal{C} \text{ tel que } I_v(l_i) = 1 & (*) \\ \text{ou il existe } l'_j \in \mathcal{C}' \text{ tel que } I_v(l'_j) = 1 & (** \text{)} \end{cases}$$



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$
 - soit $I_v(p) = 1$



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$, alors il existe $l_i \in \mathcal{C}$ tel que $I_v(l_i) = 1$ (sinon $I_v(\mathcal{C})$ serait faux)
 - soit $I_v(p) = 1$



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$, alors il existe $l_i \in \mathcal{C}$ tel que $I_v(l_i) = 1$ (sinon $I_v(\mathcal{C})$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (*);
 - soit $I_v(p) = 1$



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$, alors il existe $l_i \in \mathcal{C}$ tel que $I_v(l_i) = 1$ (sinon $I_v(\mathcal{C})$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (*);
 - soit $I_v(p) = 1$, donc $I_v(\neg p) = 0$



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$, alors il existe $l_i \in \mathcal{C}$ tel que $I_v(l_i) = 1$ (sinon $I_v(\mathcal{C})$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (*);
 - soit $I_v(p) = 1$, donc $I_v(\neg p) = 0$, alors il existe $l'_j \in \mathcal{C}'$ tel que $I_v(l'_j) = 1$ (sinon $I_v(\mathcal{C}')$ serait faux)



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$, alors il existe $l_i \in \mathcal{C}$ tel que $I_v(l_i) = 1$ (sinon $I_v(\mathcal{C})$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (*);
 - soit $I_v(p) = 1$, donc $I_v(\neg p) = 0$, alors il existe $l'_j \in \mathcal{C}'$ tel que $I_v(l'_j) = 1$ (sinon $I_v(\mathcal{C}')$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (**);



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$, alors il existe $l_i \in \mathcal{C}$ tel que $I_v(l_i) = 1$ (sinon $I_v(\mathcal{C})$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (*);
 - soit $I_v(p) = 1$, donc $I_v(\neg p) = 0$, alors il existe $l'_j \in \mathcal{C}'$ tel que $I_v(l'_j) = 1$ (sinon $I_v(\mathcal{C}')$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (**);
- donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ pour n'importe quel couple de clauses \mathcal{C} et \mathcal{C}' dont la résolvante est dans $\text{EnsRes}(\mathcal{E})$;



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$, alors il existe $l_i \in \mathcal{C}$ tel que $I_v(l_i) = 1$ (sinon $I_v(\mathcal{C})$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (*);
 - soit $I_v(p) = 1$, donc $I_v(\neg p) = 0$, alors il existe $l'_j \in \mathcal{C}'$ tel que $I_v(l'_j) = 1$ (sinon $I_v(\mathcal{C}')$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (**);
- donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ pour n'importe quel couple de clauses \mathcal{C} et \mathcal{C}' dont la résolvante est dans $\text{EnsRes}(\mathcal{E})$;
- donc $I_v(\text{EnsRes}(\mathcal{E})) = 1$ en tant qu'ensemble de résolvantes vraies ;



Généralisation aux ensembles de résolvantes (preuve propriété (\Rightarrow))

- Si $I_v(\mathcal{E}) = 1$ alors en particulier $I_v(\mathcal{C}) = 1$ et $I_v(\mathcal{C}') = 1$;
 - soit $I_v(p) = 0$, alors il existe $l_i \in \mathcal{C}$ tel que $I_v(l_i) = 1$ (sinon $I_v(\mathcal{C})$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (*);
 - soit $I_v(p) = 1$, donc $I_v(\neg p) = 0$, alors il existe $l'_j \in \mathcal{C}'$ tel que $I_v(l'_j) = 1$ (sinon $I_v(\mathcal{C}')$ serait faux), donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ d'après (**);
- donc $I_v(\text{Res}(\mathcal{C}, \mathcal{C}')) = 1$ pour n'importe quel couple de clauses \mathcal{C} et \mathcal{C}' dont la résolvante est dans $\text{EnsRes}(\mathcal{E})$;
- donc $I_v(\text{EnsRes}(\mathcal{E})) = 1$ en tant qu'ensemble de résolvantes vraies;
- comme par hypothèse, $I_v(\mathcal{E}) = 1$, alors $I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E})) = 1$. □



Ensemble de clauses saturé

- Un ensemble de clauses \mathcal{E} est saturé ssi \mathcal{E} inclut l'ensemble de ses résolvantes ;
 - autrement dit : \mathcal{E} est saturé ssi $\text{EnsRes}(\mathcal{E}) \subseteq \mathcal{E}$;



Ensemble de clauses saturé

- Un ensemble de clauses \mathcal{E} est saturé ssi \mathcal{E} inclut l'ensemble de ses résolvantes ;
 - autrement dit : \mathcal{E} est saturé ssi $\text{EnsRes}(\mathcal{E}) \subseteq \mathcal{E}$;
- saturer un ensemble de clauses \mathcal{E} revient à lui ajouter l'ensemble de ses résolvantes jusqu'à ce que le nouvel ensemble obtenu soit saturé ;



Saturation d'un ensemble de clauses

- soit $\text{Satur}(\mathcal{E})$ la fonction retournant l'ensemble de clauses \mathcal{E} saturé :



Saturation d'un ensemble de clauses

- soit $\text{Satur}(\mathcal{E})$ la fonction retournant l'ensemble de clauses \mathcal{E} saturé :

```
1: function Satur( $\mathcal{E}$ )
2:   while EnsRes( $\mathcal{E}$ )  $\notin \mathcal{E}$  do
3:      $\mathcal{E} \leftarrow \mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ 
4:   end while
5:   return  $\mathcal{E}$ 
6: end function
```



Insatisfiabilité par résolution (principe)

- On souhaite définir une procédure pour définir si une formule est insatisfiable ou non ;
- on part du constat que :
 - 1 une forme clausale \mathcal{E} contient la clause vide $\{\}$ ssi \mathcal{E} est insatisfiable (cf. Diapo 190) ;
 - 2 la clause vide $\{\}$ est produite par résolution (cf. Diapo 196) ;
 - 3 \mathcal{E} et $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ ont les mêmes modèles (cf. Diapo 197) ;



Insatisfiabilité par résolution (algorithme)

- Étant donné un ensemble de clauses \mathcal{E} , déterminer si \mathcal{E} est insatisfiable revient à appliquer l'algorithme suivant :

1: **while** $\{\} \notin \mathcal{E}$ **and** $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$ **do**

2: $\mathcal{E} \leftarrow \mathcal{E} \cup \text{EnsRes}(\mathcal{E})$

3: **end while**

4: **if** $\{\} \in \mathcal{E}$ **then**

5: **print**(" \mathcal{E} insatisfiable")

6: **else**

7: **print**(" \mathcal{E} satisfiable")

8: **end if**



Insatisfiabilité par résolution (exemple)

- Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?



Insatisfiabilité par résolution (exemple)

- Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?
1 : $\{\} \notin \mathcal{E}$,



Insatisfiabilité par résolution (exemple)

- Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?
1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$



Insatisfiabilité par résolution (exemple)

- Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?
1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$ et $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;



Insatisfiabilité par résolution (exemple)

■ Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?

1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$ et $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}\}$



Insatisfiabilité par résolution (exemple)

■ Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?

1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$ et $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}\}$

1 : $\{\} \notin \mathcal{E}$,



Insatisfiabilité par résolution (exemple)

■ Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?

1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$ et $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}\}$

1 : $\{\} \notin \mathcal{E}$,

$\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{\neg q\}, \{p, \neg p\}, \{p\}, \{p, \neg q\}, \{\neg p, q\}\}$



Insatisfiabilité par résolution (exemple)

■ Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?

1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$ et $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}\}$

1 : $\{\} \notin \mathcal{E}$,

$\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{\neg q\}, \{p, \neg p\}, \{p\}, \{p, \neg q\}, \{\neg p, q\}\}$ et
 $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;



Insatisfiabilité par résolution (exemple)

■ Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?

1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$ et $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}\}$

1 : $\{\} \notin \mathcal{E}$,

$\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{\neg q\}, \{p, \neg p\}, \{p\}, \{p, \neg q\}, \{\neg p, q\}\}$ et
 $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{p, \neg p\}\}$



Insatisfiabilité par résolution (exemple)

■ Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?

1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$ et $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}\}$

1 : $\{\} \notin \mathcal{E}$,

$\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{\neg q\}, \{p, \neg p\}, \{p\}, \{p, \neg q\}, \{\neg p, q\}\}$ et
 $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{p, \neg p\}\}$

1 : $\{\} \in \mathcal{E}$



Insatisfiabilité par résolution (exemple)

■ Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?

1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$ et $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}\}$

1 : $\{\} \notin \mathcal{E}$,

$\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{\neg q\}, \{p, \neg p\}, \{p\}, \{p, \neg q\}, \{\neg p, q\}\}$ et
 $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{p, \neg p\}\}$

1 : $\{\} \in \mathcal{E}$

4 : $\{\} \in \mathcal{E}$



Insatisfiabilité par résolution (exemple)

■ Soit $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}\}$. Satisfiable ou insatisfiable ?

1 : $\{\} \notin \mathcal{E}$, $\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}\}$ et $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}\}$

1 : $\{\} \notin \mathcal{E}$,

$\text{EnsRes}(\mathcal{E}) = \{\{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{\neg q\}, \{p, \neg p\}, \{p\}, \{p, \neg q\}, \{\neg p, q\}\}$ et
 $\text{EnsRes}(\mathcal{E}) \notin \mathcal{E}$;

2 : $\mathcal{E} = \{\{p\}, \{\neg q\}, \{p, \neg q\}, \{q, \neg p\}, \{q\}, \{\neg p\}, \{q, \neg q\}, \{\}, \{p, \neg p\}\}$

1 : $\{\} \in \mathcal{E}$

4 : $\{\} \in \mathcal{E}$

5 : \mathcal{E} est insatisfiable



Insatisfiabilité par résolution (remarque pratique)

- Dès que l'algorithme de résolution produit la clause vide {}, la formule est insatisfiable ;



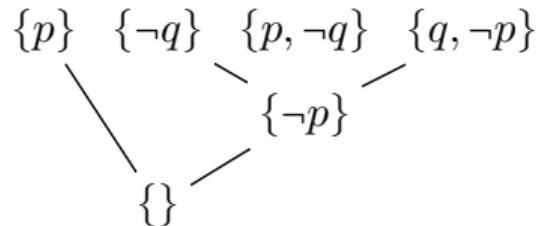
Insatisfiabilité par résolution (remarque pratique)

- Dès que l'algorithme de résolution produit la clause vide {}, la formule est insatisfiable ;
- si \mathcal{E} est « pressentie » insatisfiable, on peut donc chercher à produire {} le plus rapidement possible (même si \mathcal{E} n'est pas saturé) ;



Insatisfiabilité par résolution (remarque pratique)

- Dès que l'algorithme de résolution produit la clause vide {}, la formule est insatisfiable ;
- si \mathcal{E} est « pressentie » insatisfiable, on peut donc chercher à produire {} le plus rapidement possible (même si \mathcal{E} n'est pas saturé) ;
 - exemple précédent :





Insatisfiabilité par résolution (optimisation 1)

- Comme \mathcal{E} est un ensemble au sens mathématique (une seule occurrence de chaque élément),



Insatisfiabilité par résolution (optimisation 1)

- Comme \mathcal{E} est un ensemble au sens mathématique (une seule occurrence de chaque élément),
- $\mathcal{E} \cup \text{EnsRes}(\mathcal{E}) = \mathcal{E} \cup \text{EnsRes}'(\mathcal{E})$ où :
 - $\text{EnsRes}'(\mathcal{E})$ est l'ensemble des résolvantes non encore dans \mathcal{E} ,
 - et $\text{EnsRes}'(\mathcal{E}) = \{\mathcal{C} \in \text{EnsRes}(\mathcal{E}) : \mathcal{C} \notin \mathcal{E}\}$;



Insatisfiabilité par résolution (optimisation 1)

- Comme \mathcal{E} est un ensemble au sens mathématique (une seule occurrence de chaque élément),
- $\mathcal{E} \cup \text{EnsRes}(\mathcal{E}) = \mathcal{E} \cup \text{EnsRes}'(\mathcal{E})$ où :
 - $\text{EnsRes}'(\mathcal{E})$ est l'ensemble des résolvantes non encore dans \mathcal{E} ,
 - et $\text{EnsRes}'(\mathcal{E}) = \{\mathcal{C} \in \text{EnsRes}(\mathcal{E}) : \mathcal{C} \notin \mathcal{E}\}$;
- exemple :
 - si $\mathcal{E} = \{\{p\}, \{q\}, \{\neg q, p\}, \{\neg p, r\}\}$,



Insatisfiabilité par résolution (optimisation 1)

- Comme \mathcal{E} est un ensemble au sens mathématique (une seule occurrence de chaque élément),
- $\mathcal{E} \cup \text{EnsRes}(\mathcal{E}) = \mathcal{E} \cup \text{EnsRes}'(\mathcal{E})$ où :
 - $\text{EnsRes}'(\mathcal{E})$ est l'ensemble des résolvantes non encore dans \mathcal{E} ,
 - et $\text{EnsRes}'(\mathcal{E}) = \{\mathcal{C} \in \text{EnsRes}(\mathcal{E}) : \mathcal{C} \notin \mathcal{E}\}$;
- exemple :
 - si $\mathcal{E} = \{\{p\}, \{q\}, \{\neg q, p\}, \{\neg p, r\}\}$,
 - alors $\text{EnsRes}(\mathcal{E}) = \{\{r\}, \{p\}, \{\neg q, r\}\}$,



Insatisfiabilité par résolution (optimisation 1)

- Comme \mathcal{E} est un ensemble au sens mathématique (une seule occurrence de chaque élément),
- $\mathcal{E} \cup \text{EnsRes}(\mathcal{E}) = \mathcal{E} \cup \text{EnsRes}'(\mathcal{E})$ où :
 - $\text{EnsRes}'(\mathcal{E})$ est l'ensemble des résolvantes non encore dans \mathcal{E} ,
 - et $\text{EnsRes}'(\mathcal{E}) = \{\mathcal{C} \in \text{EnsRes}(\mathcal{E}) : \mathcal{C} \notin \mathcal{E}\}$;
- exemple :
 - si $\mathcal{E} = \{\{p\}, \{q\}, \{\neg q, p\}, \{\neg p, r\}\}$,
 - alors $\text{EnsRes}(\mathcal{E}) = \{\{r\}, \{p\}, \{\neg q, r\}\}$,
 - et $\text{EnsRes}'(\mathcal{E}) = \{\{r\}, \{\neg q, r\}\}$ (car $\{p\} \in \mathcal{E}$).



Insatisfiabilité par résolution (optimisation 2)

- Pour 2 clauses \mathcal{C} et \mathcal{C}' / $\mathcal{C} \subseteq \mathcal{C}'$, si $I_v(\mathcal{C}') = 0$ alors $I_v(\mathcal{C}') = 0$:
 - exemple : si $I_v(p \vee \neg q \vee r) = 0$ alors *a fortiori* $I_v(p \vee \neg q) = 0$,
 - remarque : $\mathcal{C} \subseteq \mathcal{C}'$ s'interprète naturellement si on représente \mathcal{C} et \mathcal{C}' par des ensembles de littéraux.



Insatisfiabilité par résolution (optimisation 2)

- Pour 2 clauses \mathcal{C} et \mathcal{C}' / $\mathcal{C} \subseteq \mathcal{C}'$, si $I_v(\mathcal{C}') = 0$ alors $I_v(\mathcal{C}) = 0$:
 - exemple : si $I_v(p \vee \neg q \vee r) = 0$ alors *a fortiori* $I_v(p \vee \neg q) = 0$,
 - remarque : $\mathcal{C} \subseteq \mathcal{C}'$ s'interprète naturellement si on représente \mathcal{C} et \mathcal{C}' par des ensembles de littéraux.
- Soit \mathcal{E} et $\mathcal{C}, \mathcal{C}' \in \text{EnsRes}(\mathcal{E})$. Alors $\mathcal{E} \cup \{\mathcal{C}, \mathcal{C}'\}$ est insatisfiable **ssi** pour tout valuation v :
 - 1 $I_v(\mathcal{E}) = 0$ ou $I_v(\mathcal{C}) = 0$;



Insatisfiabilité par résolution (optimisation 2)

- Pour 2 clauses \mathcal{C} et \mathcal{C}' / $\mathcal{C} \subseteq \mathcal{C}'$, si $I_v(\mathcal{C}') = 0$ alors $I_v(\mathcal{C}') = 0$:
 - exemple : si $I_v(p \vee \neg q \vee r) = 0$ alors *a fortiori* $I_v(p \vee \neg q) = 0$,
 - remarque : $\mathcal{C} \subseteq \mathcal{C}'$ s'interprète naturellement si on représente \mathcal{C} et \mathcal{C}' par des ensembles de littéraux.
- Soit \mathcal{E} et $\mathcal{C}, \mathcal{C}' \in \text{EnsRes}(\mathcal{E})$. Alors $\mathcal{E} \cup \{\mathcal{C}, \mathcal{C}'\}$ est insatisfiable **ssi** pour tout valuation v :
 - 1 $I_v(\mathcal{E}) = 0$ ou $I_v(\mathcal{C}) = 0$;
 - 2 ou $I_v(\mathcal{C}') = 0$ (qui implique que $I_v(\mathcal{C}) = 0$ quand $\mathcal{C} \subseteq \mathcal{C}'$).



Insatisfiabilité par résolution (optimisation 2)

- Pour 2 clauses \mathcal{C} et \mathcal{C}' / $\mathcal{C} \subseteq \mathcal{C}'$, si $I_v(\mathcal{C}') = 0$ alors $I_v(\mathcal{C}') = 0$:
 - exemple : si $I_v(p \vee \neg q \vee r) = 0$ alors *a fortiori* $I_v(p \vee \neg q) = 0$,
 - remarque : $\mathcal{C} \subseteq \mathcal{C}'$ s'interprète naturellement si on représente \mathcal{C} et \mathcal{C}' par des ensembles de littéraux.
- Soit \mathcal{E} et $\mathcal{C}, \mathcal{C}' \in \text{EnsRes}(\mathcal{E})$. Alors $\mathcal{E} \cup \{\mathcal{C}, \mathcal{C}'\}$ est insatisfiable **ssi** pour tout valuation v :
 - 1 $I_v(\mathcal{E}) = 0$ ou $I_v(\mathcal{C}) = 0$;
 - 2 ou $I_v(\mathcal{C}') = 0$ (qui implique que $I_v(\mathcal{C}) = 0$ quand $\mathcal{C} \subseteq \mathcal{C}'$).
- **Optimisation** : montrer que $\mathcal{E} \cup \{\mathcal{C}, \mathcal{C}'\}$ est insatisfiable quand $\mathcal{C} \subseteq \mathcal{C}'$ peut se ramener à montrer que $\mathcal{E} \cup \{\mathcal{C}\}$ est insatisfiable.



Insatisfiabilité par résolution (exemple avec optimisations)

- Soit $\mathcal{E} = \{\{p, \neg q\}, \{\neg p, r\}, \{p\}, \{q, r\}, \{\neg r\}\}$;



Insatisfiabilité par résolution (exemple avec optimisations)

- Soit $\mathcal{E} = \{\{p, \neg q\}, \{\neg p, r\}, \{p\}, \{q, r\}, \{\neg r\}\}$;
- $\text{EnsRes}(\mathcal{E}) = \{\{\neg q, r\}, \{p, r\}, \{r\}, \{\neg p\}, \{q\}\}$;



Insatisfiabilité par résolution (exemple avec optimisations)

- Soit $\mathcal{E} = \{\{p, \neg q\}, \{\neg p, r\}, \{p\}, \{q, r\}, \{\neg r\}\}$;
- $\text{EnsRes}(\mathcal{E}) = \{\{\neg q, \textcolor{red}{r}\}, \{p, \textcolor{red}{r}\}, \{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;
- simplifié en $\text{EnsRes}'(\mathcal{E}) = \{\{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;



Insatisfiabilité par résolution (exemple avec optimisations)

- Soit $\mathcal{E} = \{\{p, \neg q\}, \{\neg p, r\}, \{p\}, \{q, r\}, \{\neg r\}\}$;
- $\text{EnsRes}(\mathcal{E}) = \{\{\neg q, \textcolor{red}{r}\}, \{p, \textcolor{red}{r}\}, \{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;
- simplifié en $\text{EnsRes}'(\mathcal{E}) = \{\{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;
- on peut vérifier que :
 - si $\{\neg q, r\}$ et $\{p, r\}$ sont (des disjonctions de littéraux) fausses, alors nécessairement $\{r\}$ est faux ;



Insatisfiabilité par résolution (exemple avec optimisations)

- Soit $\mathcal{E} = \{\{p, \neg q\}, \{\neg p, r\}, \{p\}, \{q, r\}, \{\neg r\}\}$;
- $\text{EnsRes}(\mathcal{E}) = \{\{\neg q, \textcolor{red}{r}\}, \{p, \textcolor{red}{r}\}, \{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;
- simplifié en $\text{EnsRes}'(\mathcal{E}) = \{\{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;
- on peut vérifier que :
 - si $\{\neg q, r\}$ et $\{p, r\}$ sont (des disjonctions de littéraux) fausses, alors nécessairement $\{r\}$ est faux ;
 - si $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ est insatisfiable, alors $\mathcal{E} \cup \text{EnsRes}'(\mathcal{E})$ l'est aussi :



Insatisfiabilité par résolution (exemple avec optimisations)

- Soit $\mathcal{E} = \{\{p, \neg q\}, \{\neg p, r\}, \{p\}, \{q, r\}, \{\neg r\}\}$;
- $\text{EnsRes}(\mathcal{E}) = \{\{\neg q, \textcolor{red}{r}\}, \{p, \textcolor{red}{r}\}, \{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;
- simplifié en $\text{EnsRes}'(\mathcal{E}) = \{\{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;
- on peut vérifier que :
 - si $\{\neg q, r\}$ et $\{p, r\}$ sont (des disjonctions de littéraux) fausses, alors nécessairement $\{r\}$ est faux ;
 - si $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ est insatisfiable, alors $\mathcal{E} \cup \text{EnsRes}'(\mathcal{E})$ l'est aussi :
 - soit c'est \mathcal{E} qui est insatisfiable ;



Insatisfiabilité par résolution (exemple avec optimisations)

- Soit $\mathcal{E} = \{\{p, \neg q\}, \{\neg p, r\}, \{p\}, \{q, r\}, \{\neg r\}\}$;
- $\text{EnsRes}(\mathcal{E}) = \{\{\neg q, \textcolor{red}{r}\}, \{p, \textcolor{red}{r}\}, \{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;
- simplifié en $\text{EnsRes}'(\mathcal{E}) = \{\{\textcolor{red}{r}\}, \{\neg p\}, \{q\}\}$;
- on peut vérifier que :
 - si $\{\neg q, r\}$ et $\{p, r\}$ sont (des disjonctions de littéraux) fausses, alors nécessairement $\{r\}$ est faux ;
 - si $\mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ est insatisfiable, alors $\mathcal{E} \cup \text{EnsRes}'(\mathcal{E})$ l'est aussi :
 - soit c'est \mathcal{E} qui est insatisfiable ;
 - soit c'est $\text{EnsRes}(\mathcal{E})$, ce qui entraîne que $\text{EnsRes}'(\mathcal{E})$ aussi.



Extraction d'un modèle (idées de départ)

- Extraire les modèles de \mathcal{E} revient à trouver toutes les valuations v telles que $I_v(\mathcal{E}) = 1$;



Extraction d'un modèle (idées de départ)

- Extraire les modèles de \mathcal{E} revient à trouver toutes les valuations v telles que $I_v(\mathcal{E}) = 1$;
- idée : affecter **successivement** à **toutes** les variables propositionnelles de \mathcal{E} la valeur \perp puis \top et regarder si :
 - la formule est vrai ➔ on a trouvé un modèle,
 - la formule est fausse ➔ on a trouvé un contre-modèle.



Extraction d'un modèle (principe)

- Soit \mathcal{E} une forme clausale saturée et p une variable de \mathcal{E} :



Extraction d'un modèle (principe)

- Soit \mathcal{E} une forme clausale saturée et p une variable de \mathcal{E} :
 - $\mathcal{E}_{[\perp/p]}$: la variable p est supposée fausse partout dans \mathcal{E} ,
 - $\mathcal{E}_{[\top/p]}$: la variable p est supposée vraie partout dans \mathcal{E} ;



Extraction d'un modèle (principe)

- Soit \mathcal{E} une forme clausale saturée et p une variable de \mathcal{E} :
 - $\mathcal{E}_{[\perp/p]}$: la variable p est supposée fausse partout dans \mathcal{E} ,
 - $\mathcal{E}_{[\top/p]}$: la variable p est supposée vraie partout dans \mathcal{E} ;
- **propriété** : si \mathcal{E} est saturé, alors $\mathcal{E}_{[\perp/p]}$ et $\mathcal{E}_{[\top/p]}$ aussi
 - preuve : à faire à titre d'exercice ;



Extraction d'un modèle (principe)

- Soit \mathcal{E} une forme clausale saturée et p une variable de \mathcal{E} :
 - $\mathcal{E}_{[\perp/p]}$: la variable p est supposée fausse partout dans \mathcal{E} ,
 - $\mathcal{E}_{[\top/p]}$: la variable p est supposée vraie partout dans \mathcal{E} ;
- propriété : si \mathcal{E} est saturé, alors $\mathcal{E}_{[\perp/p]}$ et $\mathcal{E}_{[\top/p]}$ aussi
 - preuve : à faire à titre d'exercice ;
- en itérant la construction, on sait que si on trouve :
 - un ensemble vide de clauses $\{\}$, les substitutions successives constituent un modèle de \mathcal{E} ,
 - un ensemble $\{\{\}, \dots\}$ de clauses contenant la clause vide $\{\}$, les substitutions successives constituent un contre-modèle de \mathcal{E} .



Extraction d'un modèle (remarque)

- Supposer que :
 - « la variable p est fausse partout dans \mathcal{E} » ($\mathcal{E}_{[\perp/p]}$) signifie que :
 - la valuation v en cours de construction est telle que $v(p) = 0$;



Extraction d'un modèle (remarque)

- Supposer que :
 - « la variable p est **fausse** partout dans \mathcal{E} » ($\mathcal{E}_{[\perp/p]}$)
signifie que :
 - la valuation v en cours de construction est telle que $v(p) = 0$;
- de même, supposer que :
 - « la variable p est **vraie** partout dans \mathcal{E} » ($\mathcal{E}_{[\top/p]}$)
signifie que :
 - la valuation v en cours de construction est telle que $v(p) = 1$.



Extraction d'un modèle (substitution d'une variable)

- $\mathcal{E}_{[\perp/p]}$ est l'ensemble de clauses obtenu en (*) :
 - éliminant p de toutes les clauses de \mathcal{E} ;
 - éliminant toute clause contenant $\neg p$;

(*) cf. Diapo 215 pour la justification.



Extraction d'un modèle (substitution d'une variable)

- $\mathcal{E}_{[\perp/p]}$ est l'ensemble de clauses obtenu en (*) :
 - éliminant p de toutes les clauses de \mathcal{E} ;
 - éliminant toute clause contenant $\neg p$;
- $\mathcal{E}_{[\top/p]}$ est l'ensemble de clauses obtenu en (*) :
 - éliminant toute clause contenant p ;
 - éliminant le littéral $\neg p$ de toutes les clauses de \mathcal{E} .

(*) cf. Diapo 215 pour la justification.



Extraction d'un modèle (exemples de substitution)

- Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]}$



Extraction d'un modèle (exemples de substitution)

- Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$



Extraction d'un modèle (exemples de substitution)

■ Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\textcolor{brown}{\perp}/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\textcolor{brown}{\top}/p]}$



Extraction d'un modèle (exemples de substitution)

■ Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$



Extraction d'un modèle (exemples de substitution)

- Exemples :
 - $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
 - $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$
- justification de ces simplifications :



Extraction d'un modèle (exemples de substitution)

- Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$

- justification de ces simplifications :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]}$



Extraction d'un modèle (exemples de substitution)

■ Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$

■ justification de ces simplifications :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]}$ représente $(\perp \vee \neg q) \wedge (\top \vee r) \wedge (\perp \vee r)$



Extraction d'un modèle (exemples de substitution)

■ Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$

■ justification de ces simplifications :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]}$ représente $(\perp \vee \neg q) \wedge (\top \vee r) \wedge (\perp \vee r)$
 $\equiv (\neg q) \wedge (\top) \wedge (r)$



Extraction d'un modèle (exemples de substitution)

■ Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$

■ justification de ces simplifications :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]}$ représente $(\perp \vee \neg q) \wedge (\top \vee r) \wedge (\perp \vee r)$
 $\equiv (\neg q) \wedge (\top) \wedge (r) \equiv \neg q \wedge r$ (cf. Diapo 160);



Extraction d'un modèle (exemples de substitution)

■ Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$

■ justification de ces simplifications :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]}$ représente $(\perp \vee \neg q) \wedge (\top \vee r) \wedge (\perp \vee r)$
 $\equiv (\neg q) \wedge (\top) \wedge (r) \equiv \neg q \wedge r$ (cf. Diapo 160);
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]}$



Extraction d'un modèle (exemples de substitution)

■ Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$

■ justification de ces simplifications :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]}$ représente $(\perp \vee \neg q) \wedge (\top \vee r) \wedge (\perp \vee r)$
 $\equiv (\neg q) \wedge (\top) \wedge (r) \equiv \neg q \wedge r$ (cf. Diapo 160);
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]}$ représente $(\top \vee \neg q) \wedge (\perp \vee r) \wedge (\top \vee r)$



Extraction d'un modèle (exemples de substitution)

■ Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$

■ justification de ces simplifications :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]}$ représente $(\perp \vee \neg q) \wedge (\top \vee r) \wedge (\perp \vee r)$
 $\equiv (\neg q) \wedge (\top) \wedge (r) \equiv \neg q \wedge r$ (cf. Diapo 160);
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]}$ représente $(\top \vee \neg q) \wedge (\perp \vee r) \wedge (\top \vee r)$
 $\equiv (\top) \wedge (r) \wedge (\top)$



Extraction d'un modèle (exemples de substitution)

■ Exemples :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]} = \{\{\neg q\}, \{r\}\};$
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]} = \{\{r\}\};$

■ justification de ces simplifications :

- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\perp/p]}$ représente $(\perp \vee \neg q) \wedge (\top \vee r) \wedge (\perp \vee r)$
 $\equiv (\neg q) \wedge (\top) \wedge (r) \equiv \neg q \wedge r$ (cf. Diapo 160);
- $\{\{p, \neg q\}, \{\neg p, r\}, \{p, r\}\}_{[\top/p]}$ représente $(\top \vee \neg q) \wedge (\perp \vee r) \wedge (\top \vee r)$
 $\equiv (\top) \wedge (r) \wedge (\top) \equiv r$ (cf. Diapo 160);



Extraction d'un modèle (exemple de construction)

- soit $\mathcal{E} = \{\{p, \neg q\}, \{q, \neg r\}, \{p, r\}, \{p, \neg r\}, \{p, q\}, \{p\}\}$ (saturé) ;



Extraction d'un modèle (exemple de construction)

- soit $\mathcal{E} = \{\{p, \neg q\}, \{q, \neg r\}, \{p, r\}, \{p, \neg r\}, \{p, q\}, \{p\}\}$ (saturé) ;
- comme \mathcal{E} contient 3 variables, il y a **8 valuations** possibles ;



Extraction d'un modèle (exemple de construction)

- soit $\mathcal{E} = \{\{p, \neg q\}, \{q, \neg r\}, \{p, r\}, \{p, \neg r\}, \{p, q\}, \{p\}\}$ (saturé) ;
- comme \mathcal{E} contient 3 variables, il y a **8 valuations** possibles ;
 1. $\mathcal{E}_{[\perp/p]} = \{\{\neg q\}, \{q, \neg r\}, \{r\}, \{\neg r\}, \{q\}, \{\}\}$ n'a pas de modèle ;



Extraction d'un modèle (exemple de construction)

- soit $\mathcal{E} = \{\{p, \neg q\}, \{q, \neg r\}, \{p, r\}, \{p, \neg r\}, \{p, q\}, \{p\}\}$ (saturé) ;
 - comme \mathcal{E} contient 3 variables, il y a 8 valuations possibles ;
1. $\mathcal{E}_{[\perp/p]} = \{\{\neg q\}, \{q, \neg r\}, \{r\}, \{\neg r\}, \{q\}, \{\}\}$ n'a pas de modèle ;
➡ on vient de trouver 4 contre-modèles !

$$I_{v_1}(\mathcal{E}) = I_{v_2}(\mathcal{E}) = I_{v_3}(\mathcal{E}) = I_{v_4}(\mathcal{E}) = 0 \text{ pour } \begin{cases} v_1(p) = 0, v_1(q) = 0, v_1(r) = 0 \\ v_2(p) = 0, v_2(q) = 0, v_2(r) = 1 \\ v_3(p) = 0, v_3(q) = 1, v_3(r) = 0 \\ v_4(p) = 0, v_4(q) = 1, v_4(r) = 1 \end{cases}$$

car \mathcal{E} est faux dès que p est faux, indépendamment des valeurs de vérité de q et de r (4 valuations possibles).



Extraction d'un modèle (exemple de construction, suite)

2. $\mathcal{E}_{[\textcolor{red}{T}/p]} = \{\{q, \neg r\}\}$



Extraction d'un modèle (exemple de construction, suite)

2. $\mathcal{E}_{[\textcolor{red}{T}/p]} = \{\{q, \neg r\}\}$

2.1. $\{\{q, \neg r\}\}_{[\textcolor{brown}{\perp}/q]} = \{\{\neg r\}\}$



Extraction d'un modèle (exemple de construction, suite)

2. $\mathcal{E}_{[\top/p]} = \{\{q, \neg r\}\}$

2.1. $\{\{q, \neg r\}\}_{[\perp/q]} = \{\{\neg r\}\}$

2.1.1. $\{\{\neg r\}\}_{[\perp/r]} = \{\}$ est valide, donc on vient de trouver **1 modèle** :

► soit v_5 telle que $v_5(p) = 1$, $v_5(q) = 0$, $v_5(r) = 0$; alors $I_{v_5}(\mathcal{E}) = 1$;



Extraction d'un modèle (exemple de construction, suite)

2. $\mathcal{E}_{[\top/p]} = \{\{q, \neg r\}\}$

2.1. $\{\{q, \neg r\}\}_{[\perp/q]} = \{\{\neg r\}\}$

2.1.1. $\{\{\neg r\}\}_{[\perp/r]} = \{\{\}\}$ est valide, donc on vient de trouver **1 modèle** :

► soit v_5 telle que $v_5(p) = 1$, $v_5(q) = 0$, $v_5(r) = 0$; alors $I_{v_5}(\mathcal{E}) = 1$;

2.1.2. $\{\{\neg r\}\}_{[\top/r]} = \{\{\{\}\}\}$ est insatisfiable, donc on vient de trouver **1 contre-modèle** :

► soit v_6 telle que $v_6(p) = 1$, $v_6(q) = 0$, $v_6(r) = 1$; alors $I_{v_6}(\mathcal{E}) = 0$.



Extraction d'un modèle (exemple de construction, fin)

2.2. $\{\{q, \neg r\}\}_{[\top/q]} = \{\}$ est valide ;

donc on vient de trouver **2 modèles** puisque la validité de \mathcal{E} est indépendante^(*) de la valuation de r

$$I_{v_7}(\mathcal{E}) = I_{v_8}(\mathcal{E}) = 1 \text{ pour } \begin{cases} v_7(p) = 1, v_7(q) = 1, v_7(r) = 0 \\ v_8(p) = 1, v_8(q) = 1, v_8(r) = 1 \end{cases}$$

^(*) « indépendante » signifie « pour toute valuation de r ».



Retour sur l'algorithme de test de satisfiabilité d'une formule

- Pour rappel, l'algorithme (cf. Diapo 205) était le suivant :

- Déterminer si un ensemble de clauses \mathcal{E} est insatisfiable revient à appliquer l'algorithme suivant :

```
1: while {}  $\notin \mathcal{E}$  and EnsRes( $\mathcal{E}$ )  $\notin \mathcal{E}$  do
2:    $\mathcal{E} \leftarrow \mathcal{E} \cup \text{EnsRes}(\mathcal{E})$ 
3: end while
4: if {}  $\in \mathcal{E}$  then
5:   print(" $\mathcal{E}$  insatisfiable")
6: else
7:   print(" $\mathcal{E}$  satisfiable")
8: end if
```



Propriétés de l'algorithme d'insatisfiabilité (objectifs)

- On note :
 - $\mathcal{E}^{(0)}$ la forme clausale en entrée de l'algorithme,
 - $\mathcal{E}^{(n)}$ la forme clausale après $n \geq 0$ appels à EnsRes ;
- étant donné $\mathcal{E}^{(0)}$, on souhaite démontrer que pour $n \geq 0$:
 - 1 si l'algorithme produit $\{\} \in \mathcal{E}^{(n)}$ alors $\mathcal{E}^{(0)}$ est insatisfiable,
(correction ou adéquation)



Propriétés de l'algorithme d'insatisfiabilité (objectifs)

- On note :
 - $\mathcal{E}^{(0)}$ la forme clausale en entrée de l'algorithme,
 - $\mathcal{E}^{(n)}$ la forme clausale après $n \geq 0$ appels à EnsRes ;
- étant donné $\mathcal{E}^{(0)}$, on souhaite démontrer que pour $n \geq 0$:
 - 1 si l'algorithme produit $\{\} \in \mathcal{E}^{(n)}$ alors $\mathcal{E}^{(0)}$ est insatisfiable,
(correction ou adéquation)
 - 2 si l'algorithme produit $\mathcal{E}^{(n)} = \text{Satur}(\mathcal{E}^{(0)})$ avec $\{\} \notin \mathcal{E}^{(n)}$
alors $\mathcal{E}^{(0)}$ est satisfiable.
(complétude)



Propriétés de l'algorithme d'insatisfiabilité (adéquation)

- D'après ce qui précède (cf. Diapo 190 et 197 respect.) :

si $\{\} \in \mathcal{E}^{(n)}$ pour $n \geq 0$, $\mathcal{E}^{(n)}$ est insatisfiable, (R₁)

pour toute valuation v , $I_v(\mathcal{E}) = I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E}))$. (R₂)



Propriétés de l'algorithme d'insatisfiabilité (adéquation)

- D'après ce qui précède (cf. Diapo 190 et 197 respect.) :

si $\{\} \in \mathcal{E}^{(n)}$ pour $n \geq 0$, $\mathcal{E}^{(n)}$ est insatisfiable, (R₁)

pour toute valuation v , $I_v(\mathcal{E}) = I_v(\mathcal{E} \cup \text{EnsRes}(\mathcal{E}))$. (R₂)

- Démarche pour la démonstration de l'adéquation :

- induction sur le nombre $n \geq 0$ d'appels à la fonction $\text{EnsRes}()$,
- on note « unsat » pour « insatisfiable ».



Propriétés de l'algorithme d'insatisfiabilité (adéquation)

- cas de base ($n = 0$) : si $\{\} \in \mathcal{E}^{(0)}$, $\mathcal{E}^{(0)}$ unsat ?



Propriétés de l'algorithme d'insatisfiabilité (adéquation)

- cas de base ($n = 0$) : si $\{\} \in \mathcal{E}^{(0)}$, $\mathcal{E}^{(0)}$ unsat ? Trivial d'après (R_1) ;



Propriétés de l'algorithme d'insatisfiabilité (adéquation)

- cas de base ($n = 0$) : si $\{\} \in \mathcal{E}^{(0)}$, $\mathcal{E}^{(0)}$ unsat ? Trivial d'après (R_1) ;
- hypothèse d'induction (qu'on écrit en 2 temps) :
 - si $\{\} \in \mathcal{E}^{(n)}$, $\mathcal{E}^{(n)}$ est unsat d'après (R_1), et on suppose que si $\mathcal{E}^{(n)}$ est unsat alors $\mathcal{E}^{(0)}$ est unsat ; (H_1)



Propriétés de l'algorithme d'insatisfiabilité (adéquation)

- **cas de base** ($n = 0$) : si $\{\} \in \mathcal{E}^{(0)}$, $\mathcal{E}^{(0)}$ unsat ? Trivial d'après (R_1) ;
- **hypothèse d'induction** (qu'on écrit en 2 temps) :
 - si $\{\} \in \mathcal{E}^{(n)}$, $\mathcal{E}^{(n)}$ est unsat d'après (R_1), et on suppose que si $\mathcal{E}^{(n)}$ est unsat alors $\mathcal{E}^{(0)}$ est unsat ; (HI)
- **héritéité** de n à $n + 1$ (qu'on démontre en 2 temps) :
 - si $\{\} \in \mathcal{E}^{(n+1)}$, $\mathcal{E}^{(n+1)}$ est unsat d'après (R_1),
 - il reste à démontrer : si $\mathcal{E}^{(n+1)}$ unsat, alors $\mathcal{E}^{(0)}$ est unsat



Propriétés de l'algorithme d'insatisfiabilité (adéquation)

- cas de base ($n = 0$) : si $\{\} \in \mathcal{E}^{(0)}$, $\mathcal{E}^{(0)}$ unsat ? Trivial d'après (R_1) ;
- hypothèse d'induction (qu'on écrit en 2 temps) :
 - si $\{\} \in \mathcal{E}^{(n)}$, $\mathcal{E}^{(n)}$ est unsat d'après (R_1), et on suppose que si $\mathcal{E}^{(n)}$ est unsat alors $\mathcal{E}^{(0)}$ est unsat ; (HI)
- hérédité de n à $n + 1$ (qu'on démontre en 2 temps) :
 - si $\{\} \in \mathcal{E}^{(n+1)}$, $\mathcal{E}^{(n+1)}$ est unsat d'après (R_1),
 - il reste à démontrer : si $\mathcal{E}^{(n+1)}$ unsat, alors $\mathcal{E}^{(0)}$ est unsat
 - preuve : $\mathcal{E}^{(n+1)}$ unsatssi $\mathcal{E}^{(n)} \cup \text{EnsRes}(\mathcal{E}^{(n)})$ unsat (déf. de $\mathcal{E}^{(n+1)}$)



Propriétés de l'algorithme d'insatisfiabilité (adéquation)



Propriétés de l'algorithme d'insatisfiabilité (adéquation)

- **cas de base** ($n = 0$) : si $\{\} \in \mathcal{E}^{(0)}$, $\mathcal{E}^{(0)}$ unsat ? Trivial d'après (R_1) ;
- **hypothèse d'induction** (qu'on écrit en 2 temps) :
 - si $\{\} \in \mathcal{E}^{(n)}$, $\mathcal{E}^{(n)}$ est unsat d'après (R_1), et on suppose que si $\mathcal{E}^{(n)}$ est unsat alors $\mathcal{E}^{(0)}$ est unsat ; (HI)
- **héritéité** de n à $n + 1$ (qu'on démontre en 2 temps) :
 - si $\{\} \in \mathcal{E}^{(n+1)}$, $\mathcal{E}^{(n+1)}$ est unsat d'après (R_1),
 - il reste à démontrer : si $\mathcal{E}^{(n+1)}$ unsat, alors $\mathcal{E}^{(0)}$ est unsat
 - **preuve** : $\mathcal{E}^{(n+1)}$ unsat **ssi** $\mathcal{E}^{(n)} \cup \text{EnsRes}(\mathcal{E}^{(n)})$ unsat (déf. de $\mathcal{E}^{(n+1)}$)
 - ssi** $\mathcal{E}^{(n)}$ unsat (d'après (R_2))
 - ssi** $\mathcal{E}^{(0)}$ unsat (d'après (HI))



Propriétés de l'algorithme d'insatisfiabilité (adéquation)

- **cas de base** ($n = 0$) : si $\{\} \in \mathcal{E}^{(0)}$, $\mathcal{E}^{(0)}$ unsat ? Trivial d'après (R_1) ;
- **hypothèse d'induction** (qu'on écrit en 2 temps) :
 - si $\{\} \in \mathcal{E}^{(n)}$, $\mathcal{E}^{(n)}$ est unsat d'après (R_1), et on suppose que si $\mathcal{E}^{(n)}$ est unsat alors $\mathcal{E}^{(0)}$ est unsat ; (HI)
- **héritéité** de n à $n + 1$ (qu'on démontre en 2 temps) :
 - si $\{\} \in \mathcal{E}^{(n+1)}$, $\mathcal{E}^{(n+1)}$ est unsat d'après (R_1),
 - il reste à démontrer : si $\mathcal{E}^{(n+1)}$ unsat, alors $\mathcal{E}^{(0)}$ est unsat
 - **preuve** : $\mathcal{E}^{(n+1)}$ unsat **ssi** $\mathcal{E}^{(n)} \cup \text{EnsRes}(\mathcal{E}^{(n)})$ unsat (déf. de $\mathcal{E}^{(n+1)}$)
 - ssi** $\mathcal{E}^{(n)}$ unsat (d'après (R_2))
 - ssi** $\mathcal{E}^{(0)}$ unsat (d'après (HI))
- **conclusion** : pour tout $n \geq 0$, si $\{\} \in \mathcal{E}^{(n)}$ alors $\mathcal{E}^{(0)}$ est unsat. □



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- À démontrer : si l'algorithme produit $\mathcal{E}^{(n)} = \text{Satur}(\mathcal{E}^{(0)})$ et $\{\} \notin \mathcal{E}^{(n)}$, alors c'est que $\mathcal{E}^{(0)}$ est satisfiable ;



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- À démontrer : si l'algorithme produit $\mathcal{E}^{(n)} = \text{Satur}(\mathcal{E}^{(0)})$ et $\{\} \notin \mathcal{E}^{(n)}$, alors c'est que $\mathcal{E}^{(0)}$ est satisfiable ;
- on note \mathcal{E} la forme clausale d'une formule telle que \mathcal{E} est saturée et ne contient pas la clause vide ;



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- À démontrer : si l'algorithme produit $\mathcal{E}^{(n)} = \text{Satur}(\mathcal{E}^{(0)})$ et $\{\} \notin \mathcal{E}^{(n)}$, alors c'est que $\mathcal{E}^{(0)}$ est satisfiable ;
- on note \mathcal{E} la forme clausale d'une formule telle que \mathcal{E} est **saturée** et ne contient **pas** la clause vide ;
- on va montrer cette propriété :
 - en utilisant une démonstration par induction sur le nombre n de variables propositionnelles p_1, \dots, p_n apparaissant dans \mathcal{E} ,
 - et en montrant qu'on arrive à extraire au moins un modèle de \mathcal{E} (au sens de l'extraction de modèles précédente) ;



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- À démontrer : si l'algorithme produit $\mathcal{E}^{(n)} = \text{Satur}(\mathcal{E}^{(0)})$ et $\{\} \notin \mathcal{E}^{(n)}$, alors c'est que $\mathcal{E}^{(0)}$ est satisfiable ;
- on note \mathcal{E} la forme clausale d'une formule telle que \mathcal{E} est **saturée** et ne contient **pas** la clause vide ;
- on va montrer cette propriété :
 - en utilisant une démonstration par induction sur le nombre n de variables propositionnelles p_1, \dots, p_n apparaissant dans \mathcal{E} ,
 - et en montrant qu'on arrive à extraire au moins un modèle de \mathcal{E} (au sens de l'extraction de modèles précédente) ;
- on écrira « sat » pour « satisfiable ».



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- **base** ($n = 0$) : \mathcal{E} ne contient aucune variable propositionnelle :



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- **base** ($n = 0$) : \mathcal{E} ne contient aucune variable propositionnelle :
 - soit $\mathcal{E} = \{\}$ donc \mathcal{E} est valide (par définition, cf. Diapo 191),
 - soit $\mathcal{E} = \{\{\}\}$ ce qui est impossible puisque $\{\} \notin \mathcal{E}$;



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- **base** ($n = 0$) : \mathcal{E} ne contient aucune variable propositionnelle :
 - soit $\mathcal{E} = \{\}$ donc \mathcal{E} est valide (par définition, cf. Diapo 191),
 - soit $\mathcal{E} = \{\{\}\}$ ce qui est impossible puisque $\{\} \notin \mathcal{E}$;
- **(HI)** : si \mathcal{E} est construit à partir de n variables p_1, \dots, p_n tel que \mathcal{E} est saturé et $\{\} \notin \mathcal{E}$, alors on peut extraire au moins un modèle de \mathcal{E} ;
- **Hérédité** : est-ce que si \mathcal{E} est construit à partir de $n + 1$ variables p_1, \dots, p_n, p_{n+1} tel que \mathcal{E} est saturé et $\{\} \notin \mathcal{E}$, alors on peut extraire au moins un modèle de \mathcal{E} ?



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- **base** ($n = 0$) : \mathcal{E} ne contient aucune variable propositionnelle :
 - soit $\mathcal{E} = \{\}$ donc \mathcal{E} est valide (par définition, cf. Diapo 191),
 - soit $\mathcal{E} = \{\{\}\}$ ce qui est impossible puisque $\{\} \notin \mathcal{E}$;
- **(HI)** : si \mathcal{E} est construit à partir de n variables p_1, \dots, p_n tel que \mathcal{E} est saturé et $\{\} \notin \mathcal{E}$, alors on peut extraire au moins un modèle de \mathcal{E} ;
- **Hérédité** : est-ce que si \mathcal{E} est construit à partir de $n + 1$ variables p_1, \dots, p_n, p_{n+1} tel que \mathcal{E} est saturé et $\{\} \notin \mathcal{E}$, alors on peut extraire au moins un modèle de \mathcal{E} ?
 - ▶ à démontrer!



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- On cherche à extraire un modèle de \mathcal{E} (cf. Diapo 212) :



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- On cherche à extraire un modèle de \mathcal{E} (cf. Diapo 212) :
 - 1 on construit \mathcal{E}' correspondant à $\mathcal{E}_{[\perp/p_{n+1}]}$ (qui est saturé). Alors :
 - 2 on construit \mathcal{E}'' correspondant à $\mathcal{E}_{[\top/p_{n+1}]}$ (qui est saturé). Alors :



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- On cherche à extraire un modèle de \mathcal{E} (cf. Diapo 212) :
 - ① on construit \mathcal{E}' correspondant à $\mathcal{E}_{[\perp/p_{n+1}]}$ (**qui est saturé**). Alors :
 - si la clause vide $\{\} \in \mathcal{E}'$ alors \mathcal{E}' n'a pas de modèle ;
 - ② on construit \mathcal{E}'' correspondant à $\mathcal{E}_{[\top/p_{n+1}]}$ (**qui est saturé**). Alors :



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- On cherche à extraire un modèle de \mathcal{E} (cf. Diapo 212) :
 - 1 on construit \mathcal{E}' correspondant à $\mathcal{E}_{[\perp/p_{n+1}]}$ (qui est saturé). Alors :
 - si la clause vide $\{\} \in \mathcal{E}'$ alors \mathcal{E}' n'a pas de modèle ;
 - si la clause vide $\{\} \notin \mathcal{E}'$: alors \mathcal{E}' ne contient plus que n variables qui a un modèle v d'après (HI) ;
 - 2 on construit \mathcal{E}'' correspondant à $\mathcal{E}_{[\top/p_{n+1}]}$ (qui est saturé). Alors :



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- On cherche à extraire un modèle de \mathcal{E} (cf. Diapo 212) :
 - 1 on construit \mathcal{E}' correspondant à $\mathcal{E}_{[\perp/p_{n+1}]}$ (qui est saturé). Alors :
 - si la clause vide $\{\} \in \mathcal{E}'$ alors \mathcal{E}' n'a pas de modèle ;
 - si la clause vide $\{\} \notin \mathcal{E}'$: alors \mathcal{E}' ne contient plus que n variables qui a un modèle v d'après (HI) ; il suffit de définir une valuation v' telle que : $v'(p_{n+1}) = 0$, et $v'(p_i) = v(p_i)$ pour $0 \leq i \leq n$; alors $I_{v'}(\mathcal{E}') = 1$;
 - 2 on construit \mathcal{E}'' correspondant à $\mathcal{E}_{[\top/p_{n+1}]}$ (qui est saturé). Alors :



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- On cherche à extraire un modèle de \mathcal{E} (cf. Diapo 212) :
 - 1 on construit \mathcal{E}' correspondant à $\mathcal{E}_{[\perp/p_{n+1}]}$ (**qui est saturé**). Alors :
 - si la clause vide $\{\} \in \mathcal{E}'$ alors \mathcal{E}' n'a pas de modèle ;
 - si la clause vide $\{\} \notin \mathcal{E}'$: alors \mathcal{E}' ne contient plus que n variables qui a un modèle v d'après (HI) ; il suffit de définir une valuation v' telle que : $v'(p_{n+1}) = 0$, et $v'(p_i) = v(p_i)$ pour $0 \leq i \leq n$; alors $I_{v'}(\mathcal{E}') = 1$;
 - 2 on construit \mathcal{E}'' correspondant à $\mathcal{E}_{[\top/p_{n+1}]}$ (**qui est saturé**). Alors :
 - si la clause vide $\{\} \in \mathcal{E}''$ alors \mathcal{E}'' n'a pas de modèle ;



Propriétés de l'algorithme d'insatisfiabilité (complétude)

- On cherche à extraire un modèle de \mathcal{E} (cf. Diapo 212) :
 - 1 on construit \mathcal{E}' correspondant à $\mathcal{E}_{[\perp/p_{n+1}]}$ (**qui est saturé**). Alors :
 - si la clause vide $\{\} \in \mathcal{E}'$ alors \mathcal{E}' n'a pas de modèle ;
 - si la clause vide $\{\} \notin \mathcal{E}'$: alors \mathcal{E}' ne contient plus que n variables qui a un modèle v d'après (HI) ; il suffit de définir une valuation v' telle que : $v'(p_{n+1}) = 0$, et $v'(p_i) = v(p_i)$ pour $0 \leq i \leq n$; alors $I_{v'}(\mathcal{E}') = 1$;
 - 2 on construit \mathcal{E}'' correspondant à $\mathcal{E}_{[\top/p_{n+1}]}$ (**qui est saturé**). Alors :
 - si la clause vide $\{\} \in \mathcal{E}''$ alors \mathcal{E}'' n'a pas de modèle ;
 - si la clause vide $\{\} \notin \mathcal{E}''$: alors il existe une valuation v'' telle que : $v''(p_{n+1}) = 1$, et $v''(p_i) = v(p_i)$ pour $0 \leq i \leq n$ d'après (HI) ; alors $I_{v''}(\mathcal{E}'') = 1$.



Propriétés de l'algorithme d'insatisfiabilité (complétude)

■ Remarque :

■ si $\{\} \in \mathcal{E}'$ cela provient du fait que $\{p_{n+1}\} \in \mathcal{E}'$,



Propriétés de l'algorithme d'insatisfiabilité (complétude)

■ Remarque :

- si $\{\} \in \mathcal{E}'$ cela provient du fait que $\{p_{n+1}\} \in \mathcal{E}'$,
- si $\{\} \in \mathcal{E}''$ alors c'est que $\{\neg p_{n+1}\} \in \mathcal{E}''$,



Propriétés de l'algorithme d'insatisfiabilité (complétude)

■ Remarque :

- si $\{\} \in \mathcal{E}'$ cela provient du fait que $\{p_{n+1}\} \in \mathcal{E}'$,
- si $\{\} \in \mathcal{E}''$ alors c'est que $\{\neg p_{n+1}\} \in \mathcal{E}''$,
- donc :
 - si $\{\} \in \mathcal{E}'$ et $\{\} \in \mathcal{E}''$ simultanément,
 - alors $\{\{p_{n+1}\}, \{\neg p_{n+1}\}\} \subseteq \mathcal{E}$, ce qui est impossible puisque \mathcal{E} saturé;



Propriétés de l'algorithme d'insatisfiabilité (complétude)

■ Remarque :

- si $\{\} \in \mathcal{E}'$ cela provient du fait que $\{p_{n+1}\} \in \mathcal{E}'$,
- si $\{\} \in \mathcal{E}''$ alors c'est que $\{\neg p_{n+1}\} \in \mathcal{E}''$,
- donc :
 - si $\{\} \in \mathcal{E}'$ et $\{\} \in \mathcal{E}''$ simultanément,
 - alors $\{\{p_{n+1}\}, \{\neg p_{n+1}\}\} \subseteq \mathcal{E}$, ce qui est impossible puisque \mathcal{E} saturé;

■ Conséquence :

- si $\{\} \in \mathcal{E}'$ alors $\{\} \notin \mathcal{E}''$ (et on peut construire un modèle de \mathcal{E}''),



Propriétés de l'algorithme d'insatisfiabilité (complétude)

■ Remarque :

- si $\{\} \in \mathcal{E}'$ cela provient du fait que $\{p_{n+1}\} \in \mathcal{E}'$,
- si $\{\} \in \mathcal{E}''$ alors c'est que $\{\neg p_{n+1}\} \in \mathcal{E}''$,
- donc :
 - si $\{\} \in \mathcal{E}'$ et $\{\} \in \mathcal{E}''$ simultanément,
 - alors $\{\{p_{n+1}\}, \{\neg p_{n+1}\}\} \subseteq \mathcal{E}$, ce qui est impossible puisque \mathcal{E} saturé;

■ Conséquence :

- si $\{\} \in \mathcal{E}'$ alors $\{\} \notin \mathcal{E}''$ (et on peut construire un modèle de \mathcal{E}''),
- si $\{\} \in \mathcal{E}''$ alors $\{\} \notin \mathcal{E}'$ (et on peut construire un modèle de \mathcal{E}');



Propriétés de l'algorithme d'insatisfiabilité (complétude)

■ Remarque :

- si $\{\} \in \mathcal{E}'$ cela provient du fait que $\{p_{n+1}\} \in \mathcal{E}'$,
- si $\{\} \in \mathcal{E}''$ alors c'est que $\{\neg p_{n+1}\} \in \mathcal{E}''$,
- donc :
 - si $\{\} \in \mathcal{E}'$ et $\{\} \in \mathcal{E}''$ simultanément,
 - alors $\{\{p_{n+1}\}, \{\neg p_{n+1}\}\} \subseteq \mathcal{E}$, ce qui est impossible puisque \mathcal{E} saturé;

■ Conséquence :

- si $\{\} \in \mathcal{E}'$ alors $\{\} \notin \mathcal{E}''$ (et on peut construire un modèle de \mathcal{E}''),
- si $\{\} \in \mathcal{E}''$ alors $\{\} \notin \mathcal{E}'$ (et on peut construire un modèle de \mathcal{E}');
- donc il existe un modèle de \mathcal{E} .



Application : résolution et preuve de validité

- **But** : vérifier la validité d'une formule A ;
- **démarche** :
 - 1 convertir $\neg A$ en Forme Normale Conjonctive (FNC) :
 - 2 construire l'ensemble de clauses correspondant ;
 - 3 Appliquer l'algorithme de résolution :



Application : résolution et preuve de validité

- **But** : vérifier la validité d'une formule A ;
- **démarche** :
 - 1 convertir $\neg A$ en Forme Normale Conjonctive (FNC) :
 - 2 construire l'ensemble de clauses correspondant ;
 - 3 Appliquer l'algorithme de résolution :
 - si $\{\}$ est une résolvante, alors A est valide ;



Application : résolution et preuve de validité

- **But** : vérifier la validité d'une formule A ;
- **démarche** :
 - 1 convertir $\neg A$ en Forme Normale Conjonctive (FNC) :
 - 2 construire l'ensemble de clauses correspondant ;
 - 3 Appliquer l'algorithme de résolution :
 - si $\{\}$ est une résolvante, alors A est valide ;
 - si on ne peut pas obtenir $\{\}$, alors A n'est pas valide
➡ un contre-modèle peut être obtenu par extraction d'un modèle de l'ensemble de clauses saturé.



Application : résolution et preuve de conséquence

- **But** : vérifier si $\{B_1, \dots, B_n\} \vDash A$;
- **démarche** :
 - 1 convertir $(B_1 \wedge \dots \wedge B_n \wedge \neg A)$ en FNC, ce qui équivaut à :
 - convertir B_1 en FNC, ..., B_n en FNC;
 - convertir $\neg A$ en FNC;
 - prendre l'union des clauses;
 - 2 vérifier que cet ensemble est insatisfiable.



Insatisfiabilité par résolution (résumé)

- La **résolution** permet de vérifier qu'un ensemble de clauses est **insatisfiable** ;
- Deux applications :
 - vérifier qu'une formule **A est valide** :

$\models A$ ssi $\neg A$ est insatisfiable ;

- vérifier que $\{B_1, \dots, B_n\} \models A$:

$\{B_1, \dots, B_n\} \models A$ ssi $\{B_1, \dots, B_n, \neg A\}$ est insatisfiable.

où B_1, \dots, B_n doivent être mises sous **forme clausale**.



Un dernier mot sur les solveurs

- Un **solveur SAT** est un programme chargé de tester la satisfiabilité d'une formule ;
- la majorité des solveurs SAT sont des implémentations efficaces de l'**algorithme par résolution** vu ci-dessus ;
- le logiciel **TouIST** que nous allons utiliser se charge de l'interface d'édition, de la mise en FNC (format DIMACS), et de la communication avec le solveur (envoi de la FNC, réception/affichage des modèles s'ils existent).



Logique des propositions

Quantification sur ensembles finis



Exemple : *Tarski's world* restreint (objectif)

- Le but est de placer
 - des triangles et des rectangles
 - sur une grille $n \times n$;
- tout en respectant les contraintes suivantes :



Exemple : *Tarski's world* restreint (objectif)

- Le but est de placer
 - des triangles et des rectangles
 - sur une grille $n \times n$;
- tout en respectant les contraintes suivantes :
 - 1 toute colonne contient exactement un triangle;
 - 2 toute ligne contient exactement un rectangle;
 - 3 il n'y a pas d'objet (triangle ou rectangle) sur les diagonales.



Exemple : *Tarski's world* restreint (langage de modélisation)

- Les cases de la grilles sont numérotées par un couple (l, c) :
 - l représente le numéro de ligne ;
 - c représente le numéro de colonne ;
- Variables propositionnelles ($2n^2$ en tout) :

$$PROP = \bigcup_{\substack{l \in \{1..n\} \\ c \in \{1..n\}}} \{t_{l,c}\} \cup \bigcup_{\substack{l \in \{1..n\} \\ c \in \{1..n\}}} \{r_{l,c}\}$$



Exemple : *Tarski's world* restreint (une solution pour $n = 4$)

	▲	■	
▲			■
■			▲
	■	▲	

- $t_{1,2}$: « Il y a un triangle dans la case ligne 1 et colonne 2 » ;
 - $r_{4,2}$: « Il y a un rectangle dans la case ligne 4 et colonne 2 » ;
 - ...
- (➡ $2 \times 4^2 = 32$ variables propositionnelles différentes en tout)



Exemple : *Tarski's world* restreint (contrainte 1 pour $n = 4$)

- 1 « toute colonne contient exactement un triangle » signifie :



Exemple : *Tarski's world* restreint (contrainte 1 pour $n = 4$)

- 1 « toute colonne contient **exactement** un triangle » signifie :
 - chaque colonne contient **au moins** un triangle :



Exemple : *Tarski's world* restreint (contrainte 1 pour $n = 4$)

- 1 « toute colonne contient **exactement** un triangle » signifie :
 - chaque colonne contient **au moins** un triangle :
 - chaque colonne contient **au plus** un triangle :



Exemple : *Tarski's world* restreint (contrainte 1 pour $n = 4$)

- 1 « toute colonne contient **exactement** un triangle » signifie :
 - chaque colonne contient **au moins** un triangle :
$$(t_{1,1} \vee t_{2,1} \vee t_{3,1} \vee t_{4,1}) \wedge (t_{1,2} \vee t_{2,2} \vee t_{3,2} \vee t_{4,2}) \wedge (t_{1,3} \vee t_{2,3} \vee t_{3,3} \vee t_{4,3}) \wedge (t_{1,4} \vee t_{2,4} \vee t_{3,4} \vee t_{4,4})$$
 - chaque colonne contient **au plus** un triangle :



Exemple : *Tarski's world* restreint (contrainte 1 pour $n = 4$)

1 « toute colonne contient **exactement** un triangle » signifie :

■ chaque colonne contient **au moins** un triangle :

$$(t_{1,1} \vee t_{2,1} \vee t_{3,1} \vee t_{4,1}) \wedge (t_{1,2} \vee t_{2,2} \vee t_{3,2} \vee t_{4,2}) \wedge$$

$$(t_{1,3} \vee t_{2,3} \vee t_{3,3} \vee t_{4,3}) \wedge (t_{1,4} \vee t_{2,4} \vee t_{3,4} \vee t_{4,4})$$

■ chaque colonne contient **au plus** un triangle :

$$(t_{1,1} \rightarrow \neg(t_{2,1} \vee t_{3,1} \vee t_{4,1})) \wedge \cdots \wedge (t_{4,1} \rightarrow \neg(t_{1,1} \vee t_{2,1} \vee t_{3,1})) \wedge$$

...

$$(t_{1,4} \rightarrow \neg(t_{2,4} \vee t_{3,4} \vee t_{4,4})) \wedge \cdots \wedge (t_{4,4} \rightarrow \neg(t_{1,4} \vee t_{2,4} \vee t_{3,4}))$$



Exemple : *Tarski's world* restreint (contrainte 1 pour $n = 4$)

1 « toute colonne contient **exactement** un triangle » signifie :

■ chaque colonne contient **au moins** un triangle :

$$(t_{1,1} \vee t_{2,1} \vee t_{3,1} \vee t_{4,1}) \wedge (t_{1,2} \vee t_{2,2} \vee t_{3,2} \vee t_{4,2}) \wedge$$

$$(t_{1,3} \vee t_{2,3} \vee t_{3,3} \vee t_{4,3}) \wedge (t_{1,4} \vee t_{2,4} \vee t_{3,4} \vee t_{4,4})$$

■ chaque colonne contient **au plus** un triangle :

$$(t_{1,1} \rightarrow \neg(t_{2,1} \vee t_{3,1} \vee t_{4,1})) \wedge \cdots \wedge (t_{4,1} \rightarrow \neg(t_{1,1} \vee t_{2,1} \vee t_{3,1})) \wedge$$

...

$$(t_{1,4} \rightarrow \neg(t_{2,4} \vee t_{3,4} \vee t_{4,4})) \wedge \cdots \wedge (t_{4,4} \rightarrow \neg(t_{1,4} \vee t_{2,4} \vee t_{3,4}))$$

■ **nombre de littéraux** utilisés : $4^2 + 4^3 = 80$.



Exemple : *Tarski's world* restreint (constat)

- Pour la seule contrainte 1, il faut écrire deux formules contenant en tout $n^2 + n^3$ littéraux...
(➡ pour $n = 100$, plus de 10 millions de littéraux utilisés !)
- et il faut encore écrire les 2 autres contraintes !
- Conséquences :
 - lourd à écrire et à lire ;
 - difficile à étendre et à modifier ;
 - impraticable pour un n élevé.



Exemple : *Tarski's world* restreint (remarque)

- Le langage utilise des noms de variable constitués :
 - d'un symbole (*t* ou *r*) définissant le **type** de la variable ;
 - associé à deux indices (la ligne et la colonne de la case concernée, qui sont des entiers) ;
- ces variables sont reliées par des connecteurs identiques (soit \vee , soit \wedge).



Exemple : *Tarski's world* restreint (remarque)

- Le langage utilise des noms de variable constitués :
 - d'un symbole (*t* ou *r*) définissant le **type** de la variable ;
 - associé à deux indices (la ligne et la colonne de la case concernée, qui sont des entiers) ;
- ces variables sont reliées par des connecteurs identiques (soit \vee , soit \wedge).
 - ➡ en définitive, il y a des « motifs de formules » : disjonctions de variables propositionnelles de même type, conjonction de telles disjonctions.



Écriture compacte (idée de départ)

- $\sum_{i=1}^n x_i$ est l'écriture compacte de $x_1 + \dots + x_n$ où :
 - tous les opérateurs $+$ sont représentés par un unique connecteur généralisé Σ ;
 - toutes les variables x_1, \dots, x_n sont représentées par une seule variable x_i dont l'indice i varie de 1 à n ;
- cette écriture est d'autant plus compacte que n est grand;



Écriture compacte (idée de départ)

- $\sum_{i=1}^n x_i$ est l'écriture compacte de $x_1 + \dots + x_n$ où :
 - tous les opérateurs $+$ sont représentés par un unique connecteur généralisé Σ ;
 - toutes les variables x_1, \dots, x_n sont représentées par une seule variable x_i dont l'indice i varie de 1 à n ;
- cette écriture est d'autant plus compacte que n est grand ;
- **idée** : introduire des connecteurs généralisés \vee et \wedge jouant respectivement pour les connecteurs \vee et \wedge , le même rôle que Σ pour les opérateurs $+$.



Les connecteurs généralisés (définition)

- On définit 2 connecteurs généralisés ($A_i \in FORM$ pour tout $i \in \mathcal{I}$, un ensemble fini d'indices numériques ou non) :

$$\bigwedge_{i=1}^n A_i \equiv \bigwedge_{1 \leq i \leq n} A_i \equiv \bigwedge_{i \in \{1..n\}} A_i \equiv A_1 \wedge \dots \wedge A_n$$

$$\bigvee_{i=1}^n A_i \equiv \bigvee_{1 \leq i \leq n} A_i \equiv \bigvee_{i \in \{1..n\}} A_i \equiv A_1 \vee \dots \vee A_n$$



Les connecteurs généralisés (définition)

- On définit 2 connecteurs généralisés ($A_i \in FORM$ pour tout $i \in \mathcal{I}$, un ensemble fini d'indices numériques ou non) :

$$\bigwedge_{i=1}^n A_i \equiv \bigwedge_{1 \leq i \leq n} A_i \equiv \bigwedge_{i \in \{1..n\}} A_i \equiv A_1 \wedge \dots \wedge A_n$$

$$\bigvee_{i=1}^n A_i \equiv \bigvee_{1 \leq i \leq n} A_i \equiv \bigvee_{i \in \{1..n\}} A_i \equiv A_1 \vee \dots \vee A_n$$

- qui sont des **abréviations** de formules du langage de *LProp* et ne font donc **pas** partie de *LProp* ;



Les connecteurs généralisés (définition)

- On définit 2 connecteurs généralisés ($A_i \in FORM$ pour tout $i \in \mathcal{I}$, un ensemble fini d'indices numériques ou non) :

$$\bigwedge_{i=1}^n A_i \equiv \bigwedge_{1 \leq i \leq n} A_i \equiv \bigwedge_{i \in \{1..n\}} A_i \equiv A_1 \wedge \dots \wedge A_n$$

$$\bigvee_{i=1}^n A_i \equiv \bigvee_{1 \leq i \leq n} A_i \equiv \bigvee_{i \in \{1..n\}} A_i \equiv A_1 \vee \dots \vee A_n$$

- qui sont des **abréviations** de formules du langage de *LProp* et ne font donc **pas** partie de *LProp* ;
- mais qui peuvent être utilisés dans des formules de *LProp*, sachant qu'il est toujours possible de les substituer avec leur expression développée.



Les connecteurs généralisés (écriture compacte)

- Quand plusieurs connecteurs généralisés de **même type** (conjonctifs ou disjonctifs) se succèdent
- on peut les regrouper :



Les connecteurs généralisés (écriture compacte)

- Quand plusieurs connecteurs généralisés de **même type** (conjonctifs ou disjonctifs) se succèdent
- on peut les regrouper :
 - soient n **ensembles finis d'indices** $\mathcal{I}_1, \dots, \mathcal{I}_n$



Les connecteurs généralisés (écriture compacte)

- Quand plusieurs connecteurs généralisés de **même type** (conjonctifs ou disjonctifs) se succèdent
- on peut les regrouper :
 - soient n **ensembles finis d'indices** $\mathcal{I}_1, \dots, \mathcal{I}_n$
 - et n **variables d'indice** i_1, \dots, i_n ayant pour domaines respectifs $\mathcal{I}_1, \dots, \mathcal{I}_n$



Les connecteurs généralisés (écriture compacte)

- Quand plusieurs connecteurs généralisés de **même type** (conjonctifs ou disjonctifs) se succèdent
- on peut les regrouper :
 - soient n **ensembles finis d'indices** $\mathcal{I}_1, \dots, \mathcal{I}_n$
 - et n **variables d'indice** i_1, \dots, i_n ayant pour domaines respectifs $\mathcal{I}_1, \dots, \mathcal{I}_n$
 - et $A_{\{i_1, \dots, i_n\}}$ une formule dépendant d'un **sous-ensemble non vide** de variables d'indices, alors :



Les connecteurs généralisés (écriture compacte)

- Quand plusieurs connecteurs généralisés de **même type** (conjonctifs ou disjonctifs) se succèdent
- on peut les regrouper :
 - soient n **ensembles finis d'indices** $\mathcal{I}_1, \dots, \mathcal{I}_n$
 - et n **variables d'indice** i_1, \dots, i_n ayant pour domaines respectifs $\mathcal{I}_1, \dots, \mathcal{I}_n$
 - et $A_{\{i_1, \dots, i_n\}}$ une formule dépendant d'un **sous-ensemble non vide** de variables d'indices, alors :

$$\bigwedge_{i_1 \in \mathcal{I}_1} \cdots \bigwedge_{i_n \in \mathcal{I}_n} A_{\{i_1, \dots, i_n\}} \equiv \bigwedge_{\substack{i_1 \in \mathcal{I}_1 \\ \vdots \\ i_n \in \mathcal{I}_n}} A_{\{i_1, \dots, i_n\}} \quad (\text{idem pour } \vee)$$



Les connecteurs généralisés (nature des indices)

- Dans $\wedge_{i \in \mathcal{I}} A_i$ l'indice i est une méta-variable (elle ne fait pas partie du langage logique) ;



Les connecteurs généralisés (nature des indices)

- Dans $\wedge_{i \in \mathcal{I}} A_i$ l'indice i est une méta-variable (elle ne fait pas partie du langage logique) ;
- conséquence importante : dans l'abréviation suivante

$$\bigvee_{c \in \{2,4\}} t_{l,c} \equiv t_{l,2} \vee t_{l,4}$$

l'indice l n'apparaît dans aucun connecteur généralisé (on dit qu'il est non quantifié ou libre, cf. Diapo 252), alors :



Les connecteurs généralisés (nature des indices)

- Dans $\wedge_{i \in \mathcal{I}} A_i$ l'indice i est une **méta-variable** (elle ne fait **pas** partie du langage logique) ;
- **conséquence importante** : dans l'abréviation suivante

$$\bigvee_{c \in \{2,4\}} t_{l,c} \equiv t_{l,2} \vee t_{l,4}$$

l'indice l n'apparaît dans **aucun** connecteur généralisé (on dit qu'il est **non quantifié** ou **libre**, cf. Diapo 252), alors :

- la formule développée **n'est pas** une formule du langage car $t_{l,2}$ et $t_{l,4}$ **ne sont pas** des variables propositionnelles !



Les connecteurs généralisés (nature des indices)

- Dans $\wedge_{i \in \mathcal{I}} A_i$ l'indice i est une **méta-variable** (elle ne fait **pas** partie du langage logique) ;
- **conséquence importante** : dans l'abréviation suivante

$$\bigvee_{c \in \{2,4\}} t_{l,c} \equiv t_{l,2} \vee t_{l,4}$$

l'indice l n'apparaît dans **aucun** connecteur généralisé (on dit qu'il est **non quantifié** ou **libre**, cf. Diapo 252), alors :

- la formule développée **n'est pas** une formule du langage car $t_{l,2}$ et $t_{l,4}$ **ne sont pas** des variables propositionnelles !
- **tout indice apparaissant dans une formule utilisant des connecteurs généralisés doit être quantifié** par ces derniers.



Retour au *Tarski's world* (contrainte 1.1)

- Rappel de la contrainte :

$$\begin{aligned} & (t_{1,1} \vee t_{2,1} \vee t_{3,1} \vee t_{4,1}) \wedge (t_{1,2} \vee t_{2,2} \vee t_{3,2} \vee t_{4,2}) \wedge \\ & (t_{1,3} \vee t_{2,3} \vee t_{3,3} \vee t_{4,3}) \wedge (t_{1,4} \vee t_{2,4} \vee t_{3,4} \vee t_{4,4}) \end{aligned}$$



Retour au *Tarski's world* (contrainte 1.1)

- Rappel de la contrainte :

$$\begin{aligned} & (t_{1,1} \vee t_{2,1} \vee t_{3,1} \vee t_{4,1}) \wedge (t_{1,2} \vee t_{2,2} \vee t_{3,2} \vee t_{4,2}) \wedge \\ & (t_{1,3} \vee t_{2,3} \vee t_{3,3} \vee t_{4,3}) \wedge (t_{1,4} \vee t_{2,4} \vee t_{3,4} \vee t_{4,4}) \end{aligned}$$

- qui peut se réécrire ($\{1..4\}$, l'ensemble des entiers de 1 à 4) :

$$\left(\bigvee_{l \in \{1..4\}} t_{l,1} \right) \wedge \left(\bigvee_{l \in \{1..4\}} t_{l,2} \right) \wedge \left(\bigvee_{l \in \{1..4\}} t_{l,3} \right) \wedge \left(\bigvee_{l \in \{1..4\}} t_{l,4} \right)$$



Retour au *Tarski's world* (contrainte 1.1)

- Rappel de la contrainte :

$$\begin{aligned} & (t_{1,1} \vee t_{2,1} \vee t_{3,1} \vee t_{4,1}) \wedge (t_{1,2} \vee t_{2,2} \vee t_{3,2} \vee t_{4,2}) \wedge \\ & (t_{1,3} \vee t_{2,3} \vee t_{3,3} \vee t_{4,3}) \wedge (t_{1,4} \vee t_{2,4} \vee t_{3,4} \vee t_{4,4}) \end{aligned}$$

- qui peut se réécrire ($\{1..4\}$, l'ensemble des entiers de 1 à 4) :

$$\left(\bigvee_{l \in \{1..4\}} t_{l,1} \right) \wedge \left(\bigvee_{l \in \{1..4\}} t_{l,2} \right) \wedge \left(\bigvee_{l \in \{1..4\}} t_{l,3} \right) \wedge \left(\bigvee_{l \in \{1..4\}} t_{l,4} \right)$$

- qui peut se réécrire :

$$\bigwedge_{c \in \{1..4\}} \bigvee_{l \in \{1..4\}} t_{l,c}$$



Retour au *Tarski's world* (contrainte 1.2)

- Et la contrainte :

$$\begin{aligned} & \left(t_{1,1} \rightarrow \neg(t_{2,1} \vee t_{3,1} \vee t_{4,1}) \right) \wedge \cdots \wedge \left(t_{4,1} \rightarrow \neg(t_{1,1} \vee t_{2,1} \vee t_{3,1}) \right) \wedge \cdots \wedge \\ & \left(t_{1,4} \rightarrow \neg(t_{2,4} \vee t_{3,4} \vee t_{4,4}) \right) \wedge \cdots \wedge \left(t_{4,4} \rightarrow \neg(t_{1,4} \vee t_{2,4} \vee t_{3,4}) \right) \end{aligned}$$



Retour au *Tarski's world* (contrainte 1.2)

- Et la contrainte :

$$\begin{aligned} & \left(t_{1,1} \rightarrow \neg(t_{2,1} \vee t_{3,1} \vee t_{4,1}) \right) \wedge \cdots \wedge \left(t_{4,1} \rightarrow \neg(t_{1,1} \vee t_{2,1} \vee t_{3,1}) \right) \wedge \cdots \wedge \\ & \left(t_{1,4} \rightarrow \neg(t_{2,4} \vee t_{3,4} \vee t_{4,4}) \right) \wedge \cdots \wedge \left(t_{4,4} \rightarrow \neg(t_{1,4} \vee t_{2,4} \vee t_{3,4}) \right) \end{aligned}$$

- peut se réécrire :

$$\begin{aligned} & \left(t_{1,1} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{1\}} t_{l,1}\right) \right) \wedge \cdots \wedge \left(t_{4,1} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{4\}} t_{l,1}\right) \right) \wedge \cdots \wedge \\ & \left(t_{1,4} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{1\}} t_{l,4}\right) \right) \wedge \cdots \wedge \left(t_{4,4} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{4\}} t_{l,4}\right) \right) \end{aligned}$$



Retour au *Tarski's world* (contrainte 1.2)

- Et la contrainte :

$$\begin{aligned} & \left(t_{1,1} \rightarrow \neg(t_{2,1} \vee t_{3,1} \vee t_{4,1}) \right) \wedge \cdots \wedge \left(t_{4,1} \rightarrow \neg(t_{1,1} \vee t_{2,1} \vee t_{3,1}) \right) \wedge \cdots \wedge \\ & \left(t_{1,4} \rightarrow \neg(t_{2,4} \vee t_{3,4} \vee t_{4,4}) \right) \wedge \cdots \wedge \left(t_{4,4} \rightarrow \neg(t_{1,4} \vee t_{2,4} \vee t_{3,4}) \right) \end{aligned}$$

- peut se réécrire :

$$\begin{aligned} & \left(t_{1,1} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{1\}} t_{l,1}\right) \right) \wedge \cdots \wedge \left(t_{4,1} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{4\}} t_{l,1}\right) \right) \wedge \cdots \wedge \\ & \left(t_{1,4} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{1\}} t_{l,4}\right) \right) \wedge \cdots \wedge \left(t_{4,4} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{4\}} t_{l,4}\right) \right) \end{aligned}$$

- qui peut se réécrire :

$$\bigwedge_{l' \in \{1..4\}} \left(t_{l',1} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,1}\right) \right) \wedge \cdots \wedge \bigwedge_{l' \in \{1..4\}} \left(t_{l',4} \rightarrow \neg\left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,4}\right) \right)$$



Retour au *Tarski's world* (contrainte 1.2, suite et fin)

- Et la contrainte :

$$\bigwedge_{l' \in \{1..4\}} \left(t_{l',1} \rightarrow \neg \left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,1} \right) \right) \wedge \cdots \wedge \bigwedge_{l' \in \{1..4\}} \left(t_{l',4} \rightarrow \neg \left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,4} \right) \right)$$



Retour au *Tarski's world* (contrainte 1.2, suite et fin)

- Et la contrainte :

$$\bigwedge_{l' \in \{1..4\}} \left(t_{l',1} \rightarrow \neg \left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,1} \right) \right) \wedge \cdots \wedge \bigwedge_{l' \in \{1..4\}} \left(t_{l',4} \rightarrow \neg \left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,4} \right) \right)$$

- peut finalement se réécrire :

$$\bigwedge_{c \in \{1..4\}} \bigwedge_{l' \in \{1..4\}} \left(t_{l',c} \rightarrow \neg \left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,c} \right) \right)$$



Retour au *Tarski's world* (contrainte 1.2, suite et fin)

- Et la contrainte :

$$\bigwedge_{l' \in \{1..4\}} \left(t_{l',1} \rightarrow \neg \left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,1} \right) \right) \wedge \cdots \wedge \bigwedge_{l' \in \{1..4\}} \left(t_{l',4} \rightarrow \neg \left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,4} \right) \right)$$

- peut finalement se réécrire :

$$\bigwedge_{c \in \{1..4\}} \bigwedge_{l' \in \{1..4\}} \left(t_{l',c} \rightarrow \neg \left(\bigvee_{l \in \{1..4\} \setminus \{l'\}} t_{l,c} \right) \right)$$

- ou encore, plus généralement, pour $n \in \mathbb{N}$:

$$\bigwedge_{\substack{c \in \{1..n\} \\ l' \in \{1..n\}}} \left(t_{l',c} \rightarrow \neg \left(\bigvee_{l \in \{1..n\} \setminus \{l'\}} t_{l,c} \right) \right)$$



Retour au *Tarski's world* (conclusion)

- On a défini des **abréviations** qui ne sont rien d'autre que des **écritures compactes** de formules complexes à écrire :

$$\bigwedge_{c \in \{1..n\}} \bigvee_{l \in \{1..n\}} t_{l,c} \equiv (t_{1,1} \vee t_{2,1} \vee t_{3,1} \vee t_{4,1} \vee \dots) \wedge \dots$$

$$\bigwedge_{\substack{c \in \{1..n\} \\ l' \in \{1..n\}}} (t_{l',c} \rightarrow \neg(\bigvee_{l \in \{1..n\} \setminus \{l'\}} t_{l,c})) \equiv (t_{1,1} \rightarrow \neg(t_{2,1} \vee t_{3,1} \vee t_{4,1} \vee \dots)) \wedge \dots \wedge \dots$$



Retour au *Tarski's world* (conclusion)

- On a défini des **abréviations** qui ne sont rien d'autre que des **écritures compactes** de formules complexes à écrire :

$$\bigwedge_{c \in \{1..n\}} \bigvee_{l \in \{1..n\}} t_{l,c} \equiv (t_{1,1} \vee t_{2,1} \vee t_{3,1} \vee t_{4,1} \vee \dots) \wedge \dots$$

$$\bigwedge_{\substack{c \in \{1..n\} \\ l' \in \{1..n\}}} (t_{l',c} \rightarrow \neg(\bigvee_{l \in \{1..n\} \setminus \{l'\}} t_{l,c})) \equiv (t_{1,1} \rightarrow \neg(t_{2,1} \vee t_{3,1} \vee t_{4,1} \vee \dots)) \wedge \dots \wedge \dots$$

- ces abréviations n'augmentent pas l'expressivité de *LProp* ;



Retour au *Tarski's world* (conclusion)

- On a défini des **abréviations** qui ne sont rien d'autre que des **écritures compactes** de formules complexes à écrire :

$$\bigwedge_{c \in \{1..n\}} \bigvee_{l \in \{1..n\}} t_{l,c} \equiv (t_{1,1} \vee t_{2,1} \vee t_{3,1} \vee t_{4,1} \vee \dots) \wedge \dots$$

$$\bigwedge_{\substack{c \in \{1..n\} \\ l' \in \{1..n\}}} (t_{l',c} \rightarrow \neg(\bigvee_{l \in \{1..n\} \setminus \{l'\}} t_{l,c})) \equiv (t_{1,1} \rightarrow \neg(t_{2,1} \vee t_{3,1} \vee t_{4,1} \vee \dots)) \wedge \dots \wedge \dots$$

- ces abréviations n'augmentent pas l'expressivité de *LProp* ;
- mais permettent de passer de formules nécessitant $n^2 + n^3$ variables propositionnelles pour les écrire (voir Diapo. 236) à... 3 variables quel que soit n !



Retour au *Tarski's world* (exercice)

- Montrer que la contrainte 2 (voir Diapo. 232) peut s'écrire

$$\left(\bigwedge_{l \in \{1..n\}} \bigvee_{c \in \{1..n\}} r_{l,c} \right) \wedge \left(\bigwedge_{\substack{l \in \{1..n\} \\ c' \in \{1..n\}}} \left(r_{l,c'} \rightarrow \neg \left(\bigvee_{\substack{c \in \{1..n\} \setminus \{c'\}}} r_{l,c} \right) \right) \right)$$

- Montrer que la contrainte 3 peut s'écrire :

$$\left(\bigwedge_{i \in \{1..n\}} \neg t_{i,i} \right) \wedge \left(\bigwedge_{i \in \{1..n\}} \neg r_{n,n} \right) \wedge \left(\bigwedge_{i \in \{1..n\}} \neg t_{i,n+1-i} \right) \wedge \left(\bigwedge_{i \in \{1..n\}} \neg r_{i,n+1-i} \right)$$



Les connecteurs généralisés (propriétés remarquables)

- Conjonction ($A_i \in FORM$) :

$$\bigwedge_{i \in \{\}} A_i \equiv \top$$

$$\bigwedge_{i \in \{e, \dots\}} A_i \equiv A_e \wedge \bigwedge_{i \in \{\dots\}} A_i$$



Les connecteurs généralisés (propriétés remarquables)

- Conjonction ($A_i \in FORM$) :

$$\bigwedge_{i \in \{\}} A_i \equiv \top$$

$$\bigwedge_{i \in \{\textcolor{red}{e}, \dots\}} A_i \equiv A_{\textcolor{red}{e}} \wedge \bigwedge_{i \in \{\dots\}} A_i$$

- Disjonction ($A_i \in FORM$) :

$$\bigvee_{i \in \{\}} A_i \equiv \perp$$

$$\bigvee_{i \in \{\textcolor{red}{e}, \dots\}} A_i \equiv A_{\textcolor{red}{e}} \vee \bigvee_{i \in \{\dots\}} A_i$$



Les connecteurs généralisés (propriétés remarquables)

■ Conjonction ($A_i \in FORM$) :

$$\bigwedge_{i \in \{\}} A_i \equiv \top$$

$$\bigwedge_{i \in \{e, \dots\}} A_i \equiv A_e \wedge \bigwedge_{i \in \{\dots\}} A_i$$

■ Disjonction ($A_i \in FORM$) :

$$\bigvee_{i \in \{\}} A_i \equiv \perp$$

$$\bigvee_{i \in \{e, \dots\}} A_i \equiv A_e \vee \bigvee_{i \in \{\dots\}} A_i$$

■ Exemple :

$$\begin{aligned} \bigvee_{l \in \{1,2,3\}} t_{1,l} &\equiv t_{1,1} \vee \bigvee_{l \in \{2,3\}} t_{1,l} \\ &\equiv t_{1,1} \vee t_{1,2} \vee \bigvee_{l \in \{3\}} t_{1,l} \\ &\equiv t_{1,1} \vee t_{1,2} \vee t_{1,3} \vee \bigvee_{l \in \{\}} t_{1,l} \\ &\equiv t_{1,1} \vee t_{1,2} \vee t_{1,3} \vee \perp \end{aligned}$$

$$\equiv t_{1,1} \vee t_{1,2} \vee t_{1,3}$$



Les connecteurs généralisés avec contraintes (exemple)

- Comment exprimer : « Toute case hors de la diagonale contient un triangle » ?



Les connecteurs généralisés avec contraintes (exemple)

- Comment exprimer : « Toute case hors de la diagonale contient un triangle » ?
- Solution :

$$\bigwedge_{l \in \{1..4\}} \bigwedge_{c \in \{1..4\} | l \neq c} t_{l,c} \equiv \bigwedge_{\substack{l \in \{1..4\} \\ c \in \{1..4\} \\ l \neq c}} t_{l,c}$$



Les connecteurs généralisés avec contraintes (exercice)

- Exprimez (considérant une grille de dimension n) :
 - « toute case au-dessus de la diagonale $(1,1) \rightarrow (n,n)$ contient un rectangle »;
 - « les cases dont la somme des indices est paire contiennent un triangle ».



Les connecteurs généralisés avec contraintes (exercice)

■ Exprimez (considérant une grille de dimension n) :

- « toute case au-dessus de la diagonale $(1,1) \rightarrow (n,n)$ contient un rectangle »;
- « les cases dont la somme des indices est paire contiennent un triangle ».

$$\bigwedge_{\substack{l \in \{1..n\} \\ c \in \{1..n\} \\ l < c}} r_{l,c}$$



Les connecteurs généralisés avec contraintes (exercice)

■ Exprimez (considérant une grille de dimension n) :

- « toute case au-dessus de la diagonale $(1,1) \rightarrow (n,n)$ contient un rectangle »;

$$\bigwedge_{\substack{l \in \{1..n\} \\ c \in \{1..n\} \\ l < c}} r_{l,c}$$

- « les cases dont la somme des indices est paire contiennent un triangle ».

$$\bigwedge_{\substack{l \in \{1..n\} \\ c \in \{1..n\} \\ (l+c) \bmod 2 = 0}} t_{l,c}$$



Les connecteurs généralisés avec contraintes (généralisation)

- Si $\odot \in \{\wedge, \vee\}$ un(e méta-variable de) connecteur généralisé :

$$\bigodot_{i_1 \in E_1} \cdots \bigodot_{i_n \in E_n : C} A_{\{i_1, \dots, i_n\}}$$

est une imbrication de connecteurs généralisés où :

- $n \in \mathbb{N}$;
- E_1, \dots, E_n sont des ensembles finis;
- C est une contrainte Booléenne utilisant des éléments de $\{i_1, \dots, i_n\}$.
- dans ce cas, seuls les $A_{\{i_1, \dots, i_n\}}$ vérifiant la contrainte C appartiennent à la formule développée correspondante.



Les connecteurs généralisés avec contraintes (exemple)

- Pour toute ligne l il y a au moins une colonne c paire t.q. $t_{l,c}$ vrai :

$$\bigwedge_{l \in \{1..4\}} \bigvee_{c \in \{1..4\} : c \bmod 2 = 0} t_{l,c} \equiv \\ (t_{1,2} \vee t_{1,4}) \wedge (t_{2,2} \vee t_{2,4}) \wedge (t_{3,2} \vee t_{3,4}) \wedge (t_{4,2} \vee t_{4,4})$$



Variables libres, variables liées (définition)

- Chaque occurrence d'un quantificateur **lie** une variable ;
- toute variable qui n'est pas liée est **libre** ;



Variables libres, variables liées (définition)

- Chaque occurrence d'un quantificateur **lie** une variable ;
- toute variable qui n'est pas liée est **libre** ;
- exemple (d'un échiquier de 64 cases et 32 pièces) :
 - $v_{i,j}$: la case (i,j) est vide (pour $i,j \in \{1..8\}$) ;
 - $p_{k,i,j}$: la pièce $k \in \{1..32\}$ est sur la case (i,j) ;



Variables libres, variables liées (définition)

- Chaque occurrence d'un quantificateur **lie** une variable ;
- toute variable qui n'est pas liée est **libre** ;
- exemple (d'un échiquier de 64 cases et 32 pièces) :
 - $v_{i,j}$: la case (i,j) est vide (pour $i,j \in \{1..8\}$) ;
 - $p_{k,i,j}$: la pièce $k \in \{1..32\}$ est sur la case (i,j) ;
 - « une case vide ne contient aucune pièce » est représenté par :

$$v_{i,j} \rightarrow \bigwedge_{k \in \{1..32\}} \neg p_{k,i,j}$$

contient une variable liée (***k***) et deux variables libres (***i*** et ***j***) ;



Variables libres, variables liées (renommage)

- On peut **renommer** des variables liées sans modifier l'interprétation d'une formule ;
 - exemple :

$$v_{i,j} \rightarrow \bigwedge_{k \in \{1..32\}} \neg p_{k,i,j} \equiv v_{i,j} \rightarrow \bigwedge_{n \in \{1..32\}} \neg p_{n,i,j}$$



Variables libres, variables liées (renommage)

- On peut **renommer** des variables liées sans modifier l'interprétation d'une formule ;
 - exemple :

$$v_{i,j} \rightarrow \bigwedge_{k \in \{1..32\}} \neg p_{k,i,j} \equiv v_{i,j} \rightarrow \bigwedge_{n \in \{1..32\}} \neg p_{n,i,j}$$

- mais des précautions s'imposent :
 - renommer *k* en *i*

$$v_{i,j} \rightarrow \bigwedge_{k \in \{1..32\}} \neg p_{k,i,j} \equiv v_{i,j} \rightarrow \bigwedge_{i \in \{1..32\}} \neg p_{i,i,j}$$

fait passer la variable *i* précédemment **libre** au statut **lié** (et modifie le sens de la formule !).



Variables libres, variables liées (complication)

- La notion de variable liée / libre est relative à une sous-formule ;
- exemple :
 - dans

$$\bigwedge_{i \in \{1..8\}} \bigwedge_{j \in \{1..8\}} v_{i,j} \rightarrow \bigwedge_{k \in \{1..32\}} \neg p_{k,i,j}$$

toutes les variables sont liées,



Variables libres, variables liées (complication)

- La notion de variable liée / libre est relative à une sous-formule ;
- exemple :
 - dans

$$\bigwedge_{i \in \{1..8\}} \bigwedge_{j \in \{1..8\}} v_{i,j} \rightarrow \bigwedge_{k \in \{1..32\}} \neg p_{k,i,j}$$

toutes les variables sont liées, alors que dans la sous-formule

$$\bigwedge_{k \in \{1..32\}} \neg p_{k,i,j}$$

les variables *i* et *j* sont libres.



Variables libres, variables liées (simultanément)

- Dans :

$$v_{i,j} \rightarrow \bigwedge_{i \in \{1..8\}} \bigwedge_{j \in \{1..8\}} \bigwedge_{k \in \{1..32\}} \neg p_{k,i,j}$$

- i et j sont libres dans l'antécédent de l'implication ;
- et i et j sont liés dans le conséquent de l'implication ;



Variables libres, variables liées (simultanément)

- Dans :

$$v_{i,j} \rightarrow \bigwedge_{i \in \{1..8\}} \bigwedge_{j \in \{1..8\}} \bigwedge_{k \in \{1..32\}} \neg p_{k,i,j}$$

- *i* et *j* sont libres dans l'antécédent de l'implication ;
- et *i* et *j* sont liés dans le conséquent de l'implication ;
- il est préférable :
 - de renommer les variables liées ayant des occurrences libres ;
 - d'utiliser de nouveaux noms de variable lors du renommage.

$$v_{i,j} \rightarrow \bigwedge_{i' \in \{1..8\}} \bigwedge_{j' \in \{1..8\}} \bigwedge_{k \in \{1..32\}} \neg p_{k,i',j'}$$



Logique des propositions

*Toulouse Integrated
Satisfiability Tool
(TouIST)*



TouIST : qu'est-ce que c'est ?

- Outil développé à l'**IRIT** dans le département d'Intelligence Artificielle (voir www.irit.fr/TouIST/);



ToulIST : qu'est-ce que c'est ?

- Outil développé à l'IRIT dans le département d'Intelligence Artificielle (voir www.irit.fr/ToulIST/) ;
- permettant d'encoder tout problème formulé en logique propositionnelle, avec des variantes telles que :
 - la logique QBF permettant de quantifier sur les variables propositionnelles ;



ToulIST : qu'est-ce que c'est ?

- Outil développé à l'IRIT dans le département d'Intelligence Artificielle (voir www.irit.fr/ToulIST/) ;
- permettant d'encoder tout problème formulé en logique propositionnelle, avec des variantes telles que :
 - la logique QBF permettant de quantifier sur les variables propositionnelles ;
 - ou les logiques intégrant des contraintes numériques, entières ou flottantes ;



TouIST : qu'est-ce que c'est ?

- Outil développé à l'IRIT dans le département d'Intelligence Artificielle (voir www.irit.fr/TouIST/) ;
- permettant d'encoder tout problème formulé en logique propositionnelle, avec des variantes telles que :
 - la logique QBF permettant de quantifier sur les variables propositionnelles ;
 - ou les logiques intégrant des contraintes numériques, entières ou flottantes ;
- et qui utilise un langage puissant intégrant des méta-variables et des connecteurs généralisés permettant une quantification sur des ensembles finis.



- On décrit un problème avec un ensemble de formules ;
- puis on exécute le solveur :



Principe

- On décrit un problème avec un ensemble de formules ;
- puis on exécute le solveur :
 - si aucun modèle n'existe alors *TouIST* affiche « *Unsat* » (i.e., « La formule est insatisfiable ») ;



Principe

- On décrit un problème avec un ensemble de formules ;
- puis on exécute le solveur :
 - si aucun modèle n'existe alors TouIST affiche « *Unsat* » (i.e., « La formule est insatisfiable ») ;
 - sinon il affiche tous les **modèles** trouvés (mais pas les contre-modèles).



Installation

- Pour installer **TouIST**, il suffit...



Installation

- Pour installer TouIST, il suffit...
 - d'ouvrir son navigateur préféré...
 - et d'aller sur <https://touist.irit.fr/> !



Cadre de saisie des formules

The screenshot shows the TouIST web interface. At the top, there is a navigation bar with 'TouIST' dropdown, 'File' dropdown, 'SAT' dropdown, and 'Solve' button. To the right are 'Register' and 'Login' buttons. Below the navigation bar is a large input area where logical formulas can be entered. A red oval highlights the first two lines of input:

```
1 p
2 p => q
```

On the right side of the input area, there is a vertical tab bar with four tabs: 'Logical' (which is selected), 'CNF', 'DIMACS', and 'LaTeX'. Under the 'Logical' tab, the formulas p and $p \Rightarrow q$ are displayed in separate input fields.



Cadre d'affichage des formules

The screenshot shows the TouIST interface with the following elements:

- Top navigation bar: ToulIST, File, SAT, Solve, Register, Login.
- Input area: Shows two logical formulas:
 - Line 1: p
 - Line 2: $p \Rightarrow q$
- Output area: A large red box highlights the output section, which contains:
 - Logical tab (selected)
 - CNF tab
 - DIMACS tab
 - LaTeX tab

The output lines correspond to the input formulas: p and $p \Rightarrow q$.



Cadre d'affichage des FNC

TouIST ▾ File ▾ SAT ▾ Solve Register Login

1 p
2 p => q

Logical CNF DIMACS LaTeX

{ p }
{ q , -p }

>



Accès aux boutons de raccourcis

The screenshot shows the TouIST web interface. At the top, there is a navigation bar with 'TouIST' dropdown, 'File' dropdown, 'SAT' dropdown, 'Solve' button, 'Register' button, and 'Login' button. Below the navigation bar, the main area displays a logical proof:

```
1 p
2 p => q
```

To the left of the proof, there is a sidebar with four tabs: 'Logical' (which is selected), 'CNF', 'DIMACS', and 'LaTeX'. On the far left of the sidebar, there is a small button with a right-pointing arrow and a red callout bubble containing the text 'clic!' pointing towards it.



Menu des boutons de raccourcis

The screenshot shows the TouIST interface with a red box highlighting the 'Shortcuts Buttons Menu' on the left. The menu items are:

- > Connectors
- > Big ops
- > Sets
- > Other
- > Config

In the center, there is a SAT solver input area with the following content:

```
1 p
2 p => q
```

On the right, there are tabs for Logical, CNF, DIMACS, and LaTeX. The CNF tab is selected. Below it, two sets of logical expressions are listed:

- { p }
- { q , -p }



Raccourcis de connecteurs

TouIST ▾ File ▾ SAT ▾ Solve Register Login

Connectors

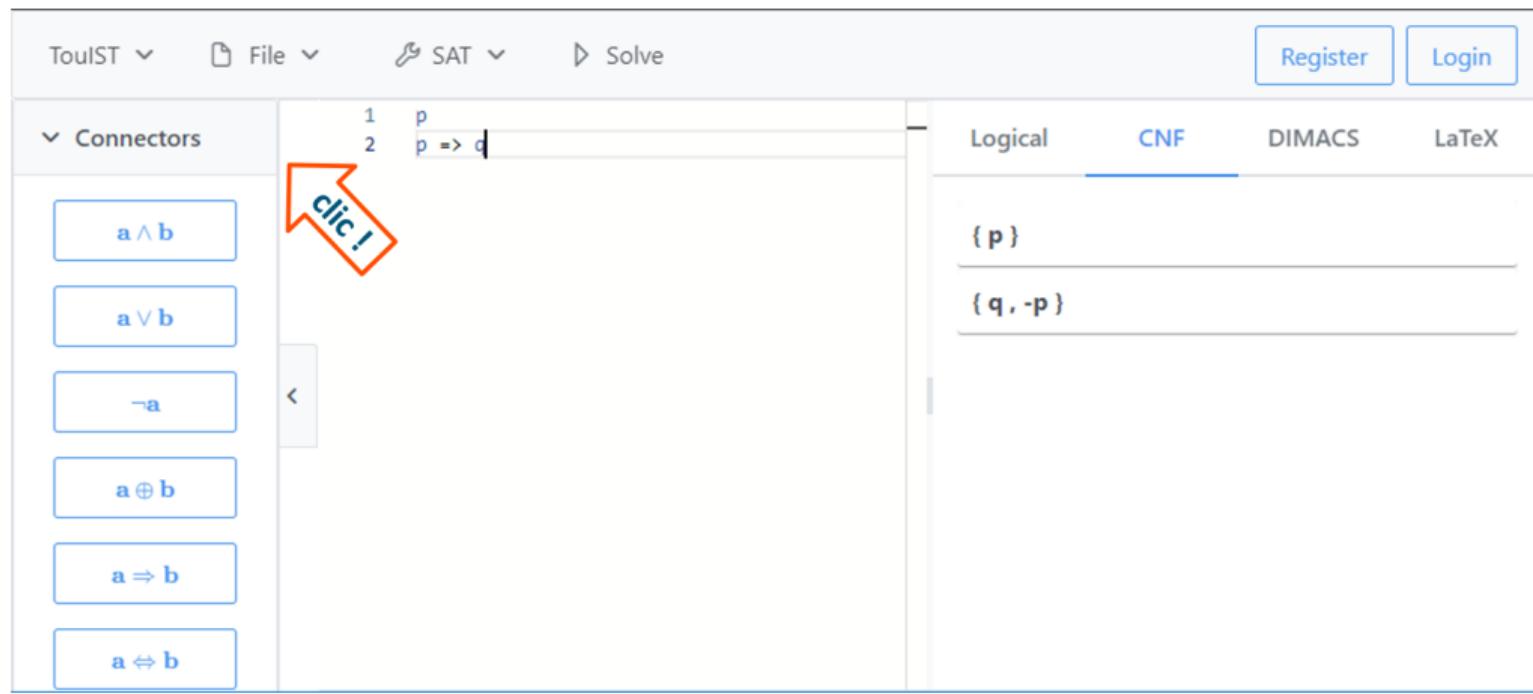
1 p
2 $p \Rightarrow q$

clic!

$a \wedge b$
 $a \vee b$
 $\neg a$
 $a \oplus b$
 $a \Rightarrow b$
 $a \Leftrightarrow b$

Logical CNF DIMACS LaTeX

{ p }
{ q , $\neg p$ }





Et autres raccourcis

TouIST ▾ File ▾ SAT ▾ Solve Register Login

> Connectors

1 p
2 p => q

◀ clic !

◀ clic !

◀

↳ Logical CNF DIMACS LaTeX

{ p }

{ q , -p }

↳

Big ops

$\wedge_{i \in a}$

$\vee_{i \in a}$

Sets

atmost(.,.)

atleast(.,.)



Choix du solveur

The screenshot shows the TouIST web application interface. On the left is a sidebar with links: Connectors, Big ops, Sets, Other, and Config. A red arrow labeled "dic!" points to the "SAT" dropdown menu, which is open and displays options: SAT, QF_IDL, QF_RDL, QF_LIA, and QF_LRA. The main area has tabs for Logical, CNF, DIMACS, and LaTeX, with "Logical" selected. Below the tabs, there are input fields for a logical expression: p and $p \Rightarrow q$.

TouIST ▾ File ▾ SAT ▾ Solve Register Login

Connectors
Big ops
Sets
Other
Config

SAT

- SAT
- QF_IDL
- QF_RDL
- QF_LIA
- QF_LRA

Logical CNF DIMACS LaTeX

p

$p \Rightarrow q$



Lancement du solveur choisi

TouIST ▾ File ▾ SAT ▾ Solve Register Login

1 p
2 $p \Rightarrow q$

clic!

Logical CNF DIMACS LaTeX

p

$p \Rightarrow q$

The screenshot shows the TouIST web interface. At the top, there's a navigation bar with 'TouIST', 'File', 'SAT', 'Solve' (with a dropdown arrow), 'Register', and 'Login'. Below the navigation is a text input area containing two lines of logic: '1 p' and '2 $p \Rightarrow q$ '. A large orange arrow points from the text area towards the 'Solve' button. To the right of the input area is a sidebar with tabs: 'Logical' (which is selected and underlined in blue), 'CNF', 'DIMACS', and 'LaTeX'. Under each tab, there are two text fields: one for 'p' and one for ' $p \Rightarrow q$ '. On the far left, there's a vertical scroll bar and a small icon with a right-pointing arrow.



Fenêtre d'affichage des modèles

TouIST

File

SAT

Solve

Register

Login

 True (1) False (0) Other

Regex

Export

Return to code

Model 1



Variables propositionnelles
apparaissant dans la formule



Une valuation de ces variables
(modèle de la formule initiale)



Modèle suivant/précédent

TouIST ▾

File ▾

SAT ▾

Solve

Register

Login

 True (1) False (0) Other

Regex

Export

Return to code

Model 1

p	1
q	1

Navigation entre les différents modèles trouvés

<< < 1 > >>



Retour à la fenêtre d'édition

TouIST ▾ File ▾ SAT ▾ Solve Register Login

True (1) False (0) Other Regex

Export Return to code

Model 1

p 1

q 1

1

« < > »

A screenshot of the TouIST interface showing a model. The model contains two variables: p and q, both assigned the value 1. A red arrow points to the "Return to code" button, which is highlighted with a red border. The word "clic!" is written in blue on the arrow. The interface includes a navigation bar at the top with links for "TouIST", "File", "SAT", "Solve", "Register", and "Login". Below the navigation bar are checkboxes for "True (1)", "False (0)", and "Other", and a "Regex" input field. On the right side, there are "Export" and "Return to code" buttons. The main area is titled "Model 1" and shows the variable assignments p=1 and q=1. At the bottom, there are navigation arrows for "«", "<", "1", ">", and "»".



Syntaxe des connecteurs logiques

■ Les connecteurs de base :

Logique propositionnelle	Langage TouIST
$\neg p$	not p
$p \wedge q$	p and q
$p \vee q$	p or q
$p \oplus q$	p xor q
$p \rightarrow q$	p => q
$p \leftrightarrow q$	p <=> q

■ Exemple : $(p \vee q) \wedge r$ s'écrit : « (p or q) and r ».



Exemple de la maladie (énoncé)

- On suppose que l'on a l'ensemble H d'hypothèses suivantes :
 - 1 Si le patient a la rougeole, il a de la fièvre ;
 - 2 Si le patient a une hépatite, mais pas la rougeole, il a le teint jaune ;
 - 3 Si le patient a de la fièvre ou le teint jaune, il a une hépatite ou la rougeole ;
 - 4 Le patient n'a pas le teint jaune mais il a de la fièvre ;
 - 5 donc il a la rougeole.



Exemple de la maladie (modélisation)

- H est l'ensemble contenant les formules suivantes :
 - 1 $\text{rougeole} \rightarrow \text{fievre}$;
 - 2 $\text{hepatite} \wedge \neg\text{rougeole} \rightarrow \text{teintJaune}$;
 - 3 $\text{fievre} \vee \text{teintJaune} \rightarrow \text{hepatite} \vee \text{rougeole}$;
 - 4 $\neg\text{teintJaune} \wedge \text{fievre}$;
- la conclusion C à vérifier est : rougeole ;



Exemple de la maladie (modélisation)

- H est l'ensemble contenant les formules suivantes :
 - 1 $\text{rougeole} \rightarrow \text{fievre}$;
 - 2 $\text{hepatite} \wedge \neg\text{rougeole} \rightarrow \text{teintJaune}$;
 - 3 $\text{fievre} \vee \text{teintJaune} \rightarrow \text{hepatite} \vee \text{rougeole}$;
 - 4 $\neg\text{teintJaune} \wedge \text{fievre}$;
- la conclusion C à vérifier est : rougeole ;
- il faut donc démontrer que $H \models C$, i.e. que $H \cup \{\neg C\}$ est insatisfiable.



Exemple de la maladie (programme TouIST)

- Code à écrire dans le **cadre de saisie des formules** :

```
;; ensemble H des hypotheses
rougeole => fievre
hepatite and not rougeole => teintJaune
fievre or teintJaune => hepatite or rougeole
not teintJaune and fievre
```

```
;; negation de la conclusion C
not rougeole
```



Exemple de la maladie (saisie dans ToulIST)

ToulIST ▾ File ▾ SAT ▾ Solve Register Login

```
1  ;; ensemble H des hypothèses
2  rougeole => fièvre
3  hépatite et non rougeole => teintJaune
4  fièvre ou teintJaune => hépatite ou rougeole
5  non teintJaune et fièvre
6
7  ;; négation de la conclusion C
8  non rougeole
```

>

Logical	CNF	DIMACS	LaTeX
$\text{rougeole} \Rightarrow \text{fièvre}$			
	$\text{hépatite} \wedge \neg \text{rougeole} \Rightarrow \text{teintJaune}$		
		$\text{fièvre} \vee \text{teintJaune} \Rightarrow \text{hépatite} \vee \text{rougeole}$	
			$\neg \text{teintJaune} \wedge \text{fièvre}$
			$\neg \text{rougeole}$



Exemple de la maladie (résultats)

TouIST ▾ File ▾ SAT ▾ Solve

```
1 ;; ensemble H des hypothèses
2 rougeole => fièvre
3 hépatite et non rougeole => teintJaune
4 fièvre ou teintJaune => hépatite ou rougeole
5 non teintJaune et fièvre
6
7 ;; négation de la conclusion C
8 non rougeole
```

i Unsat

Logical CNF DIMACS LaTeX

Réponse de TouIST (« insatisfiable »)

$\text{rougeole} \Rightarrow \text{fièvre}$

$\text{hépatite} \wedge \neg \text{rougeole} \Rightarrow \text{teintJaune}$

$\text{fièvre} \vee \text{teintJaune} \Rightarrow \text{hépatite} \vee \text{rougeole}$

$\neg \text{teintJaune} \wedge \text{fièvre}$

$\neg \text{rougeole}$



Les méta-variables

- On peut définir des méta-variables **de valeurs** (pas de formules !) ;



Les méta-variables

- On peut définir des méta-variables **de valeurs** (pas de formules !) ;
- syntaxiquement, elles sont composées du symbole **\$** suivi d'un **identificateur** commençant par une lettre ;



Les méta-variables

- On peut définir des méta-variables **de valeurs** (pas de formules !) ;
- syntaxiquement, elles sont composées du symbole **\$** suivi d'un **identificateur** commençant par une lettre ;
- Exemple :
 - $N = 3$ s'écrit $\$N = 3$



Les méta-variables

- On peut définir des méta-variables **de valeurs** (pas de formules !) ;
- syntaxiquement, elles sont composées du symbole **\$** suivi d'un **identificateur** commençant par une lettre ;
- Exemple :
 - $N = 3$ s'écrit $\$N = 3$
 - $VAL = \{1, 2, 3\}$ s'écrit $\$VAL = [1, 2, 3]$



Les méta-variables

- On peut définir des méta-variables **de valeurs** (pas de formules !) ;
- syntaxiquement, elles sont composées du symbole **\$** suivi d'un **identificateur** commençant par une lettre ;
- Exemple :
 - $N = 3$ s'écrit $\$N = 3$
 - $VAL = \{1, 2, 3\}$ s'écrit $\$VAL = [1, 2, 3]$
 - ou encore : $\$VAL = [1 .. \$N]$



Les connecteurs généralisés

- TouIST accepte de manière native les connecteurs généralisés dans son langage :

Logique propositionnelle

Pour un ensemble fini E ,

$$\bigwedge_{x \in E} p_x$$

Langage TouIST

```
bigand $x in $E:  
  p($x)  
end
```



Les connecteurs généralisés

- TouIST accepte de manière native les connecteurs généralisés dans son langage :

Logique propositionnelle

Pour un ensemble fini E ,

$$\bigwedge_{x \in E} p_x$$

$$\bigvee_{x \in E} p_x$$

Langage TouIST

```
bigand $x in $E:  
  p($x)  
end
```

```
bigor $x in $E:  
  p($x)  
end
```



Les connecteurs généralisés (exemple)

- Tous les pays (parmi : Portugal, Espagne, France, Italie) ont au moins une couleur (parmi : jaune, rouge, vert ou bleu) ;



Les connecteurs généralisés (exemple)

- Tous les pays (parmi : Portugal, Espagne, France, Italie) ont au moins une couleur (parmi : jaune, rouge, vert ou bleu) ;
- soit $colorie_{x,y}$ un ensemble de variables propositionnelles « le pays x est colorié avec la couleur y »



Les connecteurs généralisés (exemple)

- Tous les pays (parmi : Portugal, Espagne, France, Italie) ont au moins une couleur (parmi : jaune, rouge, vert ou bleu) ;
- soit $colorie_{x,y}$ un ensemble de variables propositionnelles « le pays x est colorié avec la couleur y »
- si on suppose les ensembles $COULEURS$ et $PAYS$ définis, alors la propriété ci-dessus se représente comme suit :

$$\bigwedge_{x \in PAYS} \bigvee_{y \in COULEURS} colorie_{x,y}$$



Les connecteurs généralisés (programme TouIST)

```
$PAYS = [ Portugal, Espagne, France, Italie ]  
$COULEUR = [ jaune, rouge, vert, bleu ]  
bigand $x in $PAYS:  
    bigor $y in $COULEUR:  
        colorie($x,$y)  
    end  
end
```



Les connecteurs généralisés (résultats TouIST)

TouIST ▾ File ▾ SAT ▾ Solve

True (1) False (0) Other Regex

Export

Export all models

Model 1

colorie(Espagne,bleu) 0

colorie(Espagne,jaune) 0

colorie(Espagne,rouge) 0

colorie(Espagne,vert) 1



Les connecteurs généralisés (remarque 1)

- Les résultats précédents représentent 1 valuation (mais il y en a beaucoup d'autres) ;
- comme il y a une disjonction sur les couleurs pour un même pays, on constate dans la valuation précédente que **pour un pays donné**, il y a **au moins une** variable lui attribuant une couleur qui est vraie ;
- en l'absence d'autre contrainte :
 - plusieurs couleurs peuvent être attribuées à un même pays ;
 - une même couleur peut être attribuée à plusieurs pays (voire, à tous).



Les connecteurs généralisés (remarque 2)

- Le programme précédent génère toutes les variables $colorie_{x,y}$ pour $(x,y) \in PAYS \times COULEURS$ (produit cartésien) ;
- chaque valuation attribue une valeur de vérité à **toutes** les variables propositionnelles ;
- il n'y a pas d'**hypothèse du monde clos**, où tout ce qui ne serait pas vrai serait faux ! (si la valeur de vérité n'est pas spécifiée, c'est qu'elle peut être vraie ou fausse indifféremment, **pas** qu'elle est fausse).



Les connecteurs généralisés avec contrainte (exemple)

- On cherche à caractériser si des entiers sont pairs ou non :

$$\left(\bigwedge_{\substack{i \in \{1..99\} \\ i \bmod 2 = 0}} \text{estPair}_i \right) \wedge \left(\bigwedge_{\substack{i \in \{1..99\} \\ i \bmod 2 \neq 0}} \neg \text{estPair}_i \right)$$



Les connecteurs généralisés avec contrainte (exemple)

- On cherche à caractériser si des entiers sont pairs ou non :

$$\left(\bigwedge_{\substack{i \in \{1..99\} \\ i \bmod 2 = 0}} \text{estPair}_i \right) \wedge \left(\bigwedge_{\substack{i \in \{1..99\} \\ i \bmod 2 \neq 0}} \neg \text{estPair}_i \right)$$

```
(bigand $i in [1..99] when $i mod 2 == 0 :  
    estPair($i)  
end)  
and  
(bigand $i in [1..99] when $i mod 2 != 0 :  
    not estPair($i)  
end)
```



Les connecteurs généralisés avec contrainte (résultat)

TouIST ▾ File ▾ SAT ▾ Solve Register Login

True (1) False (0) Other Regex Export Return to code

Model 1

estPair(71)	0
estPair(72)	1
estPair(73)	0
estPair(74)	1
estPair(75)	0



Les métavariables « tableau »

- On cherche :
 - à définir des variables propositionnelles $paysEtendu(x)$ signifiant : « le pays x est étendu »
 - un pays est « étendu » ssi sa superficie est $> 400000 \text{ km}^2$;
- \$PAYS = [Portugal, Espagne, France, Italie]



Les métavariables « tableau »

- On cherche :
 - à définir des variables propositionnelles $paysEtendu(x)$ signifiant : « le pays x est étendu »
 - un pays est « étendu » ssi sa superficie est $> 400000 \text{ km}^2$;

$\$PAYS = [\text{Portugal}, \text{Espagne}, \text{France}, \text{Italie}]$

$\$Superficie(\text{Portugal}) = 92042$

$\$Superficie(\text{Espagne}) = 511015$

$\$Superficie(\text{France}) = 551695$

$\$Superficie(\text{Italie}) = 301234$



Les métavariables « tableau »

- On cherche :
 - à définir des variables propositionnelles $paysEtendu(x)$ signifiant : « le pays x est étendu »
 - un pays est « étendu » ssi sa superficie est $> 400000 \text{ km}^2$;

$\$PAYS = [\text{Portugal}, \text{Espagne}, \text{France}, \text{Italie}]$

$\$Superficie(\text{Portugal}) = 92042$

$\$Superficie(\text{Espagne}) = 511015$

$\$Superficie(\text{France}) = 551695$

$\$Superficie(\text{Italie}) = 301234$

bigand \$x in \$PAYS when \$Superficie(\$x) > 400000:
 paysEtendu(\$x)

end



Structure interne de TouIST

- TouIST est en fait composé de deux couches distinctes :
 - l'**interface graphique** qui permet de saisir l'ensemble des formules du modèle en cours d'analyse ainsi que le solveur à utiliser ;
 - la **couche de calcul** (qui transforme et traduit en langage DIMACS l'ensemble des formules saisies dans la fenêtre graphique, appelle le bon solveur, et récupère éventuellement les modèles).
- La couche de calcul utilise donc la couche graphique avant d'appeler un solveur, et lors de l'affichage des modèles c'est l'inverse qui se produit.



Visualisation d'encodages (programme)

Register Login

1 carotte and (pomme or poire)
2

Logical CNF DIMACS LaTeX

carotte \wedge (pomme \vee poire)



Visualisation d'encodages (sortie FNC)

≡

Register Login

```
1 carotte and (pomme or not poire)
2
```

Logical	CNF	DIMACS	LaTeX
{ carotte }			
	{ pomme , -poire }		



Visualisation d'encodages (sortie DIMACS)

☰ Register Login

1 carotte and (pomme or not poire)
2

Logical	CNF	DIMACS	LaTeX
c poire 1			
c pomme 2			
c carotte 3			
p cnf 3 2			
3 0			
2 -1 0			

Les variables sont numérotées

Il y a 3 variables et 2 clauses

Les 2 clauses comme ensembles de littéraux
(-1 pour \neg poire, 0 termine chaque clause)

clic!



Plan

- 1** Introduction
- 2** Logique des propositions
- 3** Logique des prédictats
 - Introduction
 - Syntaxe
 - Sémantique
- 4** Logique modale propositionnelle



Logique des prédicats

Introduction



Où tout recommence...

		<i>LProp</i>	<i>LPred</i>	...
Syntaxe		✓	👉	
Sémantique		✓		
Modélisation		✓		
Méthode de preuve	• Déduction naturelle • Tableaux • Résolution			
Outils	• ToulIST	✓		



Expressivité limitée de *LProp*

- Soit le raisonnement suivant :
 - l'objet o_1 est bleu ; (b_1)
 - tout objet bleu est rond ; $(b_1 \rightarrow r_1)$
 - donc l'objet o_1 est rond ; (r_1)
- alors :



Expressivité limitée de *LProp*

- Soit le raisonnement suivant :
 - l'objet o_1 est bleu ; (b_1)
 - tout objet bleu est rond ; $(b_1 \rightarrow r_1)$
 - donc l'objet o_1 est rond ; (r_1)
- alors :
 - la taille de la modélisation croît proportionnellement avec le nombre d'objets (problème de concision) ;



Expressivité limitée de *LProp*

- Soit le raisonnement suivant :

- l'objet o_1 est bleu ; (b_1)

- tout objet bleu est rond ; $(b_1 \rightarrow r_1)$

- donc l'objet o_1 est rond ; (r_1)

- alors :

- la taille de la modélisation croît proportionnellement avec le nombre d'objets (problème de concision) ;

- on ne peut traduire « Tout objet bleu est rond » si l'ensemble des objets est infini (ou inconnu) ;



Expressivité limitée de *LProp*

- Soit le raisonnement suivant :
 - l'objet o_1 est bleu ; (b_1)
 - tout objet bleu est rond ; $(b_1 \rightarrow r_1)$
 - donc l'objet o_1 est rond ; (r_1)
- alors :
 - la taille de la modélisation croît proportionnellement avec le nombre d'objets (problème de concision) ;
 - on ne peut traduire « Tout objet bleu est rond » si l'ensemble des objets est infini (ou inconnu) ;
 - on ne peut distinguer l'objet (o_1, o_2, \dots) de sa propriété (« être bleu », « être rond », ...).



Idée de départ

- Rappel : une proposition est
 - un énoncé atomique (non décomposable) ;
 - décrivant une **propriété** (le verbe de la phrase) portant sur 1 (ensemble d')**entité(s)** (sujet [+ complément(s)] de la phrase).



Idée de départ

- Rappel : une proposition est
 - un énoncé atomique (non décomposable) ;
 - décrivant une **propriété** (le verbe de la phrase) portant sur 1 (ensemble d')**entité(s)** (sujet [+ complément(s)] de la phrase).
- Exemple : dans « Pipas boit du lait »
 - le **verbe** « boire » relie
 - « Pipas » (le **sujet** de la phrase)
 - avec du « lait » (le **complément d'objet direct**).



Idée de départ

- Rappel : une proposition est
 - un énoncé atomique (non décomposable) ;
 - décrivant une **propriété** (le verbe de la phrase) portant sur 1 (ensemble d')**entité(s)** (sujet [+ complément(s)] de la phrase).
- Exemple : dans « Pipas boit du lait »
 - le **verbe** « boire » relie
 - « Pipas » (le **sujet** de la phrase)
 - avec du « lait » (le **complément d'objet direct**).
- La logique des prédictats (*LPred*) va **raffiner** *LProp* en permettant de **décomposer syntaxiquement** une proposition en une propriété portant sur une ou plusieurs entité(s).



Conséquence de cette idée (prédictat, objets/individus)

- Langage naturel : « Si l'objet o_1 est bleu alors o_1 est rond »
(énoncé constitué de 2 propositions) ;



Conséquence de cette idée (prédictat, objets/individus)

- Langage naturel : « Si l'objet o_1 est bleu alors o_1 est rond »
(énoncé constitué de 2 propositions) ;
- $LProp : b_1 \rightarrow r_1$, où :
 - les **variables propositionnelles** b_1 et r_1 représentent des propositions (énoncés élémentaires, non décomposables) ;



Conséquence de cette idée (prédictat, objets/individus)

- Langage naturel : « Si l'objet o_1 est bleu alors o_1 est rond »
(énoncé constitué de 2 propositions) ;
- $LProp : b_1 \rightarrow r_1$, où :
 - les **variables propositionnelles** b_1 et r_1 représentent des **propositions** (énoncés élémentaires, non décomposables) ;
- $LPred : Bleu(o_1) \rightarrow Rond(o_1)$, où :
 - o_1 représente « l'objet o_1 » (le **sujet** des deux propositions) ;



Conséquence de cette idée (prédictat, objets/individus)

- Langage naturel : « Si l'objet o_1 est bleu alors o_1 est rond » (énoncé constitué de 2 propositions) ;
- $LProp : b_1 \rightarrow r_1$, où :
 - les **variables propositionnelles** b_1 et r_1 représentent des **propositions** (énoncés élémentaires, non décomposables) ;
- $LPred : Bleu(o_1) \rightarrow Rond(o_1)$, où :
 - o_1 représente « l'objet o_1 » (le **sujet** des deux propositions) ;
 - **Bleu** est le **prédictat** qui représente la propriété « est bleu » (le **verbe** de la **première** proposition) ;



Conséquence de cette idée (prédictat, objets/individus)

- Langage naturel : « Si l'objet o_1 est bleu alors o_1 est rond » (énoncé constitué de 2 propositions) ;
- $LProp : b_1 \rightarrow r_1$, où :
 - les **variables propositionnelles** b_1 et r_1 représentent des **propositions** (énoncés élémentaires, non décomposables) ;
- $LPred : Bleu(o_1) \rightarrow Rond(o_1)$, où :
 - o_1 représente « l'objet o_1 » (le **sujet** des deux propositions) ;
 - *Bleu* est le **prédictat** qui représente la propriété « est bleu » (le **verbe** de la **première** proposition) ;
 - *Rond* est le **prédictat** qui représente la propriété « est rond » (le **verbe** de la **seconde** proposition).



Conséquence de cette idée (quantification des objets)

- Langage naturel : « Tout objet bleu est rond » ;
- *LProp* :
 - pas exprimable en général (domaine infini ou non précisé) ;
 - possible sur domaine fini, mais en introduisant des connecteurs généralisés qui ne sont que des raccourcis d'écriture (la formule correspondante de *LProp* reste (très) grande) ;
- *LPred* : $\forall x. (Bleu(x) \rightarrow Rond(x))$.



Conséquence de cette idée (opérations/fonctions)

- Langage naturel : « Si le père de quelqu'un est grand alors ce dernier est également grand »
 - ▶ l'individu « le père de quelqu'un » est désigné à l'aide d'un autre individu (« quelqu'un ») ;



Conséquence de cette idée (opérations/fonctions)

- Langage naturel : « Si le père de quelqu'un est grand alors ce dernier est également grand »
 - ▶ l'individu « le père de quelqu'un » est désigné à l'aide d'un autre individu (« quelqu'un »);
- *LProp* : pas exprimable en général;



Conséquence de cette idée (opérations/fonctions)

- Langage naturel : « Si le père de quelqu'un est grand alors ce dernier est également grand »
 - ▶ l'individu « le père de quelqu'un » est désigné à l'aide d'un autre individu (« quelqu'un »);
- *LProp* : pas exprimable en général;
- *LPred* : $\forall x.(\text{Grand}(\text{pere}(x)) \rightarrow \text{Grand}(x))$, où
 - *x* désigne un **individu**;
 - *pere* est une **fonction** qui crée un **individu** *y* à partir d'un **autre individu** *x* (▶ *pere(x)* n'a **PAS** de valeur de vérité!);
 - *Grand* est un **prédictat** attribuant une propriété (vraie ou fausse) à un individu.



Conséquence de cette idée (sur la syntaxe de *LPred*)

- Dans la **logique propositionnelle**, une expression syntaxique est une **formule** (interprétée par vraie ou fausse) ;



Conséquence de cette idée (sur la syntaxe de *LPred*)

- Dans la **logique propositionnelle**, une expression syntaxique est une **formule** (interprétée par vraie ou fausse) ;
- dans la **logique des prédictats**, nous avons deux *catégories syntaxiques* :
 - les **termes** (qui représentent une entité : individu, objet, personne morale, etc.) ;
 - et les **formules** (interprétée par vraie ou fausse).



Logique des prédicats

Syntaxe



Les termes (3 types différents)

- Un terme peut représenter :
 - une **entité connue, identifiée**
(ex. l'objet o_1 , l'individu Tom , la société $SNCF$, etc.)
à l'aide d'une **constante** ;



Les termes (3 types différents)

- Un terme peut représenter :
 - une **entité connue, identifiée**
(ex. l'objet o_1 , l'individu Tom , la société $SNCF$, etc.)
à l'aide d'une **constante** ;
 - une **entité inconnue, quelconque**
(ex. un objet quelconque, un étudiant, une institution, etc.)
à l'aide d'une **variable** ;



Les termes (3 types différents)

■ Un terme peut représenter :

■ une **entité connue, identifiée**

(ex. l'objet o_1 , l'individu Tom , la société $SNCF$, etc.)
à l'aide d'une **constante** ;

■ une **entité inconnue, quelconque**

(ex. un objet quelconque, un étudiant, une institution, etc.)
à l'aide d'une **variable** ;

■ une **entité créé à partir d'une ou plusieurs autres entités**

(ex. l'original de l'objet o_1 , le père de Tom, le successeur de 3,
etc.)

à l'aide d'une **fonction**.



Les termes (définition inductive)

- Soit :
 - VAR , un ensemble de **variables** (d'entités) ;



Les termes (définition inductive)

- Soit :
 - VAR , un ensemble de **variables** (d'entités) ;
 - FON_n , un ensemble de **fonctions n -aires** (à $n \geq 0$ arguments) ;



Les termes (définition inductive)

■ Soit :

- VAR , un ensemble de **variables** (d'entités) ;
- FON_n , un ensemble de **fonctions n -aires** (à $n \geq 0$ arguments) ;
- FON_0 , l'ensemble des **constantes** (on écrit c au lieu de $c()$).



Les termes (définition inductive)

- Soit :
 - VAR , un ensemble de **variables** (d'entités) ;
 - FON_n , un ensemble de **fonctions n -aires** (à $n \geq 0$ arguments) ;
 - FON_0 , l'ensemble des **constantes** (on écrit c au lieu de $c()$) .
- L'ensemble $TERM$ des termes est défini comme le plus petit ensemble tel que :
 - 1 **variable** : si $x \in VAR$, alors $x \in TERM$;



Les termes (définition inductive)

- Soit :
 - VAR , un ensemble de **variables** (d'entités) ;
 - FON_n , un ensemble de **fonctions n -aires** (à $n \geq 0$ arguments) ;
 - FON_0 , l'ensemble des **constantes** (on écrit c au lieu de $c()$) .
- L'ensemble $TERM$ des termes est défini comme le plus petit ensemble tel que :
 - 1 **variable** : si $x \in VAR$, alors $x \in TERM$;
 - 2 **Application de fonction** : si $t_1 \in TERM, \dots, t_n \in TERM$ et $f \in FON_n$, alors $f(t_1, \dots, t_n) \in TERM$;



Les termes (définition inductive)

- Soit :
 - VAR , un ensemble de **variables** (d'entités) ;
 - FON_n , un ensemble de **fonctions n -aires** (à $n \geq 0$ arguments) ;
 - FON_0 , l'ensemble des **constantes** (on écrit c au lieu de $c()$) .
- L'ensemble $TERM$ des termes est défini comme le plus petit ensemble tel que :
 - 1 **variable** : si $x \in VAR$, alors $x \in TERM$;
 - 2 **Application de fonction** : si $t_1 \in TERM, \dots, t_n \in TERM$ et $f \in FON_n$, alors $f(t_1, \dots, t_n) \in TERM$;
 - ▶ en particulier, toute **constante** est un terme.



Les termes (exemples)

■ Soit $f \in FON_2$, $x, y \in VAR$, $g \in FON_1$, $\pi \in FON_0$, alors :

■ $f(x, \pi) \in TERM$ car :

■ $x, \pi \in TERM$;

■ comme $f \in FON_2$, alors $f(x, \pi) \in TERM$;

□



Les termes (exemples)

■ Soit $f \in FON_2$, $x, y \in VAR$, $g \in FON_1$, $\pi \in FON_0$, alors :

■ $f(x, \pi) \in TERM$ car :

■ $x, \pi \in TERM$;

■ comme $f \in FON_2$, alors $f(x, \pi) \in TERM$; □

■ $f(x, g(f(y, \pi))) \in TERM$ car :

■ $y, \pi \in TERM$;

■ comme $f \in FON_2$, alors $f(y, \pi) \in TERM$;

■ comme $g \in FON_1$, alors $g(f(y, \pi)) \in TERM$;

■ $x \in TERM$;

■ comme $f \in FON_2$, alors $f(x, g(f(y, \pi))) \in TERM$. □



Les termes (le monde de mes voisins)

- Mes voisins sont 3 : Tom, Bob et Léa ;
- comment décrire le père de Tom ? la fille de Bob et Léa ?
 - constantes : $t, b, l \in FON_0$;
 - fonctions :
 - $pere \in FON_1$;
 - $fille \in FON_2$;
 - termes : $t, b, l, pere(t), fille(b, l) \in TERM$.



Les formules atomiques

- Une formule atomique
 - est la plus petite unité **indécomposable** (« atomique ») de *LPred* ayant une **valeur de vérité** ;
 - exprime qu'une **propriété** met **ou non** en relation des **entités**.



Les formules atomiques

- Une formule atomique
 - est la plus petite unité **indécomposable** (« atomique ») de *LPred* ayant une **valeur de vérité** ;
 - exprime qu'une **propriété** met **ou non** en relation des **entités**.
- Par exemple, dans « Bob est grand » :
 - « Bob » est une **entité** (ici, un individu) ;
 - « est grand » exprime une **propriété** (celle d'être grand) à propos d'une entité ;



Les formules atomiques

- Une formule atomique
 - est la plus petite unité **indécomposable** (« atomique ») de *LPred* ayant une **valeur de vérité** ;
 - exprime qu'une **propriété** met **ou non** en relation des **entités**.
- Par exemple, dans « Bob est grand » :
 - « Bob » est une **entité** (ici, un individu) ;
 - « est grand » exprime une **propriété** (celle d'être grand) à propos d'une entité ;
 - en **appliquant** cette propriété à cette entité particulière, on exprime qu'on attribue la propriété d'être grand à **Bob**, ce qui peut être **vrai ou faux**.



Les formules atomiques (définition formelle)

- Formellement :
 - une **propriété** est représentée par un **prédictat**;
 - les **entités** sont représentées par des **termes**.
- Soit $PRED_n$ un ensemble de prédictat n -aires (à n arguments), alors :
 - si $t_1 \in TERM, \dots, t_n \in TERM$ et $P \in PRED_n$, alors $P(t_1, \dots, t_n)$ est une **formule atomique**;
 - $T, \perp \in PRED_0$ sont des formules atomiques dont le prédictat n'a pas d'argument.



Les formules atomiques (retour au monde de mes voisins)

- Dans « La fille de Bob et Léa est musicienne » :
 - les individus Bob et Léa sont représentés par des constantes (termes) ;
 - « la fille de » est représenté par une fonction (terme) ;
 - « est musicienne » est représenté par un prédictat ;
 - soit : $EstMusicien(fille(b, l))$.
- « Tom est plus jeune que Bob » s'exprime par :
 - $EstPlusJeune(t, b)$.



Les formules (définition inductive)

- L'ensemble $FORM$ des formules est défini comme le plus petit ensemble tel que ($PRED_n$ un ensemble de prédicat n -aires) :
 - 1 (prédicat) si $t_1 \in TERM, \dots, t_n \in TERM$ et $P \in PRED_n$ alors
 $P(t_1, \dots, t_n) \in FORM$;



Les formules (définition inductive)

- L'ensemble $FORM$ des formules est défini comme le plus petit ensemble tel que ($PRED_n$ un ensemble de prédicat n -aires) :
 - 1 (prédicat) si $t_1 \in TERM, \dots, t_n \in TERM$ et $P \in PRED_n$ alors
 $P(t_1, \dots, t_n) \in FORM$;
 - 2 (négation) si $A \in FORM$, alors $(\neg A) \in FORM$;



Les formules (définition inductive)

- L'ensemble $FORM$ des formules est défini comme le plus petit ensemble tel que ($PRED_n$ un ensemble de prédicat n -aires) :

- 1 (prédicat) si $t_1 \in TERM, \dots, t_n \in TERM$ et $P \in PRED_n$ alors
 $P(t_1, \dots, t_n) \in FORM$;
- 2 (négation) si $A \in FORM$, alors $(\neg A) \in FORM$;
- 3 (connecteurs binaires) Si $A \in FORM$ et $B \in FORM$ alors
 $(A \wedge B) \in FORM, (A \vee B) \in FORM, (A \rightarrow B) \in FORM$;



Les formules (définition inductive)

- L'ensemble $FORM$ des formules est défini comme le plus petit ensemble tel que ($PRED_n$ un ensemble de prédicat n -aires) :
 - 1 (prédicat) si $t_1 \in TERM, \dots, t_n \in TERM$ et $P \in PRED_n$ alors $P(t_1, \dots, t_n) \in FORM$;
 - 2 (négation) si $A \in FORM$, alors $(\neg A) \in FORM$;
 - 3 (connecteurs binaires) Si $A \in FORM$ et $B \in FORM$ alors $(A \wedge B) \in FORM, (A \vee B) \in FORM, (A \rightarrow B) \in FORM$;
 - 4 (quantificateur universel « pour tout ») Si $A \in FORM$ et $x \in VAR$, alors $(\forall x.A) \in FORM$;



Les formules (définition inductive)

- L'ensemble $FORM$ des formules est défini comme le plus petit ensemble tel que ($PRED_n$ un ensemble de prédicat n -aires) :
 - 1 (prédicat) si $t_1 \in TERM, \dots, t_n \in TERM$ et $P \in PRED_n$ alors $P(t_1, \dots, t_n) \in FORM$;
 - 2 (négation) si $A \in FORM$, alors $(\neg A) \in FORM$;
 - 3 (connecteurs binaires) Si $A \in FORM$ et $B \in FORM$ alors $(A \wedge B) \in FORM, (A \vee B) \in FORM, (A \rightarrow B) \in FORM$;
 - 4 (quantificateur universel « pour tout ») Si $A \in FORM$ et $x \in VAR$, alors $(\forall x.A) \in FORM$;
 - 5 (quantificateur existentiel « il existe ») Si $A \in FORM$ et $x \in VAR$, alors $(\exists x.A) \in FORM$.



Les formules (retour au monde des voisins)

- « Il existe une personne musicienne et non sportive » :
$$\left(\exists x. \left(EstMusicien(x) \wedge (\neg EstSportif(x)) \right) \right);$$



Les formules (retour au monde des voisins)

- « Il existe une personne musicienne et non sportive » :
$$\left(\exists x. \left(EstMusicien(x) \wedge (\neg EstSportif(x)) \right) \right);$$
- « Toute personne est plus jeune que son père » :
$$\left(\forall x. EstPlusJeune(x, pere(x)) \right);$$



Les formules (retour au monde des voisins)

- « Il existe une personne musicienne et non sportive » :
$$\left(\exists x. \left(EstMusicien(x) \wedge (\neg EstSportif(x)) \right) \right);$$
- « Toute personne est plus jeune que son père » :
$$\left(\forall x. EstPlusJeune(x, pere(x)) \right);$$
- « Toute personne musicienne et sportive a un père musicien ou sportif » :
$$\left(\forall x. \left(EstMusicien(x) \wedge EstSportif(x) \right) \rightarrow \left(EstMusicien(pere(x)) \vee EstSportif(pere(x)) \right) \right)$$



Fonctions vs prédictats (utilisation)

- Les **fonctions** permettent de désigner des entités (objets, individus, etc.) complexes, créés (obtenus) à partir d'autres entités :
 - « une tarte aux pommes » : `tarte(pomme)` ;
 - « le père de Tom » : `pere(tom)`.



Fonctions vs prédictats (utilisation)

- Les **fonctions** permettent de désigner des entités (objets, individus, etc.) complexes, créés (obtenus) à partir d'autres entités :
 - « une tarte aux pommes » : *tarte(pomme)* ;
 - « le père de Tom » : *pere(tom)*.
- Les **prédictats** permettent de décrire une situation, sous la forme d'une propriété qui met des entités en relation :
 - « Pipa achète une pomme » : *Acheter(pipa, pomme)* ;
 - « Fatou mange une tarte aux pommes » :
Mange(fatou, tarte(pomme)) ;
 - « Hien est plus jeune que Bob » : *EstPlusJeune(hien, bob)*.



Fonctions vs prédictats (attention)

- Les arguments d'un prédictat sont des termes, pas des formules ;



Fonctions vs prédictats (attention)

- Les **arguments d'un prédictat** sont des termes, **pas** des formules ;
- « Fatou mange la pomme qu'elle a achetée » :
 - $Mange(fatou, Achete(fatou, pomme))$ est **INCORRECT** ;
 - $Achete(fatou, pomme) \wedge Mange(fatou, pomme)$ est **CORRECT**.



Fonctions vs prédictats (attention)

- Les **arguments d'un prédictat** sont des termes, **pas** des formules ;
- « Fatou mange la pomme qu'elle a achetée » :
 - *Mange(fatou, Achete(fatou, pomme))* est **INCORRECT** ;
 - *Achete(fatou, pomme) \wedge Mange(fatou, pomme)* est **CORRECT**.
- **Convention** :
 - nom de fonctions : en minuscules ;
 - nom de prédictats : commence par une *Majuscule* ;
 - ▶ la police de caractères est également différent.



Conventions d'écriture

- La formule $\forall x.A \wedge B$ se lit-elle : $(\forall x.A) \wedge B$ ou $\forall x.(A \wedge B)$?



Conventions d'écriture

- La formule $\forall x.A \wedge B$ se lit-elle : $(\forall x.A) \wedge B$ ou $\forall x.(A \wedge B)$?
- **Convention** : le quantificateur porte sur la plus grande formule possible à partir du point après la variable quantifiée :
 - dans l'exemple ci-dessus, il faut comprendre $\forall x.(A \wedge B)$.



Conventions d'écriture

- La formule $\forall x.A \wedge B$ se lit-elle : $(\forall x.A) \wedge B$ ou $\forall x.(A \wedge B)$?
- **Convention** : le quantificateur porte sur la plus grande formule possible à partir du point après la variable quantifiée :
 - dans l'exemple ci-dessus, il faut comprendre $\forall x.(A \wedge B)$.
- Attention :
 - $((\forall x.A) \rightarrow (\forall y.B))$ signifie $(\forall x.A) \rightarrow \forall y.B$;
 - $(\forall x.(A \rightarrow (\forall y.B)))$ signifie $\forall x.A \rightarrow \forall y.B$
 - $A \wedge \forall x.B$ est **différents** de $\forall x.B \wedge A$!!!



Limites de l'expressivité (contournables)

- *L_{Pred}* pas toujours adaptée pour modéliser certaines propriétés ;
- dans certains cas, on peut y arriver en introduisant des prédictats spécifiques ;
- par exemple, temporels :
 - « Si le composant c_1 tombe en panne, il émet un signal » ;
 - peut être codé par :

$$\forall t. \text{EstEnPanneAuTemps}(c_1, t) \rightarrow (\exists t'. (t' > t) \wedge \text{EmetUnSignalAuTemps}(c_1, t'))$$



Limites de l'expressivité (essentielles)

- Certaines limites sont **incontournable** ;
- par exemple, utiliser des **fonctions** comme objets...
 - en les **quantifiant** : « Toute fonction a une inverse » :

$$\forall f. \exists g. \forall x. EstEgalA(f(g(x)), x)$$

(➡ interdit)



Limites de l'expressivité (essentielles)

- Certaines limites sont **incontournable** ;
- par exemple, utiliser des **fonctions** comme objets...
 - en les **quantifiant** : « Toute fonction a une inverse » :

$$\forall f. \exists g. \forall x. \text{EstEgalA}(f(g(x)), x)$$

(➡ interdit)

- en les utilisant comme **argument** :

$$\forall f. \forall x. \text{EstEgalA}(f(\text{inverse}(f)(x)), x)$$

(➡ interdit)



Limites de l'expressivité (essentielles)

- Certaines limites sont **incontournable** ;
- par exemple, utiliser des **fonctions** comme objets...
 - en les **quantifiant** : « Toute fonction a une inverse » :

$$\forall f. \exists g. \forall x. \text{EstEgalA}(f(g(x)), x) \quad (\rightarrow \text{interdit})$$

- en les utilisant comme **argument** :

$$\forall f. \forall x. \text{EstEgalA}(f(\text{inverse}(f)(x)), x) \quad (\rightarrow \text{interdit})$$

- ... ou quantifier sur des **prédicts** ou des **ensembles** ;



Limites de l'expressivité (essentielles)

- Certaines limites sont **incontournable** ;
- par exemple, utiliser des **fonctions** comme objets...
 - en les **quantifiant** : « Toute fonction a une inverse » :

$$\forall f. \exists g. \forall x. \text{EstEgalA}(f(g(x)), x)$$

(➡ interdit)

- en les utilisant comme **argument** :

$$\forall f. \forall x. \text{EstEgalA}(f(\text{inverse}(f)(x)), x)$$

(➡ interdit)

- ... ou quantifier sur des **prédicts** ou des **ensembles** ;
- ➡ *L^{Pred}* est **inappropriée** pour raisonner sur certains programmes fonctionnels.



Variables libres et variables liées (définition)

- Toute occurrence d'une variable dans une formule est soit **liée**, soit **libre**, soit **liante**;



Variables libres et variables liées (définition)

- Toute occurrence d'une variable dans une formule est soit **liée**, soit **libre**, soit **liante**;
- par exemple, dans

$$\left(Q(x) \vee \exists x. \forall y. P(f(x), z) \wedge Q(y) \right) \vee \forall x. R(x, z, g(x))$$

il y a :



Variables libres et variables liées (définition)

- Toute occurrence d'une variable dans une formule est soit **liée**, soit **libre**, soit **liante**;
- par exemple, dans

$$\left(Q(x) \vee \exists x. \forall y. P(f(\textcolor{red}{x}), z) \wedge Q(\textcolor{red}{y}) \right) \vee \forall x. R(\textcolor{red}{x}, z, g(\textcolor{red}{x}))$$

il y a :

- des **variables liées** ;



Variables libres et variables liées (définition)

- Toute occurrence d'une variable dans une formule est soit **liée**, soit **libre**, soit **liante**;
- par exemple, dans

$$\left(Q(\textcolor{green}{x}) \vee \exists x. \forall y. P(f(\textcolor{red}{x}), \textcolor{green}{z}) \wedge Q(\textcolor{red}{y}) \right) \vee \forall x. R(\textcolor{red}{x}, \textcolor{green}{z}, g(\textcolor{red}{x}))$$

il y a :

- des **variables liées** ;
- des **variables libres** ;



Variables libres et variables liées (définition)

- Toute occurrence d'une variable dans une formule est soit **liée**, soit **libre**, soit **liante**;
- par exemple, dans

$$\left(Q(\textcolor{green}{x}) \vee \exists \textcolor{blue}{x}. \forall \textcolor{blue}{y}. P(f(\textcolor{red}{x}), \textcolor{green}{z}) \wedge Q(\textcolor{red}{y}) \right) \vee \forall \textcolor{blue}{x}. R(\textcolor{red}{x}, \textcolor{green}{z}, g(\textcolor{red}{x}))$$

il y a :

- des **variables liées** ;
- des **variables libres** ;
- des **variables liantes** ;



Variables libres et variables liées (définition)

- Toute occurrence d'une variable dans une formule est soit liée, soit libre, soit liante;
- par exemple, dans

$$\left(Q(\textcolor{green}{x}) \vee \exists \textcolor{blue}{x}. \forall \textcolor{blue}{y}. P(f(\textcolor{red}{x}), \textcolor{green}{z}) \wedge Q(\textcolor{red}{y}) \right) \vee \forall \textcolor{blue}{x}. R(\textcolor{red}{x}, \textcolor{green}{z}, g(\textcolor{red}{x}))$$

il y a :

- des variables liées ;
- des variables libres ;
- des variables liantes ;
- une formule sans occurrence libre de variables est close.



Variables libres et variables liées (remarques)

- La quantification dans L_{Pred} est sujette aux mêmes règles (de portée, renommage, etc.) que celle sur des ensembles finis (L_{Prop}).



Variables libres et variables liées (remarques)

- La quantification dans L_{Pred} est sujette aux **mêmes règles** (de portée, renommage, etc.) que celle sur des ensembles finis (L_{Prop}).
- Si on compare les 2 formules suivantes :

$$\bigwedge_{i \in E} r_{i,j} \wedge \bigvee_{k \in E} s_{i,k} \quad (\text{formule de } L_{Prop})$$
$$\forall i.r(i, j) \wedge \exists k.s(i, k) \quad (\text{formule de } L_{Pred})$$



Variables libres et variables liées (remarques)

- La quantification dans L_{Pred} est sujette aux **mêmes règles** (de portée, renommage, etc.) que celle sur des ensembles finis (L_{Prop}).
- Si on compare les 2 formules suivantes :

$$\begin{array}{ll} \bigwedge_{i \in E} r_{i,j} \wedge \bigvee_{k \in E} s_{i,k} & (\text{formule de } L_{Prop}) \\ \forall i.r(i,j) \wedge \exists k.s(i,k) & (\text{formule de } L_{Pred}) \end{array}$$

- une formule quantifiée sur des ensembles finis n'est **pas bien définie dans L_{Prop}** si elle contient des **indices libres** ;



Variables libres et variables liées (remarques)

- La quantification dans $LPred$ est sujette aux **mêmes règles** (de portée, renommage, etc.) que celle sur des ensembles finis ($LProp$).
- Si on compare les 2 formules suivantes :

$$\begin{array}{ll} \bigwedge_{i \in E} r_{i,j} \wedge \bigvee_{k \in E} s_{i,k} & (\text{formule de } LProp) \\ \forall i.r(i,j) \wedge \exists k.s(i,k) & (\text{formule de } LPred) \end{array}$$

- une formule quantifiée sur des ensembles finis n'est **pas bien définie dans $LProp$** si elle contient des **indices libres** ;
- mais une formule de $LPred$ **peut contenir des variables libres**.



Logique des prédicats

Sémantique



Où on va donner du sens aux formules...

		<i>LProp</i>	<i>LPred</i>	...
Syntaxe		✓	✓	
Sémantique		✓	👉	
Modélisation		✓	👉	
Méthode de preuve	<ul style="list-style-type: none"> • Déduction naturelle • Tableaux • Résolution 			
Outils	<ul style="list-style-type: none"> • ToulIST 	✓		



Sémantique de la logique propositionnelle (rappel)

- Dans le cadre de la logique propositionnelle :
 - un fonction de valuation v détermine la valeur de vérité des variables propositionnelles d'une formule :

$$v(p) = 0 \text{ et } v(q) = 1 \quad (\text{exemple pour 2 variables})$$



Sémantique de la logique propositionnelle (rappel)

- Dans le cadre de la logique propositionnelle :
 - un fonction de valuation v détermine la valeur de vérité des variables propositionnelles d'une formule :

$$v(p) = 0 \text{ et } v(q) = 1 \quad (\text{exemple pour 2 variables})$$

- une fonction d'interprétation I_v étend la valuation v à une formule :

$$I_v(\neg p \wedge q) = 1 \quad (\text{exemple de formule})$$



Ce qui change dans la logique des prédictats

- La logique des prédictats est plus complexe, puisqu'elle fait la distinction entre :
 - les **termes** (qui n'ont pas de valeur de vérité) ;



Ce qui change dans la logique des prédictats

- La logique des prédictats est plus complexe, puisqu'elle fait la distinction entre :
 - les **termes** (qui n'ont pas de valeur de vérité) ;
 - les **formules atomiques** qui sont les formules les plus simples ayant une valeur de vérité (qui sont à *LPred* ce que les variables propositionnelles sont à *LProp*) ;



Ce qui change dans la logique des prédictats

- La logique des prédictats est plus complexe, puisqu'elle fait la distinction entre :
 - les **termes** (qui n'ont pas de valeur de vérité) ;
 - les **formules atomiques** qui sont les formules les plus simples ayant une valeur de vérité (qui sont à *LPred* ce que les variables propositionnelles sont à *LProp*) ;
 - les **formules** qui correspondent à des formules atomiques reliées à l'aide de connecteurs logiques ;



Ce qui change dans la logique des prédictats

- La logique des prédictats est plus complexe, puisqu'elle fait la **distinction** entre :
 - les **termes** (qui n'ont pas de valeur de vérité) ;
 - les **formules atomiques** qui sont les formules les plus simples ayant une valeur de vérité (qui sont à *LPred* ce que les variables propositionnelles sont à *LProp*) ;
 - les **formules** qui correspondent à des formules atomiques reliées à l'aide de connecteurs logiques ;
- et elle inclut également des **quantificateurs** (qu'il va falloir correctement interpréter).



Interprétation des termes (exemple)

- Quelle est la signification du terme $f(x, y)$?



Interprétation des termes (exemple)

- Quelle est la signification du terme $f(x, y)$?

■ Si $\begin{cases} x \text{ représente } 2 \\ y \text{ représente } 3 \\ f \text{ représente la fonction d'addition } + \end{cases}$



Interprétation des termes (exemple)

- Quelle est la signification du terme $f(x, y)$?

■ Si $\begin{cases} x \text{ représente } 2 \\ y \text{ représente } 3 \\ f \text{ représente la fonction d'addition } + \end{cases}$
alors $f(x, y)$ représente $2 + 3$, donc 5 .



Interprétation des termes (exemple)

- Quelle est la signification du terme $f(x, y)$?

■ Si $\begin{cases} x \text{ représente } 2 \\ y \text{ représente } 3 \\ f \text{ représente la fonction d'addition } + \end{cases}$
alors $f(x, y)$ représente $2 + 3$, donc 5 .

■ Si $\begin{cases} x \text{ représente } 2 \\ y \text{ représente } 3 \\ f \text{ représente la fonction de multiplication } \times \end{cases}$



Interprétation des termes (exemple)

- Quelle est la signification du terme $f(x, y)$?

■ Si $\begin{cases} x \text{ représente } 2 \\ y \text{ représente } 3 \\ f \text{ représente la fonction d'addition } + \end{cases}$
alors $f(x, y)$ représente $2 + 3$, donc 5 .

■ Si $\begin{cases} x \text{ représente } 2 \\ y \text{ représente } 3 \\ f \text{ représente la fonction de multiplication } \times \end{cases}$
alors $f(x, y)$ représente 2×3 , donc 6 .



Interprétation des termes (valuation des variables et fonctions)

- Soit un **domaine d'interprétation** \mathcal{D} (ensemble non vide) ;
- alors on définit :
 - une **valuation pour les variables** :

$$v_{VAR} : VAR \longrightarrow \mathcal{D}$$

- un **ensemble de valuations pour les fonctions** d'arité au plus n :

$$\left\{ v_{FON_n} : FON_n \longrightarrow (\mathcal{D}^n \longrightarrow \mathcal{D}) \right\}_{n \in \mathbb{N}}$$



Interprétation des termes (exemple : valuations)

- Domaine $\mathcal{D} = \mathbb{N}$;



Interprétation des termes (exemple : valuations)

- Domaine $\mathcal{D} = \mathbb{N}$;
- la valuation des variables ($v_{VAR} : VAR \longrightarrow \mathbb{N}$) est telle que :
 - $v_{VAR}(x) = 2$ et $v_{VAR}(y) = 3$;
 - qu'on note par commodité : $v_{VAR} = [x \mapsto 2, y \mapsto 3]$;



Interprétation des termes (exemple : valuations)

- Domaine $\mathcal{D} = \mathbb{N}$;
- la valuation des variables ($v_{VAR} : VAR \longrightarrow \mathbb{N}$) est telle que :
 - $v_{VAR}(x) = 2$ et $v_{VAR}(y) = 3$;
 - qu'on note par commodité : $v_{VAR} = [x \mapsto 2, y \mapsto 3]$;
- la valuation de fonctions binaires ($v_{FON_2} : FON_2 \longrightarrow (\mathbb{N}^2 \longrightarrow \mathbb{N})$) :
 - $v_{FON_2}(f)$ est l'addition $+$ et $v_{FON_2}(g)$ est la multiplication \times ;
 - qu'on note par commodité $v_{FON_2} = [f \mapsto +, g \mapsto \times]$;



Interprétation des termes (exemple : valuations)

- Domaine $\mathcal{D} = \mathbb{N}$;
- la valuation des variables ($v_{VAR} : VAR \rightarrow \mathbb{N}$) est telle que :
 - $v_{VAR}(x) = 2$ et $v_{VAR}(y) = 3$;
 - qu'on note par commodité : $v_{VAR} = [x \mapsto 2, y \mapsto 3]$;
- la valuation de fonctions binaires ($v_{FON_2} : FON_2 \rightarrow (\mathbb{N}^2 \rightarrow \mathbb{N})$) :
 - $v_{FON_2}(f)$ est l'addition $+$ et $v_{FON_2}(g)$ est la multiplication \times ;
 - qu'on note par commodité $v_{FON_2} = [f \mapsto +, g \mapsto \times]$;
- ➡ On va montrer que le terme $f(x, g(x, y))$ est interprété à l'aide de ces valuations par $2 + (2 \times 3) = 8$ (cf. Diapo 326).



Interprétation des termes (définition formelle)

- L'interprétation des termes est l'extension I_v des fonctions de valuation v_{VAR} et v_{FON_n} à tout terme ;



Interprétation des termes (définition formelle)

- L'interprétation des termes est l'extension I_v des fonctions de valuation v_{VAR} et v_{FON_n} à tout terme ;
- formellement, I_v est définie par le triplet

$$I_v = (\mathcal{D}, v_{VAR}, \{v_{FON_n}\}_{n \in \mathbb{N}})$$

comme suit :



Interprétation des termes (définition formelle)

- L'interprétation des termes est l'extension I_v des fonctions de valuation v_{VAR} et v_{FON_n} à tout terme ;
- formellement, I_v est définie par le triplet

$$I_v = (\mathcal{D}, v_{VAR}, \{v_{FON_n}\}_{n \in \mathbb{N}})$$

comme suit :

- $I_v(x) = v_{VAR}(x)$ pour tout $x \in VAR$; (Variables)



Interprétation des termes (définition formelle)

- L'interprétation des termes est l'extension I_v des fonctions de valuation v_{VAR} et v_{FON_n} à tout terme ;
- formellement, I_v est définie par le triplet

$$I_v = (\mathcal{D}, v_{VAR}, \{v_{FON_n}\}_{n \in \mathbb{N}})$$

comme suit :

■ $I_v(x) = v_{VAR}(x)$ pour tout $x \in VAR$; (Variables)

■ $I_v(f(t_1, \dots, t_n)) = v_{FON_n}(f)(I_v(t_1), \dots, I_v(t_n))$ (Fonctions)

pour toute fonction $f \in FON_n$ et $t_1 \dots t_n \in TERM$.



Interprétation des termes (exemple : interprétation)

- Dans l'exemple précédent, on avait :

$$I_v = (\mathbb{N}, v_{VAR} = [x \mapsto 2, y \mapsto 3], \{v_{FON_2} = [f \mapsto +, g \mapsto \times]\})$$



Interprétation des termes (exemple : interprétation)

- Dans l'exemple précédent, on avait :

$$I_v = (\mathbb{N}, v_{VAR} = [x \mapsto 2, y \mapsto 3], \{v_{FON_2} = [f \mapsto +, g \mapsto \times]\})$$

d'où : $I_v(f(x, g(x, y))) = v_{FON_2}(f)(I_v(x), I_v(g(x, y)))$



Interprétation des termes (exemple : interprétation)

- Dans l'exemple précédent, on avait :

$$I_v = (\mathbb{N}, v_{VAR} = [x \mapsto 2, y \mapsto 3], \{v_{FON_2} = [f \mapsto +, g \mapsto \times]\})$$

$$\begin{aligned} \text{d'où : } I_v(f(x, g(x, y))) &= v_{FON_2}(f)(I_v(x), I_v(g(x, y))) \\ &= +\left(v_{VAR}(x), v_{FON_2}(g)(I_v(x), I_v(y))\right) \end{aligned}$$



Interprétation des termes (exemple : interprétation)

- Dans l'exemple précédent, on avait :

$$I_v = (\mathbb{N}, v_{VAR} = [x \mapsto 2, y \mapsto 3], \{v_{FON_2} = [f \mapsto +, g \mapsto \times]\})$$

$$\begin{aligned} d'où : I_v(f(x, g(x, y))) &= v_{FON_2}(f)(I_v(x), I_v(g(x, y))) \\ &= +\left(v_{VAR}(x), v_{FON_2}(g)(I_v(x), I_v(y))\right) \\ &= +(2, \times(v_{VAR}(x), v_{VAR}(y))) \end{aligned}$$



Interprétation des termes (exemple : interprétation)

- Dans l'exemple précédent, on avait :

$$I_v = (\mathbb{N}, v_{VAR} = [x \mapsto 2, y \mapsto 3], \{v_{FON_2} = [f \mapsto +, g \mapsto \times]\})$$

$$\begin{aligned} d'où : I_v(f(x, g(x, y))) &= v_{FON_2}(f)(I_v(x), I_v(g(x, y))) \\ &= +(v_{VAR}(x), v_{FON_2}(g)(I_v(x), I_v(y))) \\ &= +(2, \times(v_{VAR}(x), v_{VAR}(y))) \\ &= +(2, \times(2, 3)) \end{aligned}$$



Interprétation des termes (exemple : interprétation)

- Dans l'exemple précédent, on avait :

$$I_v = (\mathbb{N}, v_{VAR} = [x \mapsto 2, y \mapsto 3], \{v_{FON_2} = [f \mapsto +, g \mapsto \times]\})$$

$$\begin{aligned}
 d'où : I_v(f(x, g(x, y))) &= v_{FON_2}(f)(I_v(x), I_v(g(x, y))) \\
 &= +(v_{VAR}(x), v_{FON_2}(g)(I_v(x), I_v(y))) \\
 &= +(2, \times(v_{VAR}(x), v_{VAR}(y))) \\
 &= +(2, \times(2, 3)) \\
 &= 2 + 2 \times 3 \quad \text{(notation infixée)}
 \end{aligned}$$



Interprétation des termes (exemple : interprétation)

- Dans l'exemple précédent, on avait :

$$I_v = (\mathbb{N}, v_{VAR} = [x \mapsto 2, y \mapsto 3], \{v_{FON_2} = [f \mapsto +, g \mapsto \times]\})$$

d'où : $I_v(f(x, g(x, y))) = v_{FON_2}(f)(I_v(x), I_v(g(x, y)))$

$$\begin{aligned} &= +\left(v_{VAR}(x), v_{FON_2}(g)(I_v(x), I_v(y))\right) \\ &= +(2, \times(v_{VAR}(x), v_{VAR}(y))) \\ &= +(2, \times(2, 3)) \\ &= 2 + 2 \times 3 && \text{(notation infixée)} \\ &= 8 \end{aligned}$$

□



Interprétation des formules (intuitions)

- Quelle est la **signification** de la **formule atomique** $P(x)$?
- exemples d'interprétation :
 - si x représente **2** et P représente la propriété « être un nombre **pair** », alors $P(x)$ est une proposition vraie ;



Interprétation des formules (intuitions)

- Quelle est la **signification** de la **formule atomique** $P(x)$?
- exemples d'interprétation :
 - si x représente **2** et P représente la propriété « être un nombre **pair** », alors $P(x)$ est une proposition vraie ;
 - si x représente **π** et P représente la propriété « être un nombre **rationnel** », alors $P(x)$ est une proposition fausse ;



Interprétation des formules (valuation des prédictats)

- Une fonction de valuation des prédictats associe à chaque prédictat n -aire ($n \in \mathbb{N}$) une fonction n -aire dans $\{0, 1\}$:
 - $v_{PRED_n} : PRED_n \rightarrow (\mathcal{D}^n \rightarrow \{0, 1\})$ est une fonction de valuation pour les prédictats n -aires ;



Interprétation des formules (valuation des prédictats)

- Une fonction de valuation des prédictats associe à chaque prédictat n -aire ($n \in \mathbb{N}$) une fonction n -aire dans $\{0, 1\}$:
 - $v_{PRED_n} : PRED_n \rightarrow (\mathcal{D}^n \rightarrow \{0, 1\})$ est une fonction de valuation pour les prédictats n -aires ;
- cette fonction étend la fonction d'interprétation des termes afin de pouvoir interpréter les formules atomiques.



Interprétation des formules (définition formelle)

- La fonction d'**interprétation des formules** est le quadruplet

$$I_v = (\mathcal{D}, v_{VAR}, \{v_{FON_n}\}_{n \in \mathbb{N}}, \{v_{PRED_n}\}_{n \in \mathbb{N}}) \quad \text{où :}$$



Interprétation des formules (définition formelle)

- La fonction d'**interprétation des formules** est le quadruplet

$$I_v = (\mathcal{D}, v_{VAR}, \{v_{FON_n}\}_{n \in \mathbb{N}}, \{v_{PRED_n}\}_{n \in \mathbb{N}}) \quad \text{où :}$$

- $I_v(P(t_1, \dots, t_n)) = v_{PRED_n}(P)(I_v(t_1), \dots, I_v(t_n));$ (prédicat)



Interprétation des formules (définition formelle)

- La fonction d'**interprétation des formules** est le quadruplet

$$I_v = (\mathcal{D}, v_{VAR}, \{v_{FON_n}\}_{n \in \mathbb{N}}, \{v_{PRED_n}\}_{n \in \mathbb{N}}) \quad \text{où :}$$

- $I_v(P(t_1, \dots, t_n)) = v_{PRED_n}(P)(I_v(t_1), \dots, I_v(t_n));$ (prédictat)
- négation et connecteurs binaires : comme pour $LProp$;



Interprétation des formules (définition formelle)

- La fonction d'interprétation des formules est le quadruplet

$$I_v = (\mathcal{D}, v_{VAR}, \{v_{FON_n}\}_{n \in \mathbb{N}}, \{v_{PRED_n}\}_{n \in \mathbb{N}}) \quad \text{où :}$$

- $I_v(P(t_1, \dots, t_n)) = v_{PRED_n}(P)(I_v(t_1), \dots, I_v(t_n));$ (prédicat)
- négation et connecteurs binaires : comme pour $LProp$;
- $I_v(\forall x.A) = 1$ ssi $I_{v[x:=d]}(A) = 1$ pour tout $d \in \mathcal{D};$ (quantif. univ.)



Interprétation des formules (définition formelle)

- La fonction d'interprétation des formules est le quadruplet

$$I_v = (\mathcal{D}, v_{VAR}, \{v_{FON_n}\}_{n \in \mathbb{N}}, \{v_{PRED_n}\}_{n \in \mathbb{N}}) \quad \text{où :}$$

- $I_v(P(t_1, \dots, t_n)) = v_{PRED_n}(P)(I_v(t_1), \dots, I_v(t_n))$; (prédicat)
- négation et connecteurs binaires : comme pour $LProp$;
- $I_v(\forall x.A) = 1$ ssi $I_{v[x:=d]}(A) = 1$ pour tout $d \in \mathcal{D}$; (quantif. univ.)
- $I_v(\exists x.A) = 1$ ssi $I_{v[x:=d]}(A) = 1$ pour quelque $d \in \mathcal{D}$; (quantif. exist.)



Interprétation des formules (définition formelle)

- La fonction d'interprétation des formules est le quadruplet

$$I_v = (\mathcal{D}, v_{VAR}, \{v_{FON_n}\}_{n \in \mathbb{N}}, \{v_{PRED_n}\}_{n \in \mathbb{N}}) \quad \text{où :}$$

- $I_v(P(t_1, \dots, t_n)) = v_{PRED_n}(P)(I_v(t_1), \dots, I_v(t_n))$; (prédicat)
- négation et connecteurs binaires : comme pour $LProp$;
- $I_v(\forall x.A) = 1$ ssi $I_{v[x:=d]}(A) = 1$ pour tout $d \in \mathcal{D}$; (quantif. univ.)
- $I_v(\exists x.A) = 1$ ssi $I_{v[x:=d]}(A) = 1$ pour quelque $d \in \mathcal{D}$; (quantif. exist.)
- avec $I_{v[x:=d]} = (\mathcal{D}, v_{VAR_{[x:=d]}}, \{v_{FON_n}\}_{n \in \mathbb{N}}, \{v_{PRED_n}\}_{n \in \mathbb{N}})$ avec

$$v_{VAR_{[x:=d]}}(y) = \begin{cases} d & \text{si } x = y \\ v_{VAR}(y) & \text{si } x \neq y \end{cases}$$



Interprétation des formules (exemple sur domaine fini)

- Soit $x \bullet y \equiv (x + y) \bmod 3$ (l'addition modulo 3) ;



Interprétation des formules (exemple sur domaine fini)

- Soit $x \bullet y \equiv (x + y) \bmod 3$ (l'addition modulo 3) ;
- soit $I_v = (N_3, v_{VAR}, \{v_{FON_0}, v_{FON_2}\}, \{v_{PRED_1}\})$ où
 - $N_3 = \{0, 1, 2\}$;
 - $v_{VAR} = [x \mapsto 1, y \mapsto 2]$;
 - $\{v_{FON_0} = [\mathsf{n}_0 \mapsto 0, \mathsf{n}_1 \mapsto 1, \mathsf{n}_2 \mapsto 2], v_{FON_2} = [\mathsf{f} \mapsto \bullet]\}$;
 - $\{v_{PRED_1} = [P \mapsto \text{EstPair}]\}$;



Interprétation des formules (exemple sur domaine fini)

- Soit $x \bullet y \equiv (x + y) \bmod 3$ (l'addition modulo 3) ;
- soit $I_v = (N_3, v_{VAR}, \{v_{FON_0}, v_{FON_2}\}, \{v_{PRED_1}\})$ où
 - $N_3 = \{0, 1, 2\}$;
 - $v_{VAR} = [x \mapsto 1, y \mapsto 2]$;
 - $\{v_{FON_0} = [\mathsf{n}_0 \mapsto 0, \mathsf{n}_1 \mapsto 1, \mathsf{n}_2 \mapsto 2], v_{FON_2} = [\mathsf{f} \mapsto \bullet]\}$;
 - $\{v_{PRED_1} = [P \mapsto \text{EstPair}]\}$;
- alors $I_v(P(x)) = ?$



Interprétation des formules (exemple sur domaine fini)

- Soit $x \oplus y \equiv (x + y) \bmod 3$ (l'addition modulo 3) ;
- soit $I_v = (N_3, v_{VAR}, \{v_{FON_0}, v_{FON_2}\}, \{v_{PRED_1}\})$ où
 - $N_3 = \{0, 1, 2\}$;
 - $v_{VAR} = [x \mapsto 1, y \mapsto 2]$;
 - $\{v_{FON_0} = [\mathsf{n}_0 \mapsto 0, \mathsf{n}_1 \mapsto 1, \mathsf{n}_2 \mapsto 2], v_{FON_2} = [\mathsf{f} \mapsto \oplus]\}$;
 - $\{v_{PRED_1} = [P \mapsto \text{EstPair}]\}$;
- alors $I_v(P(x)) = v_{PRED_1}(P)(v_{VAR}(x))$



Interprétation des formules (exemple sur domaine fini)

- Soit $x \oplus y \equiv (x + y) \bmod 3$ (l'addition modulo 3) ;
- soit $I_v = (N_3, v_{VAR}, \{v_{FON_0}, v_{FON_2}\}, \{v_{PRED_1}\})$ où
 - $N_3 = \{0, 1, 2\}$;
 - $v_{VAR} = [x \mapsto 1, y \mapsto 2]$;
 - $\{v_{FON_0} = [\mathsf{n}_0 \mapsto 0, \mathsf{n}_1 \mapsto 1, \mathsf{n}_2 \mapsto 2], v_{FON_2} = [\mathsf{f} \mapsto \oplus]\}$;
 - $\{v_{PRED_1} = [P \mapsto \text{EstPair}]\}$;
- alors $I_v(P(x)) = v_{PRED_1}(P)(v_{VAR}(x)) = \text{EstPair}(1)$



Interprétation des formules (exemple sur domaine fini)

- Soit $x \bullet 3 y \equiv (x + y) \bmod 3$ (l'addition modulo 3) ;
- soit $I_v = (N_3, v_{VAR}, \{v_{FON_0}, v_{FON_2}\}, \{v_{PRED_1}\})$ où
 - $N_3 = \{0, 1, 2\}$;
 - $v_{VAR} = [x \mapsto 1, y \mapsto 2]$;
 - $\{v_{FON_0} = [\mathsf{n}_0 \mapsto 0, \mathsf{n}_1 \mapsto 1, \mathsf{n}_2 \mapsto 2], v_{FON_2} = [\mathsf{f} \mapsto \bullet 3]\}$;
 - $\{v_{PRED_1} = [P \mapsto \text{EstPair}]\}$;
- alors $I_v(P(x)) = v_{PRED_1}(P)(v_{VAR}(x)) = \text{EstPair}(1) = 0$ (faux) ;



Interprétation des formules (exemple sur domaine fini avec \forall)

- Comment calculer $I_v(\forall x.P(f(y, x)))$?



Interprétation des formules (exemple sur domaine fini avec \forall)

- Comment calculer $I_v(\forall x.P(f(y, x)))$?

$I_v(\forall x.P(f(y, x))) = I_{v[x:=d]}(P(f(y, x)))$ pour tout $d \in N_3$



Interprétation des formules (exemple sur domaine fini avec \forall)

- Comment calculer $I_v(\forall x.P(f(y, x)))$?

$$I_v(\forall x.P(f(y, x))) = I_{v[x:=d]}(P(f(y, x))) \text{ pour tout } d \in N_3$$

$$= v_{PRED_1}(P)(v_{FON_2}(f)(v_{VAR_{[x:=d]}}(y), v_{VAR_{[x:=d]}}(x)))$$



Interprétation des formules (exemple sur domaine fini avec \forall)

- Comment calculer $I_v(\forall x.P(f(y, x)))$?

$$I_v(\forall x.P(f(y, x))) = I_{v[x:=d]}(P(f(y, x))) \text{ pour tout } d \in N_3$$

$$\begin{aligned} &= v_{PRED_1}(P)(v_{FON_2}(f)(v_{VAR_{[x:=d]}}(y), v_{VAR_{[x:=d]}}(x))) \\ &= EstPair(2 \bullet d) \end{aligned} \quad (\text{notation infixée})$$



Interprétation des formules (exemple sur domaine fini avec \forall)

- Comment calculer $I_v(\forall x.P(f(y, x)))$?

$$\begin{aligned}
 I_v(\forall x.P(f(y, x))) &= I_{v[x:=d]}(P(f(y, x))) \text{ pour tout } d \in N_3 \\
 &= v_{PRED_1}(P)(v_{FON_2}(f)(v_{VAR_{[x:=d]}}(y), v_{VAR_{[x:=d]}}(x))) \\
 &= EstPair(2 \bullet d) \quad (\text{notation infixée})
 \end{aligned}$$

- pour $d = 0$: $EstPair(2 \bullet 0) = EstPair(2) = 1$;



Interprétation des formules (exemple sur domaine fini avec \forall)

- Comment calculer $I_v(\forall x.P(f(y, x)))$?

$$\begin{aligned}
 I_v(\forall x.P(f(y, x))) &= I_{v[x:=d]}(P(f(y, x))) \text{ pour tout } d \in N_3 \\
 &= v_{PRED_1}(P)(v_{FON_2}(f)(v_{VAR_{[x:=d]}}(y), v_{VAR_{[x:=d]}}(x))) \\
 &= EstPair(2 \bullet d) \quad (\text{notation infixée})
 \end{aligned}$$

- pour $d = 0$: $EstPair(2 \bullet 0) = EstPair(2) = 1$;
- pour $d = 1$: $EstPair(2 \bullet 1) = EstPair(0) = 1$;



Interprétation des formules (exemple sur domaine fini avec \forall)

- Comment calculer $I_v(\forall x.P(f(y, x)))$?

$$\begin{aligned}
 I_v(\forall x.P(f(y, x))) &= I_{v[x:=d]}(P(f(y, x))) \text{ pour tout } d \in N_3 \\
 &= v_{PRED_1}(P)(v_{FON_2}(f)(v_{VAR_{[x:=d]}}(y), v_{VAR_{[x:=d]}}(x))) \\
 &= EstPair(2 \bullet d) \quad (\text{notation infixée})
 \end{aligned}$$

- pour $d = 0$: $EstPair(2 \bullet 0) = EstPair(2) = 1$;
- pour $d = 1$: $EstPair(2 \bullet 1) = EstPair(0) = 1$;
- pour $d = 2$: $EstPair(2 \bullet 2) = EstPair(1) = 0$;



Interprétation des formules (exemple sur domaine fini avec \forall)

- Comment calculer $I_v(\forall x.P(f(y, x)))$?

$$\begin{aligned}
 I_v(\forall x.P(f(y, x))) &= I_{v[x:=d]}(P(f(y, x))) \text{ pour tout } d \in N_3 \\
 &= v_{PRED_1}(P)(v_{FON_2}(f)(v_{VAR_{[x:=d]}}(y), v_{VAR_{[x:=d]}}(x))) \\
 &= EstPair(2 \bullet d) \quad (\text{notation infixée})
 \end{aligned}$$

- pour $d = 0$: $EstPair(2 \bullet 0) = EstPair(2) = 1$;
- pour $d = 1$: $EstPair(2 \bullet 1) = EstPair(0) = 1$;
- pour $d = 2$: $EstPair(2 \bullet 2) = EstPair(1) = 0$;

► conclusion : l'interprétation n'est pas vraie pour tout $x \in N_3$,
donc

$$I_v(\forall x.P(f(y, x))) = 0.$$