

# UE Modélisation et Architecture

Matière Modélisation

- ▶ Responsable UE et matière : Marc Pantel  
<http://pantel.perso.enseeiht.fr>

# UE Modélisation et Architecture

## Matière Modélisation

- ▶ Responsable UE et matière : Marc Pantel  
<http://pantel.perso.enseeiht.fr>
- ▶ Cours : 5 séances (Marc Pantel)
- ▶ Travaux Dirigés : 7 séances
- ▶ Travaux Pratiques en binôme : 7 séances

# UE Modélisation et Architecture

## Matière Modélisation

- ▶ Responsable UE et matière : Marc Pantel  
<http://pantel.perso.enseeiht.fr>
- ▶ Cours : 5 séances (Marc Pantel)
- ▶ Travaux Dirigés : 7 séances
- ▶ Travaux Pratiques en binôme : 7 séances
- ▶ Examen écrit 60% : Similaire TD, Formulaire fourni (sans autres documents)
- ▶ Bureau d'étude en temps limité 40% : Similaire TP

# UE Modélisation et Architecture

## Matière Modélisation

- ▶ Responsable UE et matière : Marc Pantel  
<http://pantel.perso.enseeiht.fr>
- ▶ Cours : 5 séances (Marc Pantel)
- ▶ Travaux Dirigés : 7 séances
- ▶ Travaux Pratiques en binôme : 7 séances
- ▶ Examen écrit 60% : Similaire TD, Formulaire fourni (sans autres documents)
- ▶ Bureau d'étude en temps limité 40% : Similaire TP
- ▶ **Attention** : Le vocabulaire et les notations dans ce domaine sont multiples et pas encore stabilisés. Cette matière est la synthèse de nombreux documents. Il n'y a pas d'ouvrage de référence associé.

# Objectifs

## Informatique et Mathématiques

- ▶ Les sciences physiques, de la vie, humaines et sociales s'appuient sur des modèles mathématiques pour la compréhension, la prédiction et la prescription (définition d'un nouveau produit).
  - ▶ Modèle en science (compréhension et prédiction) : Approximation de la réalité  
Le modèle doit être aussi proche que possible de la réalité.
  - ▶ Modèle en ingénierie : Prescription de la réalité  
La réalité doit être aussi proche que possible du modèle.

# Objectifs

## Informatique et Mathématiques

- ▶ Les sciences physiques, de la vie, humaines et sociales s'appuient sur des modèles mathématiques pour la compréhension, la prédiction et la prescription (définition d'un nouveau produit).
  - ▶ Modèle en science (compréhension et prédiction) : Approximation de la réalité  
Le modèle doit être aussi proche que possible de la réalité.
  - ▶ Modèle en ingénierie : Prescription de la réalité  
La réalité doit être aussi proche que possible du modèle.
- ▶ Qu'en est il de l'informatique ?
  - ▶ Science formelle similaire aux mathématiques
  - ▶ Thèse de Church-Turing (calculabilité) :  
Les programmes informatiques permettent de calculer les même fonctions que les mathématiques
  - ▶ Correspondance/Isomorphisme de Curry-Howard :  
Un programme  $P$  bien typé de type  $\tau$  est isomorphe à la preuve en logique constructive d'une formule isomorphe à  $\tau$

# Objectifs

## Informatique et Mathématiques

- ▶ Les sciences physiques, de la vie, humaines et sociales s'appuient sur des modèles mathématiques pour la compréhension, la prédiction et la prescription (définition d'un nouveau produit).
  - ▶ Modèle en science (compréhension et prédiction) : Approximation de la réalité  
Le modèle doit être aussi proche que possible de la réalité.
  - ▶ Modèle en ingénierie : Prescription de la réalité  
La réalité doit être aussi proche que possible du modèle.
- ▶ Qu'en est il de l'informatique ?
  - ▶ Science formelle similaire aux mathématiques
  - ▶ Thèse de Church-Turing (calculabilité) :  
Les programmes informatiques permettent de calculer les même fonctions que les mathématiques
  - ▶ Correspondance/Isomorphisme de Curry-Howard :  
Un programme  $P$  bien typé de type  $\tau$  est isomorphe à la preuve en logique constructive d'une formule isomorphe à  $\tau$
- ▶ Etude des techniques de modélisation de programmes et de langages
- ▶ Etude des techniques de preuve de programmes

# Plan

## Modélisation

- ▶ Modélisation et Preuve de programmes
  - ▶ Logique des propositions : C1, TD1, TD2, TP1
  - ▶ Logique des prédicats : C2, TD3, TP2
  - ▶ Preuve de programmes fonctionnels : C3, TD4, TP3
  - ▶ Preuve de programmes impératifs : C4, TD5, TP4
- ▶ Modélisation des langages : C5, TD6, TD7, TP5, TP6, TP7
  - ▶ Théorie des langages
  - ▶ Expressions régulières, Grammaires
  - ▶ XML, JSON



# Notation

## Règles de déduction

- Soient  $J_1, \dots, J_n$  et  $J$  des jugements :

	Notation	Signification
Déduction	$\frac{J_1 \quad J_n}{J}$	si $J_1$ et ... et $J_n$ sont valides alors $J$ est valide
Axiome	$\frac{}{J}$	$J$ est valide

# Notation

## Règles de déduction

- Soient  $J_1, \dots, J_n$  et  $J$  des jugements :

	Notation	Signification
Déduction	$\frac{J_1 \quad J_n}{J}$	si $J_1$ et ... et $J_n$ sont valides alors $J$ est valide
Axiome	$\frac{}{J}$	$J$ est valide

- sémantique :  $(\bigwedge_{i \in [1 \dots n]} J_i) \rightarrow J$  et  $\top \rightarrow J$

# Notation

## Règles de déduction

- Soient  $J_1, \dots, J_n$  et  $J$  des jugements :

	Notation	Signification
Déduction	$\frac{J_1 \quad J_n}{J}$	si $J_1$ et ... et $J_n$ sont valides alors $J$ est valide
Axiome	$\frac{}{J}$	$J$ est valide

- sémantique :  $(\bigwedge_{i \in [1 \dots n]} J_i) \rightarrow J$  et  $\top \rightarrow J$
- méthode de chaînage arrière :  
pour prouver  $J$ , il suffit de prouver  $J_1$  et ... et  $J_n$

# Notation

## Règles de déduction

- Soient  $J_1, \dots, J_n$  et  $J$  des jugements :

	Notation	Signification
Déduction	$\frac{J_1 \quad J_n}{J}$	si $J_1$ et ... et $J_n$ sont valides alors $J$ est valide
Axiome	$\frac{}{J}$	$J$ est valide

- sémantique :  $(\bigwedge_{i \in [1 \dots n]} J_i) \rightarrow J$  et  $\top \rightarrow J$
- méthode de chaînage arrière :  
pour prouver  $J$ , il suffit de prouver  $J_1$  et ... et  $J_n$
- Exemples de jugements :
  - Typage :  $x_1 : \tau_1, \dots, x_n : \tau_n \vdash e : \tau$
  - Calcul :  $x_1 \mapsto v_1, \dots, x_n \mapsto v_n \vdash e \Rightarrow v$
  - Preuve :  $H_1, \dots, H_n \vdash \varphi$

# Systèmes formels

## Définitions

- ▶ Syntaxe concrète : Vision utilisateur  
Logique : Formule avec constantes, variables, opérateurs, lieux (définition variables) **et parenthèses**
- ▶ Syntaxe abstraite : Vision information structurée  
Logique : Arbre étiqueté par constantes, variables, opérateurs, lieux **sans parenthèses**
- ▶ Sémantique : Signification  
Logique : Valide, Satisfiable, Invalide, Insatisfiable, Inconnue  
Notation  $\models \varphi$
- ▶ Axiomatisation de la sémantique  
Modélise la sémantique par la construction de preuves (démonstration)  
Approche syntaxique de la sémantique  
Logique : Notation  $\vdash \varphi$
- ▶ Mécanisation de l'axiomatisation  
Construction automatique des preuves

# Systèmes formels

## Propriétés souhaitées

- ▶ Consistance sémantique : La sémantique ne peut pas être
  - ▶ Valide **et** Invalide
  - ▶ Satisfiable **et** Insatisfiable
- ▶ Complétude sémantique : La sémantique est toujours
  - ▶ Valide **ou** Invalide
  - ▶ Satisfiable **ou** Insatisfiable
  - ▶ **Jamais** inconnue

# Systèmes formels

## Propriétés souhaitées

- ▶ Consistance sémantique : La sémantique ne peut pas être
  - ▶ Valide **et** Invalide
  - ▶ Satisfiable **et** Insatisfiable
- ▶ Complétude sémantique : La sémantique est toujours
  - ▶ Valide **ou** Invalide
  - ▶ Satisfiable **ou** Insatisfiable
  - ▶ **Jamais** inconnue
- ▶ Correction axiomatisation :  $\forall \varphi. \vdash \varphi \rightarrow \models \varphi$
- ▶ Complétude axiomatisation :  $\forall \varphi. \models \varphi \rightarrow \vdash \varphi$   
Exemple : Incomplétude de l'arithmétique (théorème de Gödel)

# Systèmes formels

## Propriétés souhaitées

- ▶ Consistance sémantique : La sémantique ne peut pas être
  - ▶ Valide **et** Invalide
  - ▶ Satisfiable **et** Insatisfiable
- ▶ Complétude sémantique : La sémantique est toujours
  - ▶ Valide **ou** Invalide
  - ▶ Satisfiable **ou** Insatisfiable
  - ▶ **Jamais** inconnue
- ▶ Correction axiomatisation :  $\forall \varphi. \vdash \varphi \rightarrow \models \varphi$
- ▶ Complétude axiomatisation :  $\forall \varphi. \models \varphi \rightarrow \vdash \varphi$   
Exemple : Incomplétude de l'arithmétique (théorème de Gödel)
- ▶ Décidabilité : Mécanisation construit une preuve en temps fini
- ▶ Semi-décidabilité : Mécanisation calcule en temps fini quand la preuve existe (valide, satisfiable)
- ▶ Indécidabilité : Mécanisation peut ne pas se terminer  
Exemple : Test d'arrêt de la machine de Turing



# Syntaxe

## Vision algébrique

- ▶ Notons  $\Phi$  l'ensemble dénombrable des formules bien formées de logique des propositions
- ▶ Éléments lexicaux :
  - ▶ Propositions (variables propositionnelles) : mots, phrases, ... (ensemble  $\mathcal{P}$  dénombrables)
  - ▶ Opérateurs :  $\perp, \top, \neg, \vee, \wedge, \rightarrow, \leftrightarrow$
  - ▶ Contrôle structure (associativité, priorité) :  $(, )$

# Syntaxe

## Vision algébrique

- ▶ Notons  $\Phi$  l'ensemble dénombrable des formules bien formées de logique des propositions
- ▶ Eléments lexicaux :
  - ▶ Propositions (variables propositionnelles) : mots, phrases, ... (ensemble  $\mathcal{P}$  dénombrables)
  - ▶ Opérateurs :  $\perp, \top, \neg, \vee, \wedge, \rightarrow, \leftrightarrow$
  - ▶ Contrôle structure (associativité, priorité) :  $(, )$
- ▶ Eléments grammaticaux :
  - ▶ Constantes (Opérateurs zéro-aire) : Propositions,  $\top$  (Té) et  $\perp$  (Anti-Té)
  - ▶ Opérateur unaire :  $\neg$  (Négation)
  - ▶ Opérateurs binaires associatifs et commutatifs :  $\vee$  (disjonction),  $\wedge$  (conjonction),  $\leftrightarrow$  (équivalence)
  - ▶ Opérateur binaire associatif à droite :  $\rightarrow$  (implication)
  - ▶ Priorité croissante :  $\rightarrow, \leftrightarrow, \vee, \wedge, \neg$

Soit  $\mathcal{P}$  un ensemble dénombrable de variables propositionnelles

► Version classique

Axiomes  $\frac{}{\top \in \Phi} \quad \frac{}{\perp \in \Phi} \quad \frac{}{P \in \Phi} \quad (P \in \mathcal{P})$

Déductions  $\frac{\varphi \in \Phi}{(\varphi) \in \Phi} \quad \frac{\varphi \in \Phi}{\neg \varphi \in \Phi}$   
 $\frac{\varphi \in \Phi \quad \psi \in \Phi}{\varphi \text{ op } \psi \in \Phi} \quad (op \in \{\wedge, \vee, \rightarrow, \leftrightarrow\})$

Soit  $\mathcal{P}$  un ensemble dénombrable de variables propositionnelles

► Version classique

Axiomes  $\frac{}{\top \in \Phi} \quad \frac{}{\perp \in \Phi} \quad \frac{}{P \in \Phi} \quad (P \in \mathcal{P})$

Déductions  $\frac{\varphi \in \Phi}{(\varphi) \in \Phi} \quad \frac{\varphi \in \Phi}{\neg \varphi \in \Phi}$   
 $\frac{\varphi \in \Phi \quad \psi \in \Phi}{\varphi \text{ op } \psi \in \Phi} \quad (op \in \{\wedge, \vee, \rightarrow, \leftrightarrow\})$

► Il existe une définition stratifiée plus complexe pour éliminer les paradoxes de Russell (théorie des types, New Foundations, ...)

# Exploitation pour la modélisation

## Logique de propositions

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en propositions combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : parties de l'énoncé liées par les conjonctions (phrases, groupe nominal, groupe sujet et verbal, ...)

# Exploitation pour la modélisation

## Logique de propositions

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en propositions combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : parties de l'énoncé liées par les conjonctions (phrases, groupe nominal, groupe sujet et verbal, ...)
- ▶ Exemples : Lorsque je dors, je fais des rêves ou des cauchemars. Je ne suis reposé que lorsque j'ai fait des rêves. Or je suis reposé donc je n'ai pas fait de cauchemars.
- ▶ Propositions :  $D$  = je dors;  $V$  = je fais des rêves;  $C$  = je fais des cauchemars;  $P$  = je suis reposé.
- ▶ Formule :

# Exploitation pour la modélisation

## Logique de propositions

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en propositions combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : parties de l'énoncé liées par les conjonctions (phrases, groupe nominal, groupe sujet et verbal, ...)
- ▶ Exemples : **Lorsque je dors, je fais des rêves ou des cauchemars.** Je ne suis reposé que lorsque j'ai fait des rêves. Or je suis reposé donc je n'ai pas fait de cauchemars.
- ▶ Propositions :  $D$  = je dors;  $V$  = je fais des rêves;  $C$  = je fais des cauchemars;  $P$  = je suis reposé.
- ▶ Formule :  $(D \rightarrow V \vee C)$

# Exploitation pour la modélisation

## Logique de propositions

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en propositions combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : parties de l'énoncé liées par les conjonctions (phrases, groupe nominal, groupe sujet et verbal, ...)
- ▶ Exemples : Lorsque je dors, je fais des rêves ou des cauchemars. **Je ne suis reposé que lorsque j'ai fait des rêves.** Or je suis reposé donc je n'ai pas fait de cauchemars.
- ▶ Propositions :  $D$  = je dors;  $V$  = je fais des rêves;  $C$  = je fais des cauchemars;  $P$  = je suis reposé.
- ▶ Formule :  $(D \rightarrow V \vee C) \wedge (P \rightarrow V)$



# Exploitation pour la modélisation

## Logique de propositions

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en propositions combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : parties de l'énoncé liées par les conjonctions (phrases, groupe nominal, groupe sujet et verbal, ...)
- ▶ Exemples : Lorsque je dors, je fais des rêves ou des cauchemars. Je ne suis reposé que lorsque j'ai fait des rêves. Or je suis reposé donc je n'ai pas fait de cauchemars.
- ▶ Propositions :  $D$  = je dors;  $V$  = je fais des rêves;  $C$  = je fais des cauchemars;  $P$  = je suis reposé.
- ▶ Formule :  $(D \rightarrow V \vee C) \wedge (P \rightarrow V) \wedge P$

# Exploitation pour la modélisation

## Logique de propositions

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en propositions combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : parties de l'énoncé liées par les conjonctions (phrases, groupe nominal, groupe sujet et verbal, ...)
- ▶ Exemples : Lorsque je dors, je fais des rêves ou des cauchemars. Je ne suis reposé que lorsque j'ai fait des rêves. Or je suis reposé **donc je n'ai pas fait de cauchemars.**
- ▶ Propositions :  $D$  = je dors;  $V$  = je fais des rêves;  $C$  = je fais des cauchemars;  $P$  = je suis reposé.
- ▶ Formule :  $(D \rightarrow V \vee C) \wedge (P \rightarrow V) \wedge P \rightarrow \neg C$

# Sémantique

## Tables de vérité

- Valeurs de vérité notées  $V$  (vrai) et  $F$  (faux)  
Autres notations possibles ( $T$  et  $F$ , 1 et 0, ...)

# Sémantique

## Tables de vérité

- Valeurs de vérité notées  $V$  (vrai) et  $F$  (faux)  
Autres notations possibles ( $T$  et  $F$ , 1 et 0, ...)
- Opérateurs définis pour chaque valeur de vérité des opérandes

$\neg$	
$F$	$V$
$V$	$F$

$\wedge$	$F$	$V$
$F$	$F$	$F$
$V$	$F$	$V$

$\vee$	$F$	$V$
$F$	$F$	$V$
$V$	$V$	$V$

$\rightarrow$	$F$	$V$
$F$	$V$	$V$
$V$	$F$	$V$

$\leftrightarrow$	$F$	$V$
$F$	$V$	$F$
$V$	$F$	$V$

# Sémantique

## Tables de vérité

- Valeurs de vérité notées  $V$  (vrai) et  $F$  (faux)  
Autres notations possibles ( $T$  et  $F$ , 1 et 0, ...)
- Opérateurs définis pour chaque valeur de vérité des opérandes

$\neg$	
$F$	$V$
$V$	$F$

$\wedge$	$F$	$V$
$F$	$F$	$F$
$V$	$F$	$V$

$\vee$	$F$	$V$
$F$	$F$	$V$
$V$	$V$	$V$

$\rightarrow$	$F$	$V$
$F$	$V$	$V$
$V$	$F$	$V$

$\leftrightarrow$	$F$	$V$
$F$	$V$	$F$
$V$	$F$	$V$

- Notation sous la forme de formules élémentaires :

$A$	$B$	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
$F$	$F$	$V$	$F$	$F$	$V$	$V$
$F$	$V$	$V$	$F$	$V$	$V$	$F$
$V$	$F$	$F$	$F$	$V$	$F$	$F$
$V$	$V$	$F$	$V$	$V$	$V$	$V$

# Sémantique

## Construction tables de vérité

- ▶ Formule  $\varphi \in \Phi$  contient variables  $\{P_i\}_{i \in [1 \dots n]} \subseteq \mathcal{P}$
- ▶ Variables propositionnelles  $P_i$  reçoivent valeurs de vérité

# Sémantique

## Construction tables de vérité

- ▶ Formule  $\varphi \in \Phi$  contient variables  $\{P_i\}_{i \in [1 \dots n]} \subseteq \mathcal{P}$
- ▶ Variables propositionnelles  $P_i$  reçoivent valeurs de vérité
- ▶  $n$  variables :  $2^n$  lignes
- ▶ Discriminant ligne : Formule uniquement satisfaite par la ligne
- ▶ Discriminant ligne :  $\bigwedge_{i \in [1 \dots n]} \alpha_i$  avec  $\begin{cases} \alpha_i = P_i & \text{si valeur } V \\ \alpha_i = \neg P_i & \text{si valeur } F \end{cases}$

# Sémantique

## Construction tables de vérité

- ▶ Formule  $\varphi \in \Phi$  contient variables  $\{P_i\}_{i \in [1 \dots n]} \subseteq \mathcal{P}$
- ▶ Variables propositionnelles  $P_i$  reçoivent valeurs de vérité
- ▶  $n$  variables :  $2^n$  lignes
- ▶ Discriminant ligne : Formule uniquement satisfaite par la ligne
- ▶ Discriminant ligne :  $\bigwedge_{i \in [1 \dots n]} \alpha_i$  avec  $\begin{cases} \alpha_i = P_i & \text{si valeur } V \\ \alpha_i = \neg P_i & \text{si valeur } F \end{cases}$
- ▶ 1 colonne par variable propositionnelle
- ▶ 1 colonne par opérateur de la formule
- ▶ dont 1 colonne pour la formule complète



# Sémantique

## Exemple de tables de vérité

► Formule :  $(A \wedge B) \rightarrow (B \vee A)$

► Table de vérité

	<i>A</i>	<i>B</i>			
	<i>F</i>	<i>F</i>			
	<i>F</i>	<i>V</i>			
	<i>V</i>	<i>F</i>			
	<i>V</i>	<i>V</i>			

# Sémantique

## Exemple de tables de vérité

► Formule :  $(A \wedge B) \rightarrow (B \vee A)$

► Table de vérité

Discriminant	$A$	$B$			
$\neg A \wedge \neg B$	$F$	$F$			
$\neg A \wedge B$	$F$	$V$			
$A \wedge \neg B$	$V$	$F$			
$A \wedge B$	$V$	$V$			

# Sémantique

## Exemple de tables de vérité

► Formule :  $(A \wedge B) \rightarrow (B \vee A)$

► Table de vérité

Discriminant	$A$	$B$	$A \wedge B$		
$\neg A \wedge \neg B$	$F$	$F$	$F$		
$\neg A \wedge B$	$F$	$V$	$F$		
$A \wedge \neg B$	$V$	$F$	$F$		
$A \wedge B$	$V$	$V$	$V$		

# Sémantique

## Exemple de tables de vérité

► Formule :  $(A \wedge B) \rightarrow (B \vee A)$

► Table de vérité

Discriminant	$A$	$B$	$A \wedge B$	$B \vee A$	
$\neg A \wedge \neg B$	$F$	$F$	$F$	$F$	
$\neg A \wedge B$	$F$	$V$	$F$	$V$	
$A \wedge \neg B$	$V$	$F$	$F$	$V$	
$A \wedge B$	$V$	$V$	$V$	$V$	

# Sémantique

## Exemple de tables de vérité

► Formule :  $(A \wedge B) \rightarrow (B \vee A)$

► Table de vérité

Discriminant	$A$	$B$	$A \wedge B$	$B \vee A$	$(A \wedge B) \rightarrow (B \vee A)$
$\neg A \wedge \neg B$	$F$	$F$	$F$	$F$	$V$
$\neg A \wedge B$	$F$	$V$	$F$	$V$	$V$
$A \wedge \neg B$	$V$	$F$	$F$	$V$	$V$
$A \wedge B$	$V$	$V$	$V$	$V$	$V$

# Sémantique

## Exemple de tables de vérité

► Formule :  $(A \wedge B) \rightarrow (B \vee A)$

► Table de vérité

Discriminant	$A$	$B$	$A \wedge B$	$B \vee A$	$(A \wedge B) \rightarrow (B \vee A)$
$\neg A \wedge \neg B$	$F$	$F$	$F$	$F$	$V$
$\neg A \wedge B$	$F$	$V$	$F$	$V$	$V$
$A \wedge \neg B$	$V$	$F$	$F$	$V$	$V$
$A \wedge B$	$V$	$V$	$V$	$V$	$V$

► Formule satisfiable et valide : Théorème, Tautologie

# Sémantique

## Vocabulaire

Selon sa table de vérité,  $\varphi \in \Phi$  est :

- ▶ Valide, tautologie, ... :

Toutes les lignes V

Notée  $\models \varphi$

# Sémantique

## Vocabulaire

Selon sa table de vérité,  $\varphi \in \Phi$  est :

- ▶ Valide, tautologie, ... :  
Toutes les lignes  $V$   
Notée  $\models \varphi$
- ▶ Satisfiable, consistante, cohérente, ... :  
Au moins une ligne  $V$  (modèle de  $\varphi$ )  
Si  $L$  est son discriminant alors  $\models L \rightarrow \varphi$   
Notée  $\neg \models \neg \varphi$   
Si Valide alors Satisfiable



# Sémantique

## Vocabulaire

Selon sa table de vérité,  $\varphi \in \Phi$  est :

- ▶ Valide, tautologie, ... :  
Toutes les lignes  $V$   
Notée  $\models \varphi$
- ▶ Satisfiable, consistante, cohérente, ... :  
Au moins une ligne  $V$  (modèle de  $\varphi$ )  
Si  $L$  est son discriminant alors  $\models L \rightarrow \varphi$   
Notée  $\neg \models \neg \varphi$   
Si Valide alors Satisfiable
- ▶ Invalide, ... :  
Au moins une ligne  $F$   
Si et seulement si  $\neg \varphi$  satisfiable  
Notée  $\neg \models \varphi$

# Sémantique

## Vocabulaire

Selon sa table de vérité,  $\varphi \in \Phi$  est :

- ▶ Valide, tautologie, ... :  
Toutes les lignes  $V$   
Notée  $\models \varphi$
- ▶ Satisfiable, consistante, cohérente, ... :  
Au moins une ligne  $V$  (modèle de  $\varphi$ )  
Si  $L$  est son discriminant alors  $\models L \rightarrow \varphi$   
Notée  $\neg \models \neg \varphi$   
Si Valide alors Satisfiable
- ▶ Invalide, ... :  
Au moins une ligne  $F$   
Si et seulement si  $\neg \varphi$  satisfiable  
Notée  $\neg \models \varphi$
- ▶ Insatisfiable, inconsistante, incohérente, antilogie, ... :  
Toutes les lignes  $F$   
Si et seulement si  $\neg \varphi$  valide  
Notée  $\models \neg \varphi$   
Si Insatisfiable alors Invalide

# Sémantique

## Relation d'équivalence

- ▶ Soient  $\varphi, \psi, \chi \in \Phi$  :
- ▶  $\varphi = \psi$  si et seulement si  $\varphi$  et  $\psi$  ont la même table de vérité
- ▶  $\varphi = \psi$  si et seulement si  $\models \varphi \leftrightarrow \psi$

# Sémantique

## Relation d'équivalence

- ▶ Soient  $\varphi, \psi, \chi \in \Phi$  :
- ▶  $\varphi = \psi$  si et seulement si  $\varphi$  et  $\psi$  ont la même table de vérité
- ▶  $\varphi = \psi$  si et seulement si  $\models \varphi \leftrightarrow \psi$
- ▶ Equivalence de  $\rightarrow$  et  $\leftrightarrow$  :

$$\varphi \rightarrow \psi = \neg\varphi \vee \psi$$

$$\varphi \leftrightarrow \psi = (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$

$$\varphi \leftrightarrow \psi = (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$$

# Sémantique

## Relation d'équivalence

- Soient  $\varphi, \psi, \chi \in \Phi$  :
- $\varphi = \psi$  si et seulement si  $\varphi$  et  $\psi$  ont la même table de vérité
- $\varphi = \psi$  si et seulement si  $\models \varphi \leftrightarrow \psi$
- Equivalence de  $\rightarrow$  et  $\leftrightarrow$  :

$$\varphi \rightarrow \psi = \neg\varphi \vee \psi$$

$$\varphi \leftrightarrow \psi = (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$

$$\varphi \leftrightarrow \psi = (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$$

- Lois de De Morgan :

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

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

# Sémantique

## Relation d'équivalence

- ▶ Soient  $\varphi, \psi, \chi \in \Phi$  :
- ▶  $\varphi = \psi$  si et seulement si  $\varphi$  et  $\psi$  ont la même table de vérité
- ▶  $\varphi = \psi$  si et seulement si  $\models \varphi \leftrightarrow \psi$
- ▶ Equivalence de  $\rightarrow$  et  $\leftrightarrow$  :

$$\varphi \rightarrow \psi = \neg\varphi \vee \psi$$

$$\varphi \leftrightarrow \psi = (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$

$$\varphi \leftrightarrow \psi = (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$$

- ▶ Lois de De Morgan :

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

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

- ▶ Opposé, éléments neutres et absorbants :

$$\varphi \wedge \neg\varphi = \perp$$

$$\varphi \rightarrow \varphi = \top$$

$$\varphi \wedge \perp = \perp$$

$$\varphi \vee \perp = \varphi$$

$$\neg\neg\varphi = \varphi$$

$$\varphi \vee \neg\varphi = \top$$

$$\varphi \leftrightarrow \varphi = \top$$

$$\varphi \wedge \top = \varphi$$

$$\varphi \vee \top = \top$$

# Sémantique

## Relation d'équivalence

- ▶ Soient  $\varphi, \psi, \chi \in \Phi$  :
- ▶ Idempotence :

$$\varphi \wedge \varphi = \varphi \quad \varphi \vee \varphi = \varphi$$

# Sémantique

## Relation d'équivalence

► Soient  $\varphi, \psi, \chi \in \Phi$  :

► Idempotence :

$$\varphi \wedge \varphi = \varphi \quad \varphi \vee \varphi = \varphi$$

► Commutativité :

$$\varphi \wedge \psi = \psi \wedge \varphi \quad \varphi \vee \psi = \psi \vee \varphi \quad \varphi \leftrightarrow \psi = \psi \leftrightarrow \varphi$$



# Sémantique

## Relation d'équivalence

► Soient  $\varphi, \psi, \chi \in \Phi$  :

► Idempotence :

$$\varphi \wedge \varphi = \varphi \quad \varphi \vee \varphi = \varphi$$

► Commutativité :

$$\varphi \wedge \psi = \psi \wedge \varphi \quad \varphi \vee \psi = \psi \vee \varphi \quad \varphi \leftrightarrow \psi = \psi \leftrightarrow \varphi$$

► Associativité :

$$\begin{aligned} (\varphi \wedge \psi) \wedge \chi &= \varphi \wedge \psi \wedge \chi = \varphi \wedge (\psi \wedge \chi) \\ (\varphi \vee \psi) \vee \chi &= \varphi \vee \psi \vee \chi = \varphi \vee (\psi \vee \chi) \\ (\varphi \rightarrow \psi) \rightarrow \chi &\neq \varphi \rightarrow \psi \rightarrow \chi = \varphi \rightarrow (\psi \rightarrow \chi) \end{aligned}$$

# Sémantique

## Relation d'équivalence

► Soient  $\varphi, \psi, \chi \in \Phi$  :

► Idempotence :

$$\varphi \wedge \varphi = \varphi \quad \varphi \vee \varphi = \varphi$$

► Commutativité :

$$\varphi \wedge \psi = \psi \wedge \varphi \quad \varphi \vee \psi = \psi \vee \varphi \quad \varphi \leftrightarrow \psi = \psi \leftrightarrow \varphi$$

► Associativité :

$$\begin{aligned} (\varphi \wedge \psi) \wedge \chi &= \varphi \wedge \psi \wedge \chi = \varphi \wedge (\psi \wedge \chi) \\ (\varphi \vee \psi) \vee \chi &= \varphi \vee \psi \vee \chi = \varphi \vee (\psi \vee \chi) \\ (\varphi \rightarrow \psi) \rightarrow \chi &\neq \varphi \rightarrow \psi \rightarrow \chi = \varphi \rightarrow (\psi \rightarrow \chi) \end{aligned}$$

► Distributivité :

$$\begin{aligned} \varphi \wedge (\psi \vee \chi) &= (\varphi \wedge \psi) \vee (\varphi \wedge \chi) \\ \varphi \vee (\psi \wedge \chi) &= (\varphi \vee \psi) \wedge (\varphi \vee \chi) \end{aligned}$$

# Sémantique

## Relation d'équivalence

► Soient  $\varphi, \psi, \chi \in \Phi$  :

► Idempotence :

$$\varphi \wedge \varphi = \varphi \quad \varphi \vee \varphi = \varphi$$

► Commutativité :

$$\varphi \wedge \psi = \psi \wedge \varphi \quad \varphi \vee \psi = \psi \vee \varphi \quad \varphi \leftrightarrow \psi = \psi \leftrightarrow \varphi$$

► Associativité :

$$\begin{aligned} (\varphi \wedge \psi) \wedge \chi &= \varphi \wedge \psi \wedge \chi = \varphi \wedge (\psi \wedge \chi) \\ (\varphi \vee \psi) \vee \chi &= \varphi \vee \psi \vee \chi = \varphi \vee (\psi \vee \chi) \\ (\varphi \rightarrow \psi) \rightarrow \chi &\neq \varphi \rightarrow \psi \rightarrow \chi = \varphi \rightarrow (\psi \rightarrow \chi) \end{aligned}$$

► Distributivité :

$$\begin{aligned} \varphi \wedge (\psi \vee \chi) &= (\varphi \wedge \psi) \vee (\varphi \wedge \chi) \\ \varphi \vee (\psi \wedge \chi) &= (\varphi \vee \psi) \wedge (\varphi \vee \chi) \end{aligned}$$

► Simplification :

$$\begin{aligned} \varphi \vee (\neg \varphi \wedge \psi) &= \varphi \vee \psi & \varphi \vee (\varphi \wedge \psi) &= \varphi \\ \varphi \wedge (\neg \varphi \vee \psi) &= \varphi \wedge \psi & \varphi \wedge (\varphi \vee \psi) &= \varphi \end{aligned}$$

# Sémantique

## Formes normales

Pour toute formule  $\varphi \in \Phi$ , il existe :

- Une formule équivalente en forme normale disjonctive :

$$\varphi = \bigvee_{i \in [1 \dots n]} \beta_i$$

$$\beta_i = \bigwedge_{j \in [1 \dots m_i]} \alpha_{i,j}$$

$$\alpha_{i,j} \in \mathcal{P} \cup \{\neg P \mid P \in \mathcal{P}\}$$

# Sémantique

## Formes normales

Pour toute formule  $\varphi \in \Phi$ , il existe :

- Une formule équivalente en forme normale disjonctive :

$$\begin{aligned}\varphi &= \bigvee_{i \in [1 \dots n]} \beta_i \\ \beta_i &= \bigwedge_{j \in [1 \dots m_i]} \alpha_{i,j} \\ \alpha_{i,j} &\in \mathcal{P} \cup \{\neg P \mid P \in \mathcal{P}\}\end{aligned}$$

- Une formule équivalente en forme normale conjonctive :

$$\begin{aligned}\varphi &= \bigwedge_{i \in [1 \dots n]} \beta_i \\ \beta_i &= \bigvee_{j \in [1 \dots m_i]} \alpha_{i,j} \\ \alpha_{i,j} &\in \mathcal{P} \cup \{\neg P \mid P \in \mathcal{P}\}\end{aligned}$$

# Sémantique

## Formes normales

Pour toute formule  $\varphi \in \Phi$ , il existe :

- ▶ Une formule équivalente en forme normale disjonctive :

$$\begin{aligned}\varphi &= \bigvee_{i \in [1 \dots n]} \beta_i \\ \beta_i &= \bigwedge_{j \in [1 \dots m_i]} \alpha_{i,j} \\ \alpha_{i,j} &\in \mathcal{P} \cup \{\neg P \mid P \in \mathcal{P}\}\end{aligned}$$

- ▶ Une formule équivalente en forme normale conjonctive :

$$\begin{aligned}\varphi &= \bigwedge_{i \in [1 \dots n]} \beta_i \\ \beta_i &= \bigvee_{j \in [1 \dots m_i]} \alpha_{i,j} \\ \alpha_{i,j} &\in \mathcal{P} \cup \{\neg P \mid P \in \mathcal{P}\}\end{aligned}$$

- ▶ Ces formules sont obtenues en :
  - ▶ Remplaçant  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents
  - ▶ Rapprochant les négations  $\neg$  des variables propositionnelles
  - ▶ Effectuant les distributivités de  $\wedge$  sur  $\vee$  (respectivement de  $\vee$  sur  $\wedge$ )

# Sémantique

## Exemple d'équivalence sémantique

- ▶ Formule :  $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
- ▶ Raisonnement équationnel
  - ▶ Remplacer  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents  
 $((\neg A \vee B) \wedge (\neg \neg B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(\neg \neg B \vee \neg A))$

# Sémantique

## Exemple d'équivalence sémantique

- ▶ Formule :  $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
- ▶ Raisonnement équationnel
  - ▶ Remplacer  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents  
 $((\neg A \vee B) \wedge (\neg \neg B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(\neg \neg B \vee \neg A))$
  - ▶ Rapprocher les négations  $\neg$  des variables propositionnelles  
 $((\neg A \vee B) \wedge (B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(B \vee \neg A))$



# Sémantique

## Exemple d'équivalence sémantique

- ▶ Formule :  $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
- ▶ Raisonnement équationnel
  - ▶ Remplacer  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents
$$((\neg A \vee B) \wedge (\neg \neg B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(\neg \neg B \vee \neg A))$$
  - ▶ Rapprocher les négations  $\neg$  des variables propositionnelles
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(B \vee \neg A))$$
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((\neg \neg A \wedge \neg B) \wedge (\neg B \wedge \neg \neg A))$$

# Sémantique

## Exemple d'équivalence sémantique

- ▶ Formule :  $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
- ▶ Raisonnement équationnel
  - ▶ Remplacer  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents
$$((\neg A \vee B) \wedge (\neg \neg B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(\neg \neg B \vee \neg A))$$
  - ▶ Rapprocher les négations  $\neg$  des variables propositionnelles
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(B \vee \neg A))$$
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((\neg \neg A \wedge \neg B) \wedge (\neg B \wedge \neg \neg A))$$
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((A \wedge \neg B) \wedge (\neg B \wedge A))$$

# Sémantique

## Exemple d'équivalence sémantique

- ▶ Formule :  $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
- ▶ Raisonnement équationnel
  - ▶ Remplacer  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents
$$((\neg A \vee B) \wedge (\neg \neg B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(\neg \neg B \vee \neg A))$$
  - ▶ Rapprocher les négations  $\neg$  des variables propositionnelles
$$\begin{aligned} & ((\neg A \vee B) \wedge (B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(B \vee \neg A)) \\ & ((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((\neg \neg A \wedge \neg B) \wedge (\neg B \wedge \neg \neg A)) \\ & ((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((A \wedge \neg B) \wedge (\neg B \wedge A)) \end{aligned}$$
  - ▶ Simplification par Idempotence et Commutativité
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee (A \wedge \neg B)$$

# Sémantique

## Exemple d'équivalence sémantique

- ▶ Formule :  $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
- ▶ Raisonnement équationnel
  - ▶ Remplacer  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents
$$((\neg A \vee B) \wedge (\neg \neg B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(\neg \neg B \vee \neg A))$$
  - ▶ Rapprocher les négations  $\neg$  des variables propositionnelles
$$\begin{aligned} & ((\neg A \vee B) \wedge (B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(B \vee \neg A)) \\ & ((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((\neg \neg A \wedge \neg B) \wedge (\neg B \wedge \neg \neg A)) \\ & ((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((A \wedge \neg B) \wedge (\neg B \wedge A)) \end{aligned}$$
  - ▶ Simplification par Idempotence et Commutativité
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee (A \wedge \neg B)$$
  - ▶ Distributivité
$$((\neg A \wedge (B \vee \neg A)) \vee (B \wedge (B \vee \neg A))) \vee (A \wedge \neg B)$$

# Sémantique

## Exemple d'équivalence sémantique

- ▶ Formule :  $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
- ▶ Raisonnement équationnel
  - ▶ Remplacer  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents
$$((\neg A \vee B) \wedge (\neg \neg B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(\neg \neg B \vee \neg A))$$
  - ▶ Rapprocher les négations  $\neg$  des variables propositionnelles
$$\begin{aligned} & ((\neg A \vee B) \wedge (B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(B \vee \neg A)) \\ & ((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((\neg \neg A \wedge \neg B) \wedge (\neg B \wedge \neg \neg A)) \\ & ((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((A \wedge \neg B) \wedge (\neg B \wedge A)) \end{aligned}$$
  - ▶ Simplification par Idempotence et Commutativité
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee (A \wedge \neg B)$$
  - ▶ Distributivité
$$\begin{aligned} & ((\neg A \wedge (B \vee \neg A)) \vee (B \wedge (B \vee \neg A))) \vee (A \wedge \neg B) \\ & (((\neg A \wedge B) \vee (\neg A \wedge \neg A)) \vee ((B \wedge B) \vee (B \wedge \neg A))) \vee (A \wedge \neg B) \end{aligned}$$

# Sémantique

## Exemple d'équivalence sémantique

- ▶ Formule :  $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
- ▶ Raisonnement équationnel
  - ▶ Remplacer  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents
$$((\neg A \vee B) \wedge (\neg \neg B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(\neg \neg B \vee \neg A))$$
  - ▶ Rapprocher les négations  $\neg$  des variables propositionnelles
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(B \vee \neg A))$$
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((\neg \neg A \wedge \neg B) \wedge (\neg B \wedge \neg \neg A))$$
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((A \wedge \neg B) \wedge (\neg B \wedge A))$$
  - ▶ Simplification par Idempotence et Commutativité
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee (A \wedge \neg B)$$
  - ▶ Distributivité
$$((\neg A \wedge (B \vee \neg A)) \vee (B \wedge (B \vee \neg A))) \vee (A \wedge \neg B)$$
$$(((\neg A \wedge B) \vee (\neg A \wedge \neg A)) \vee ((B \wedge B) \vee (B \wedge \neg A))) \vee (A \wedge \neg B)$$
  - ▶ Simplification par Associativité, Idempotence et Commutativité
$$((\neg A \wedge B) \vee \neg A) \vee (B \vee (A \wedge \neg B))$$

# Sémantique

## Exemple d'équivalence sémantique

- ▶ Formule :  $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
- ▶ Raisonnement équationnel
  - ▶ Remplacer  $\rightarrow$  et  $\leftrightarrow$  par leurs équivalents
$$((\neg A \vee B) \wedge (\neg \neg B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(\neg \neg B \vee \neg A))$$
  - ▶ Rapprocher les négations  $\neg$  des variables propositionnelles
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee (\neg(\neg A \vee B) \wedge \neg(B \vee \neg A))$$
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((\neg \neg A \wedge \neg B) \wedge (\neg B \wedge \neg \neg A))$$
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee ((A \wedge \neg B) \wedge (\neg B \wedge A))$$
  - ▶ Simplification par Idempotence et Commutativité
$$((\neg A \vee B) \wedge (B \vee \neg A)) \vee (A \wedge \neg B)$$
  - ▶ Distributivité
$$((\neg A \wedge (B \vee \neg A)) \vee (B \wedge (B \vee \neg A))) \vee (A \wedge \neg B)$$
$$(((\neg A \wedge B) \vee (\neg A \wedge \neg A)) \vee ((B \wedge B) \vee (B \wedge \neg A))) \vee (A \wedge \neg B)$$
  - ▶ Simplification par Associativité, Idempotence et Commutativité
$$((\neg A \wedge B) \vee \neg A) \vee (B \vee (A \wedge \neg B))$$
  - ▶ Simplification
$$\neg A \vee (B \vee A)$$
$$\top$$

# Sémantique

## Base minimale d'opérateurs

- ▶ La mise en forme normale montre que  $\{\vee, \wedge, \neg\}$  sont suffisants pour représenter toute formule



# Sémantique

## Base minimale d'opérateurs

- ▶ La mise en forme normale montre que  $\{\vee, \wedge, \neg\}$  sont suffisants pour représenter toute formule
- ▶ Il existe des bases minimales d'opérateurs
  - ▶  $\{\wedge, \neg\}$  ou  $\{\vee, \neg\}$  par De Morgan
  - ▶  $\{\rightarrow, \neg\}$  car  $\varphi \vee \psi = \neg\varphi \rightarrow \psi$
  - ▶  $\{\rightarrow, \perp\}$  car  $\neg\varphi = \varphi \rightarrow \perp$

# Déduction naturelle

## Cadre général

- ▶ Axiomatisation par des règles de déduction
- ▶ Approche par chaînage arrière : De la conclusion aux hypothèses
- ▶ Jugement  $\Gamma \vdash \psi$  avec  $\Gamma = \varphi_1, \dots, \varphi_n$   
et  $\varphi_1, \dots, \varphi_n, \psi \in \Phi$   
 $\varphi_i$  sont les hypothèses disponibles pour prouver  $\psi$
- ▶ Sémantique :  $\bigwedge_{i \in [1 \dots n]} \varphi_i \rightarrow \psi$

# Déduction naturelle

## Cadre général

- ▶ Axiomatisation par des règles de déduction
- ▶ Approche par chaînage arrière : De la conclusion aux hypothèses
- ▶ Jugement  $\Gamma \vdash \psi$  avec  $\Gamma = \varphi_1, \dots, \varphi_n$   
et  $\varphi_1, \dots, \varphi_n, \psi \in \Phi$   
 $\varphi_i$  sont les hypothèses disponibles pour prouver  $\psi$
- ▶ Sémantique : 
$$\bigwedge_{i \in [1 \dots n]} \varphi_i \rightarrow \psi$$
- ▶ Axiome de l'hypothèse :  
$$\frac{}{\Gamma, \varphi \vdash \varphi} \text{Hyp}$$

# Déduction naturelle

## Règles de déduction constructive

Introduction	Élimination
$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \rightarrow \psi} I_{\rightarrow}$	$\frac{\Gamma \vdash \varphi \rightarrow \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi} E_{\rightarrow}$
$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} I_{\wedge}$	$\frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi} E_{\wedge}^G \quad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi} E_{\wedge}^D$
$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} I_{\vee}^G \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi} I_{\vee}^D$	$\frac{\Gamma \vdash \varphi \vee \psi \quad \Gamma, \varphi \vdash \chi \quad \Gamma, \psi \vdash \chi}{\Gamma \vdash \chi} E_{\vee}$
$\frac{\Gamma, \varphi \vdash \perp}{\Gamma \vdash \neg \varphi} I_{\neg}$	$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \neg \varphi}{\Gamma \vdash \perp} E_{\neg}$
$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \neg \varphi}{\Gamma \vdash \perp} I_{\perp}$	$\frac{\Gamma \vdash \perp}{\Gamma \vdash \varphi} E_{\perp}$

# Déduction naturelle

## Heuristique/Méthode de preuve

- ▶ Construire la preuve de bas en haut en appliquant par ordre de préférence :
- ▶ Les axiomes (règle de l'hypothèse, ... ) ;
- ▶ Les règles d'élimination sur les hypothèses pour extraire la conclusion si elle figure dans une hypothèse ;
- ▶ Les règles d'introduction pour décomposer la conclusion jusqu'à obtenir un élément disponible dans les hypothèses ou une variable ;
- ▶ La règle  $E_{\perp}$  (preuve par l'absurde constructive) s'il n'est pas possible de faire apparaître en conclusion un élément figurant dans les hypothèses.

# Déduction naturelle

## Heuristique/Méthode de preuve

- ▶ Construire la preuve de bas en haut en appliquant par ordre de préférence :
- ▶ Les axiomes (règle de l'hypothèse, ... ) ;
- ▶ Les règles d'élimination sur les hypothèses pour extraire la conclusion si elle figure dans une hypothèse ;
- ▶ Les règles d'introduction pour décomposer la conclusion jusqu'à obtenir un élément disponible dans les hypothèses ou une variable ;
- ▶ La règle  $E_{\perp}$  (preuve par l'absurde constructive) s'il n'est pas possible de faire apparaître en conclusion un élément figurant dans les hypothèses.
- ▶ Exemple

$$\frac{\frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash B} E_{\wedge}^D \quad \frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash A} E_{\wedge}^G}{A \wedge B \vdash B \wedge A} I_{\wedge}}{\vdash A \wedge B \rightarrow B \wedge A} I_{\rightarrow}$$

# Déduction naturelle

## Heuristique/Méthode de preuve

- ▶ Construire la preuve de bas en haut en appliquant par ordre de préférence :
- ▶ Les axiomes (règle de l'hypothèse, ... ) ;
- ▶ Les règles d'élimination sur les hypothèses pour extraire la conclusion si elle figure dans une hypothèse ;
- ▶ Les règles d'introduction pour décomposer la conclusion jusqu'à obtenir un élément disponible dans les hypothèses ou une variable ;
- ▶ La règle  $E_{\perp}$  (preuve par l'absurde constructive) s'il n'est pas possible de faire apparaître en conclusion un élément figurant dans les hypothèses.
- ▶ Exemple

$$\frac{\frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash B} \text{Hyp } E_{\wedge}^D}{\frac{A \wedge B \vdash B \wedge A}{\vdash A \wedge B \rightarrow B \wedge A} I_{\rightarrow}} I_{\wedge} \quad \left| \quad \frac{\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \rightarrow \psi} I_{\rightarrow}}{\begin{array}{l} \Gamma = \emptyset \\ \varphi = A \wedge B \\ \psi = B \wedge A \end{array}} \right.$$

# Déduction naturelle

## Heuristique/Méthode de preuve

- ▶ Construire la preuve de bas en haut en appliquant par ordre de préférence :
- ▶ Les axiomes (règle de l'hypothèse, ... ) ;
- ▶ Les règles d'élimination sur les hypothèses pour extraire la conclusion si elle figure dans une hypothèse ;
- ▶ Les règles d'introduction pour décomposer la conclusion jusqu'à obtenir un élément disponible dans les hypothèses ou une variable ;
- ▶ La règle  $E_{\perp}$  (preuve par l'absurde constructive) s'il n'est pas possible de faire apparaître en conclusion un élément figurant dans les hypothèses.
- ▶ Exemple

$$\frac{\frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash B} \text{Hyp } E_{\wedge}^D}{A \wedge B \vdash A} \text{Hyp } E_{\wedge}^G \quad \frac{A \wedge B \vdash B \wedge A}{\vdash A \wedge B \rightarrow B \wedge A} I_{\rightarrow}}{\vdash A \wedge B \rightarrow B \wedge A} I_{\wedge} \quad \left| \quad \frac{\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} I_{\wedge}}{\Gamma = A \wedge B} \text{Hyp } E_{\wedge}^D \quad \frac{\Gamma = A \wedge B}{\varphi = B} \text{Hyp } E_{\wedge}^D \quad \frac{\Gamma = A \wedge B}{\psi = A} \text{Hyp } E_{\wedge}^D$$



# Déduction naturelle

## Heuristique/Méthode de preuve

- ▶ Construire la preuve de bas en haut en appliquant par ordre de préférence :
- ▶ Les axiomes (règle de l'hypothèse, ... ) ;
- ▶ Les règles d'élimination sur les hypothèses pour extraire la conclusion si elle figure dans une hypothèse ;
- ▶ Les règles d'introduction pour décomposer la conclusion jusqu'à obtenir un élément disponible dans les hypothèses ou une variable ;
- ▶ La règle  $E_{\perp}$  (preuve par l'absurde constructive) s'il n'est pas possible de faire apparaître en conclusion un élément figurant dans les hypothèses.
- ▶ Exemple

$$\begin{array}{c} \frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash B} \text{Hyp } E_{\wedge}^D}{\frac{A \wedge B \vdash B \wedge A}{\vdash A \wedge B \rightarrow B \wedge A} I_{\rightarrow}} I_{\wedge} \end{array} \quad \left| \quad \begin{array}{c} \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi} E_{\wedge}^D \\ \Gamma = A \wedge B \\ \varphi = A \\ \psi = B \end{array} \right.$$

# Déduction naturelle

## Heuristique/Méthode de preuve

- ▶ Construire la preuve de bas en haut en appliquant par ordre de préférence :
- ▶ Les axiomes (règle de l'hypothèse, ... ) ;
- ▶ Les règles d'élimination sur les hypothèses pour extraire la conclusion si elle figure dans une hypothèse ;
- ▶ Les règles d'introduction pour décomposer la conclusion jusqu'à obtenir un élément disponible dans les hypothèses ou une variable ;
- ▶ La règle  $E_{\perp}$  (preuve par l'absurde constructive) s'il n'est pas possible de faire apparaître en conclusion un élément figurant dans les hypothèses.
- ▶ Exemple

$$\frac{\frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash B} E_{\wedge}^D}{A \wedge B \vdash B \wedge A} I_{\wedge} \quad \frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash A} E_{\wedge}^G}{\vdash A \wedge B \rightarrow B \wedge A} I_{\rightarrow} \quad \left| \quad \frac{}{\Gamma, \varphi \vdash \varphi} \text{Hyp} \right.$$
$$\Gamma = A \wedge B$$
$$\varphi = A \wedge B$$

# Déduction naturelle

## Heuristique/Méthode de preuve

- ▶ Construire la preuve de bas en haut en appliquant par ordre de préférence :
- ▶ Les axiomes (règle de l'hypothèse, ... ) ;
- ▶ Les règles d'élimination sur les hypothèses pour extraire la conclusion si elle figure dans une hypothèse ;
- ▶ Les règles d'introduction pour décomposer la conclusion jusqu'à obtenir un élément disponible dans les hypothèses ou une variable ;
- ▶ La règle  $E_{\perp}$  (preuve par l'absurde constructive) s'il n'est pas possible de faire apparaître en conclusion un élément figurant dans les hypothèses.
- ▶ Exemple

$$\frac{\frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash B} \text{Hyp}}{A \wedge B \vdash B} E_{\wedge}^D \quad \frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash A} \text{Hyp}}{A \wedge B \vdash A} E_{\wedge}^G}{\frac{A \wedge B \vdash B \wedge A}{\vdash A \wedge B \rightarrow B \wedge A} I_{\rightarrow}} I_{\wedge}$$

$$\frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi} E_{\wedge}^G$$
$$\Gamma = A \wedge B$$
$$\varphi = A$$
$$\psi = B$$

# Déduction naturelle

## Heuristique/Méthode de preuve

- ▶ Construire la preuve de bas en haut en appliquant par ordre de préférence :
- ▶ Les axiomes (règle de l'hypothèse, ... ) ;
- ▶ Les règles d'élimination sur les hypothèses pour extraire la conclusion si elle figure dans une hypothèse ;
- ▶ Les règles d'introduction pour décomposer la conclusion jusqu'à obtenir un élément disponible dans les hypothèses ou une variable ;
- ▶ La règle  $E_{\perp}$  (preuve par l'absurde constructive) s'il n'est pas possible de faire apparaître en conclusion un élément figurant dans les hypothèses.
- ▶ Exemple

$$\frac{\frac{\frac{A \wedge B \vdash A \wedge B}{A \wedge B \vdash B} \text{Hyp } E_{\wedge}^D}{\frac{A \wedge B \vdash B \wedge A}{\vdash A \wedge B \rightarrow B \wedge A} I_{\rightarrow}} I_{\wedge} \quad \left| \quad \frac{\frac{}{\Gamma, \varphi \vdash \varphi} \text{Hyp } E_{\wedge}^G}{\begin{array}{l} \Gamma = A \wedge B \\ \varphi = A \wedge B \end{array}} \right.$$

# Déduction naturelle

## Logique constructive et classique

- ▶ Logique constructive : Approche philosophique
- ▶ Interdiction du tiers-exclus (Axiome  $\varphi \vee \neg\varphi$ )
- ▶ Interdiction de l'axiome du choix

# Déduction naturelle

## Logique constructive et classique

- ▶ Logique constructive : Approche philosophique
- ▶ Interdiction du tiers-exclus (Axiome  $\varphi \vee \neg\varphi$ )
- ▶ Interdiction de l'axiome du choix
- ▶ La logique classique autorise ces principes à travers les règles :

Tiers-exclu	Preuve par l'absurde
$\frac{}{\Gamma \vdash \varphi \vee \neg\varphi} \text{ TE}$	$\frac{\Gamma, \neg\varphi \vdash \perp}{\Gamma \vdash \varphi} \text{ Abs}$

# Déduction naturelle

## Logique constructive et classique

- ▶ Logique constructive : Approche philosophique
- ▶ Interdiction du tiers-exclus (Axiome  $\varphi \vee \neg\varphi$ )
- ▶ Interdiction de l'axiome du choix
- ▶ La logique classique autorise ces principes à travers les règles :

Tiers-exclu	Preuve par l'absurde
$\frac{}{\Gamma \vdash \varphi \vee \neg\varphi} \text{ TE}$	$\frac{\Gamma, \neg\varphi \vdash \perp}{\Gamma \vdash \varphi} \text{ Abs}$

- ▶ La preuve par l'absurde classique exploite le tiers-exclu

$$\frac{\frac{}{\Gamma \vdash \varphi \vee \neg\varphi} \text{ TE} \quad \frac{}{\Gamma, \varphi \vdash \varphi} \text{ Hyp} \quad \frac{\Gamma, \neg\varphi \vdash \perp}{\Gamma, \neg\varphi \vdash \varphi} \begin{matrix} E_{\perp} \\ E_{\vee} \end{matrix}}{\Gamma \vdash \varphi}$$

# Logique des propositions

## Conclusion

La logique des propositions est :

- ▶ Complète sémantiquement et axiomatiquement
- ▶ Consistante sémantiquement
- ▶ Correcte axiomatiquement
- ▶ Décidable mécaniquement
- ▶ **Mais** Très peu expressive
- ▶ Introduction des quantificateurs, des relations et des structures :  
Logique des prédicats



# Logique des propositions

Mise en pratique

## L'assistant de preuve Coq

- ▶ Développé au sein d'INRIA
- ▶ Système  $F$  :  $\lambda$ -calcul typé second ordre (Girard et Reynolds)
- ▶ Calcul des constructions inductives (Coquand)
- ▶ Correspondance de Curry-Howard
  - ▶ Formule = Type
  - ▶ Preuve = Programme

## Le langage de développement prouvé Why3

- ▶ Développé au sein du LRI et d'INRIA
- ▶ Logique des prédicats du premier ordre et Logique de Hoare
- ▶ Passerelle vers de nombreux outils de vérification :
  - ▶ Automatique : SAT solver (résolution par saturation), SMT (SAT Modulo Theory)
  - ▶ Semi-automatique : Assistants de preuve

# Logique des prédicats

## Logique et Structure

- ▶ Objectif mathématique : Modélisation des structures algébriques
- ▶ Objectif informatique : Modélisation des données et des opérations
- ▶ Extension de la logique des propositions :
  - ▶ Univers (objets mathématiques ou informatiques) :  $\mathcal{U}$
  - ▶ Algèbre de termes (représentation des objets) : constantes et opérateurs sur  $\mathcal{U}$
  - ▶ Quantificateurs pour variables dans  $\mathcal{U}$  :  $\forall, \exists$
  - ▶ Relations sur  $\mathcal{U}$  (permet aussi de représenter les termes)
- ▶ Sémantique : Logique des propositions + Modèles des structures

# Syntaxe

## Vision algébrique – Éléments lexicaux

- ▶ Extension de la logique des propositions  $\perp \top \neg \wedge \vee \rightarrow \leftrightarrow \mathcal{P} ( )$
- ▶ Ensembles dénombrables de symboles :
  - ▶ Variables  $\mathcal{V}$
  - ▶ Relations (prédicats)  $\mathcal{R}$  munie d'une arité  $\in \mathbb{N}^*$
  - ▶ Propositions  $\mathcal{P}$  (relations d'arité 0)
  - ▶ Fonctions  $\mathcal{F}$  munie d'une arité  $\in \mathbb{N}^*$
  - ▶ Constantes  $\mathcal{C}$  (fonctions d'arité 0)
- ▶ Lieurs :  $\forall \exists$
- ▶ Paramètres des relations et fonctions :  $( , )$

# Syntaxe

## Vision algébrique – Éléments grammaticaux

- ▶  $\mathcal{T} = \bigcup_{i \in \mathbb{N}} \mathcal{T}_i$  ensemble dénombrable stratifié des termes bien formés
- ▶  $\Phi = \bigcup_{i \in \mathbb{N}} \Phi_i$  ensemble dénombrable stratifié des formules bien formées
- ▶ Les constantes et les fonctions avec leurs paramètres sont des termes bien formés ;
- ▶ Les variables sont soit des termes bien formés, soit des relations, soit des fonctions ;
- ▶ Les relations avec leurs paramètres sont des formules bien formées ;
- ▶ Les lieurs définissant une variable dans une formule bien formée (portée de la variable) sont des formules bien formées ;
- ▶ Les lieurs sont moins prioritaires que tous les autres opérateurs ;
- ▶ Les relations et fonctions prennent comme paramètre un nombre de termes bien formés égal à leur arité.

# Syntaxe

## Termes : Vision déductive

Soient  $\mathcal{V}, \mathcal{C}, \mathcal{F}$  des ensembles dénombrables de symboles :

$\mathcal{F} = \bigcup_{i \in \mathbb{N}} \mathcal{F}_i$  se décompose selon l'arité du symbole

Notons  $\mathcal{T} = \bigcup_{i \in \mathbb{N}} \mathcal{T}_i$  l'ensemble stratifié des termes bien formés :

Déductions	Version classique
	$\frac{e \in \mathcal{C} \cup \mathcal{V}}{e \in \mathcal{T}}$
$i \in \mathbb{N}^*$	$\frac{f \in \mathcal{V} \cup \mathcal{F}_i \quad t_1 \in \mathcal{T} \quad t_i \in \mathcal{T}}{f(t_1, \dots, t_i) \in \mathcal{T}}$

Exemple :  $\mathcal{V} = \{n\}, \mathcal{C} = \{un, deux\}, \mathcal{F}_2 = \{somme, produit\}$

$somme(produit(deux, n), un)$

# Syntaxe

## Prédicats : Vision déductive

Soient  $\mathcal{V}, \mathcal{R}$  des ensembles dénombrables de symboles :

$\mathcal{R} = \bigcup_{i \in \mathbb{N}} \mathcal{R}_i$  se décompose selon l'arité du symbole

Notons  $\Phi = \bigcup_{i \in \mathbb{N}} \Phi_i$  l'ensemble stratifié des formules bien formées :

Déductions	Version classique
	$\frac{x \in \mathcal{V}}{x \in \Phi}$
$i \in \mathbb{N}^*$	$\frac{p \in \mathcal{V} \cup \mathcal{R}_i \quad t_1 \in \mathcal{T} \quad t_i \in \mathcal{T}}{p(t_1, \dots, t_i) \in \Phi}$
$q \in \{\forall, \exists\}$	$\frac{x \in \mathcal{V} \quad \varphi \in \Phi}{q x. \varphi \in \Phi}$

# Syntaxe

## Précautions et Remarques

- ▶ **Attention :** Les notations suivantes ne font pas partie de la syntaxe des formules bien formées en logique des prédicats
  - ▶  $\vec{x} \in e$  représentée par  $e(\vec{x})$
  - ▶  $\forall x \in e. \varphi$  représentée par  $\forall x. e(x) \rightarrow \varphi$
  - ▶  $\exists x \in e. \varphi$  représentée par  $\exists x. e(x) \wedge \varphi$
- ▶ Remarque : les constantes et les fonctions peuvent être représentées par des relations
  - ▶ Une constante  $c \in \mathcal{C}$  peut être représentée par une relation  $r_c$  qui teste si une variable a la valeur  $c$
  - ▶ Une fonction  $f \in \mathcal{F}_n$  peut être représentée par une relation  $r_c$  qui satisfait :  $\forall x_1 \dots \forall x_n. r_c(x_1, \dots, x_n, f(x_1, \dots, x_n))$

# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations  
(adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)



# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations (adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)
- ▶ Exemples : Tous les hommes sont mortels, or Socrate est un homme donc Socrate est mortel.
- ▶ Variables :  $E$  = un élément quelconque;  $S$  = Socrate.
- ▶ Prédicats :  $H$  = est un homme;  $M$  = est mortel.
- ▶ Formule :

# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations (adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)
- ▶ Exemples : **Tous les** hommes sont mortels, or Socrate est un homme donc Socrate est mortel.
- ▶ Variables :  $E$  = un élément quelconque;  $S$  = Socrate.
- ▶ Prédicats :  $H$  = est un homme;  $M$  = est mortel.
- ▶ Formule :  $\forall E$ .

# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations (adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)
- ▶ Exemples : Tous les hommes sont mortels, or **Socrate** est un homme donc Socrate est mortel.
- ▶ Variables :  $E$  = un élément quelconque;  $S$  = Socrate.
- ▶ Prédicats :  $H$  = est un homme;  $M$  = est mortel.
- ▶ Formule :  $\forall E. \exists S.$

# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations (adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)
- ▶ Exemples : Tous les **hommes** sont mortels, or Socrate est un homme donc Socrate est mortel.
- ▶ Variables :  $E$  = un élément quelconque;  $S$  = Socrate.
- ▶ Prédicats :  $H$  = est un homme;  $M$  = est mortel.
- ▶ Formule :  $\forall E. \exists S. ((H(E)$

# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations (adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)
- ▶ Exemples : Tous les hommes **sont mortels**, or Socrate est un homme donc Socrate est mortel.
- ▶ Variables :  $E$  = un élément quelconque;  $S$  = Socrate.
- ▶ Prédicats :  $H$  = est un homme;  $M$  = est mortel.
- ▶ Formule :  $\forall E. \exists S. ((H(E) \rightarrow M(E)))$

# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations (adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)
- ▶ Exemples : Tous les hommes sont mortels, **or** Socrate est un homme donc Socrate est mortel.
- ▶ Variables :  $E$  = un élément quelconque;  $S$  = Socrate.
- ▶ Prédicats :  $H$  = est un homme;  $M$  = est mortel.
- ▶ Formule :  $\forall E. \exists S. ((H(E) \rightarrow M(E)) \wedge$

# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations (adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)
- ▶ Exemples : Tous les hommes sont mortels, or Socrate **est un homme** donc Socrate est mortel.
- ▶ Variables :  $E$  = un élément quelconque;  $S$  = Socrate.
- ▶ Prédicats :  $H$  = est un homme;  $M$  = est mortel.
- ▶ Formule :  $\forall E. \exists S. ((H(E) \rightarrow M(E)) \wedge H(S))$

# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations (adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)
- ▶ Exemples : Tous les hommes sont mortels, or Socrate est un homme **donc** Socrate est mortel.
- ▶ Variables :  $E$  = un élément quelconque;  $S$  = Socrate.
- ▶ Prédicats :  $H$  = est un homme;  $M$  = est mortel.
- ▶ Formule :  $\forall E. \exists S. ((H(E) \rightarrow M(E)) \wedge H(S)) \rightarrow$



# Exploitation pour la modélisation

## Logique de prédicats

- ▶ Modélisation d'énoncé en langage naturel
- ▶ Décomposition de l'énoncé en prédicats combinées par les opérateurs
- ▶ Opérateurs : conjonction de coordination, de subordination
- ▶ Propositions : prédicats sans paramètres
- ▶ Prédicats : parties de l'énoncé liées par les relations (adjectifs/sujets, verbe/sujet, verbe/sujet/compléments, ...)
- ▶ Exemples : Tous les hommes sont mortels, or Socrate est un homme donc **Socrate est mortel**.
- ▶ Variables :  $E$  = un élément quelconque;  $S$  = Socrate.
- ▶ Prédicats :  $H$  = est un homme;  $M$  = est mortel.
- ▶ Formule :  $\forall E. \exists S. ((H(E) \rightarrow M(E)) \wedge H(S)) \rightarrow M(S)$

# Syntaxe

## Ordre de la logique des prédicats

- ▶ Ordre supérieur : les lieurs peuvent quantifier des termes, des relations, des propositions, des fonctions, des constantes
- ▶ Premier ordre (First Order Logic) : Les lieurs ne peuvent quantifier que des termes
- ▶ Exemple du premier ordre :  
 $\mathcal{V} = \{m, n\}, \mathcal{R}_1 = \{entier\}, \mathcal{R}_2 = \{egal\}$   
 $\forall m. (entier(m) \rightarrow (impair(m) \leftrightarrow$   
 $(\exists n. (entier(n) \wedge egal(m, somme(produit(deux, n), un))))))$
- ▶ Exemple du second ordre :  
 $g$  muni de l'opération binaire  $o$  est un groupe

$$\left\{ \begin{array}{l} \forall g. \forall o. groupe(g, o) \leftrightarrow \\ \forall x_1. \forall x_2. g(x_1) \wedge g(x_2) \rightarrow g(o(x_1, x_2)) \\ \wedge \end{array} \right.$$

# Syntaxe

## Ordre de la logique des prédicats

- ▶ Ordre supérieur : les lieurs peuvent quantifier des termes, des relations, des propositions, des fonctions, des constantes
- ▶ Premier ordre (First Order Logic) : Les lieurs ne peuvent quantifier que des termes
- ▶ Exemple du premier ordre :  
 $\mathcal{V} = \{m, n\}, \mathcal{R}_1 = \{\text{entier}\}, \mathcal{R}_2 = \{\text{egal}\}$   
 $\forall m. (\text{entier}(m) \rightarrow (\text{impair}(m) \leftrightarrow$   
 $(\exists n. (\text{entier}(n) \wedge \text{egal}(m, \text{somme}(\text{produit}(\text{deux}, n), \text{un}))))))$
- ▶ Exemple du second ordre :  
 $g$  muni de l'opération binaire  $o$  est un groupe

$$\begin{aligned} & \forall g. \forall o. \text{groupe}(g, o) \leftrightarrow \\ & \left\{ \begin{array}{l} \forall x_1. \forall x_2. g(x_1) \wedge g(x_2) \rightarrow g(o(x_1, x_2)) \\ \wedge \exists e. g(e) \wedge \left\{ \begin{array}{l} \forall x. g(x) \rightarrow \text{egal}(o(x, e), x) \wedge \text{egal}(o(e, x), x) \\ \wedge \end{array} \right. \end{array} \right. \end{aligned}$$

# Syntaxe

## Ordre de la logique des prédicats

- ▶ Ordre supérieur : les lieurs peuvent quantifier des termes, des relations, des propositions, des fonctions, des constantes
- ▶ Premier ordre (First Order Logic) : Les lieurs ne peuvent quantifier que des termes
- ▶ Exemple du premier ordre :  
 $\mathcal{V} = \{m, n\}, \mathcal{R}_1 = \{\text{entier}\}, \mathcal{R}_2 = \{\text{egal}\}$   
 $\forall m. (\text{entier}(m) \rightarrow (\text{impair}(m) \leftrightarrow$   
 $(\exists n. (\text{entier}(n) \wedge \text{egal}(m, \text{somme}(\text{produit}(\text{deux}, n), \text{un}))))))$
- ▶ Exemple du second ordre :  
 $g$  muni de l'opération binaire  $o$  est un groupe

$$\begin{aligned} & \forall g. \forall o. \text{groupe}(g, o) \leftrightarrow \\ & \left\{ \begin{array}{l} \forall x_1. \forall x_2. g(x_1) \wedge g(x_2) \rightarrow g(o(x_1, x_2)) \\ \wedge \exists e. g(e) \wedge \left\{ \begin{array}{l} \forall x. g(x) \rightarrow \text{egal}(o(x, e), x) \wedge \text{egal}(o(e, x), x) \\ \wedge \forall x_1. \forall x_2. \forall x_3. g(x_1) \wedge g(x_2) \wedge g(x_3) \rightarrow \\ \text{egal}(o(o(x_1, x_2), x_3), o(x_1, o(x_2, x_3))) \end{array} \right. \\ \wedge \end{array} \right. \end{aligned}$$

# Syntaxe

## Ordre de la logique des prédicats

- ▶ Ordre supérieur : les lieurs peuvent quantifier des termes, des relations, des propositions, des fonctions, des constantes
- ▶ Premier ordre (First Order Logic) : Les lieurs ne peuvent quantifier que des termes

- ▶ Exemple du premier ordre :

$$\mathcal{V} = \{m, n\}, \mathcal{R}_1 = \{\text{entier}\}, \mathcal{R}_2 = \{\text{egal}\}$$

$$\forall m. (\text{entier}(m) \rightarrow (\text{impair}(m) \leftrightarrow$$

$$(\exists n. (\text{entier}(n) \wedge \text{egal}(m, \text{somme}(\text{produit}(\text{deux}, n), \text{un}))))))$$

- ▶ Exemple du second ordre :

$g$  muni de l'opération binaire  $o$  est un groupe

$$\forall g. \forall o. \text{groupe}(g, o) \leftrightarrow$$

$$\left\{ \begin{array}{l} \forall x_1. \forall x_2. g(x_1) \wedge g(x_2) \rightarrow g(o(x_1, x_2)) \\ \wedge \exists e. g(e) \wedge \left\{ \begin{array}{l} \forall x. g(x) \rightarrow \text{egal}(o(x, e), x) \wedge \text{egal}(o(e, x), x) \\ \wedge \forall x_1. \forall x_2. \forall x_3. g(x_1) \wedge g(x_2) \wedge g(x_3) \rightarrow \\ \text{egal}(o(o(x_1, x_2), x_3), o(x_1, o(x_2, x_3))) \\ \wedge \forall x_1. g(x_1) \rightarrow \exists x_2. g(x_2) \wedge \text{egal}(o(x_1, x_2), e) \wedge \text{egal}(o(x_2, x_1), e) \end{array} \right. \end{array} \right.$$

# Variables libres

$VL(\varphi)$  : Variables de  $\varphi$  qui ne sont pas liées par  $\forall$  ou  $\exists$

$VL(c) = \emptyset$	$c \in \{\perp, \top\} \cup \mathcal{P}$
$VL((op\ \varphi)) = VL(\varphi)$	$op \in \{\neg\}$
$VL((\varphi\ op\ \psi)) = VL(\varphi) \cup VL(\psi)$	$op \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$
$VL(x) = \{x\}$	$x \in \mathcal{V}$
$VL(c) = \emptyset$	$c \in \mathcal{C}$
$VL(op(t_1, \dots, t_n)) = \bigcup_{i \in [1 \dots n]} VL(t_i)$	$op \in \mathcal{R}_n \cup \mathcal{F}_n$
$VL((q\ x.\ \varphi)) = VL(\varphi) \setminus \{x\}$	$q \in \{\forall, \exists\}$

Exemple :

$$VL(\forall x. (\varphi \leftrightarrow \exists y. \psi))$$

# Variables libres

$VL(\varphi)$  : Variables de  $\varphi$  qui ne sont pas liées par  $\forall$  ou  $\exists$

$VL(c) = \emptyset$	$c \in \{\perp, \top\} \cup \mathcal{P}$
$VL(op\ \varphi) = VL(\varphi)$	$op \in \{\neg\}$
$VL((\varphi\ op\ \psi)) = VL(\varphi) \cup VL(\psi)$	$op \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$
$VL(x) = \{x\}$	$x \in \mathcal{V}$
$VL(c) = \emptyset$	$c \in \mathcal{C}$
$VL(op(t_1, \dots, t_n)) = \bigcup_{i \in [1 \dots n]} VL(t_i)$	$op \in \mathcal{R}_n \cup \mathcal{F}_n$
$VL((q\ x.\ \varphi)) = VL(\varphi) \setminus \{x\}$	$q \in \{\forall, \exists\}$

Exemple :

$$\begin{aligned} & VL(\forall x. (\varphi \leftrightarrow \exists y. \psi)) \\ &= VL(\varphi \leftrightarrow \exists y. \psi) \setminus \{x\} \end{aligned}$$

# Variables libres

$VL(\varphi)$  : Variables de  $\varphi$  qui ne sont pas liées par  $\forall$  ou  $\exists$

$VL(c) = \emptyset$	$c \in \{\perp, \top\} \cup \mathcal{P}$
$VL(op\ \varphi) = VL(\varphi)$	$op \in \{\neg\}$
$VL((\varphi\ op\ \psi)) = VL(\varphi) \cup VL(\psi)$	$op \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$
$VL(x) = \{x\}$	$x \in \mathcal{V}$
$VL(c) = \emptyset$	$c \in \mathcal{C}$
$VL(op(t_1, \dots, t_n)) = \bigcup_{i \in [1 \dots n]} VL(t_i)$	$op \in \mathcal{R}_n \cup \mathcal{F}_n$
$VL((q\ x.\ \varphi)) = VL(\varphi) \setminus \{x\}$	$q \in \{\forall, \exists\}$

Exemple :

$$\begin{aligned} & VL(\forall x. (\varphi \leftrightarrow \exists y. \psi)) \\ &= VL(\varphi \leftrightarrow \exists y. \psi) \setminus \{x\} \\ &= (VL(\varphi) \cup VL(\exists y. \psi)) \setminus \{x\} \end{aligned}$$



# Variables libres

$VL(\varphi)$  : Variables de  $\varphi$  qui ne sont pas liées par  $\forall$  ou  $\exists$

$VL(c) = \emptyset$	$c \in \{\perp, \top\} \cup \mathcal{P}$
$VL(op\ \varphi) = VL(\varphi)$	$op \in \{\neg\}$
$VL((\varphi\ op\ \psi)) = VL(\varphi) \cup VL(\psi)$	$op \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$
$VL(x) = \{x\}$	$x \in \mathcal{V}$
$VL(c) = \emptyset$	$c \in \mathcal{C}$
$VL(op(t_1, \dots, t_n)) = \bigcup_{i \in [1 \dots n]} VL(t_i)$	$op \in \mathcal{R}_n \cup \mathcal{F}_n$
$VL((q\ x.\ \varphi)) = VL(\varphi) \setminus \{x\}$	$q \in \{\forall, \exists\}$

Exemple :

$$\begin{aligned} & VL(\forall x. (\varphi \leftrightarrow \exists y. \psi)) \\ &= VL(\varphi \leftrightarrow \exists y. \psi) \setminus \{x\} \\ &= (VL(\varphi) \cup VL(\exists y. \psi)) \setminus \{x\} \\ &= (VL(\varphi) \cup (VL(\psi) \setminus \{y\})) \setminus \{x\} \end{aligned}$$

# Substitution

$[t/x]\varphi$  remplace  $x \in \mathcal{V}$  par  $t \in \mathcal{T}$  dans  $\varphi \in \Phi$

$([t/x] c) = c$	$c \in \{\perp, \top\} \cup \mathcal{P}$
$([t/x] (op \varphi)) = op([t/x] \varphi)$	$op \in \{\neg\}$
$([t/x] (\varphi op \psi)) = ([t/x] \varphi) op ([t/x] \psi)$	$op \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$
$([t/x] x) = t$	
$([t/x] y) = y$	$x \neq y$
$([t/x] c) = c$	$c \in \mathcal{C}$
$([t/x] op(t_1, \dots, t_n)) = op([t/x] t_1, \dots, [t/x] t_n)$	$op \in \mathcal{R}_n \cup \mathcal{F}_n$
$([t/x] (q x. \varphi)) = q x. \varphi$	$q \in \{\forall, \exists\}$
$([t/x] (q y. \varphi)) = q y. ([t/x] \varphi)$	$\begin{cases} q \in \{\forall, \exists\} \\ x \neq y \\ y \notin VL(t) \end{cases}$
$([t/x] (q y. \varphi)) = q z. ([t/x] ([z/y] \varphi))$	$\begin{cases} q \in \{\forall, \exists\} \\ x \neq y \\ z \text{ inutilisée} \end{cases}$

# Substitution

## Exemple

$$[f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))$$

# Substitution

## Exemple

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \end{aligned}$$

# Substitution

## Exemple

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (([f(x, y)/x] x) \rightarrow ([f(x, y)/x] y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \end{aligned}$$

# Substitution

## Exemple

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (([f(x, y)/x] x) \rightarrow ([f(x, y)/x] y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \end{aligned}$$

# Substitution

## Exemple

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (([f(x, y)/x] x) \rightarrow ([f(x, y)/x] y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] [z/y] (x \vee ((\forall x. \varphi) \rightarrow y)))) \end{aligned}$$

# Substitution

## Exemple

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (([f(x, y)/x] x) \rightarrow ([f(x, y)/x] y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] [z/y] (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (([z/y] x) \vee ([z/y] ((\forall x. \varphi) \rightarrow y))))) \end{aligned}$$



# Substitution

## Exemple

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (([f(x, y)/x] x) \rightarrow ([f(x, y)/x] y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] [z/y] (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (([z/y] x) \vee ([z/y] ((\forall x. \varphi) \rightarrow y))))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (x \vee (([z/y] (\forall x. \varphi)) \rightarrow ([z/y] y))))) \end{aligned}$$

# Substitution

## Example

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (([f(x, y)/x] x) \rightarrow ([f(x, y)/x] y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] [z/y] (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (([z/y] x) \vee ([z/y] ((\forall x. \varphi) \rightarrow y))))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (x \vee (([z/y] (\forall x. \varphi)) \rightarrow ([z/y] y))))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (x \vee ((\forall x. ([z/y] \varphi)) \rightarrow z)))) \end{aligned}$$

# Substitution

## Example

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (([f(x, y)/x] x) \rightarrow ([f(x, y)/x] y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] [z/y] (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (([z/y] x) \vee ([z/y] ((\forall x. \varphi) \rightarrow y))))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (x \vee (([z/y] (\forall x. \varphi)) \rightarrow ([z/y] y))))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (x \vee ((\forall x. ([z/y] \varphi)) \rightarrow z)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ((([f(x, y)/x] x) \vee ([f(x, y)/x] ((\forall x. ([z/y] \varphi)) \rightarrow z)))) \end{aligned}$$

# Substitution

## Example

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (([f(x, y)/x] x) \rightarrow ([f(x, y)/x] y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] [z/y] (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (([z/y] x) \vee ([z/y] ((\forall x. \varphi) \rightarrow y))))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (x \vee (([z/y] (\forall x. \varphi)) \rightarrow ([z/y] y))))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (x \vee ((\forall x. ([z/y] \varphi)) \rightarrow z)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ((([f(x, y)/x] x) \vee ([f(x, y)/x] ((\forall x. ([z/y] \varphi)) \rightarrow z)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. (f(x, y) \vee (([f(x, y)/x] (\forall x. ([z/y] \varphi))) \rightarrow ([f(x, y)/x] z))) \end{aligned}$$

# Substitution

## Example

$$\begin{aligned} & [f(x, y)/x] ((x \rightarrow y) \wedge \exists y. (x \vee ((\forall x. \varphi) \rightarrow y))) \\ &= ([f(x, y)/x] (x \rightarrow y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (([f(x, y)/x] x) \rightarrow ([f(x, y)/x] y)) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge ([f(x, y)/x] (\exists y. (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] [z/y] (x \vee ((\forall x. \varphi) \rightarrow y)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (([z/y] x) \vee ([z/y] ((\forall x. \varphi) \rightarrow y))))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (x \vee (([z/y] (\forall x. \varphi)) \rightarrow ([z/y] y))))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ([f(x, y)/x] (x \vee ((\forall x. ([z/y] \varphi)) \rightarrow z)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. ((([f(x, y)/x] x) \vee ([f(x, y)/x] ((\forall x. ([z/y] \varphi)) \rightarrow z)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. (f(x, y) \vee ((([f(x, y)/x] (\forall x. ([z/y] \varphi))) \rightarrow ([f(x, y)/x] z)))) \\ &= (f(x, y) \rightarrow y) \wedge (\exists z. (f(x, y) \vee ((\forall x. ([z/y] \varphi)) \rightarrow z))) \end{aligned}$$

# Sémantique

## Théorie des modèles

- ▶ Extension de la logique des propositions
- ▶ Tables de vérité pour la partie logique des propositions
- ▶ Modèle sémantique  $\mathcal{M}$  pour les constantes, fonctions et prédicats
  - ▶ L'univers  $\mathcal{U}$  contient les objets représentés par les termes
  - ▶  $\mathcal{M}(c) \in \mathcal{U}$  est la sémantique de la constante  $c \in \mathcal{C}$
  - ▶  $\mathcal{M}(f) \in \mathcal{U}^n \mapsto \mathcal{U}$  est la sémantique de la fonction  $f \in \mathcal{F}_n$
  - ▶  $\mathcal{M}(p) \subseteq \mathcal{U}^n$  est la sémantique de la relation  $p \in \mathcal{R}_n$
- ▶ Expansion pour les quantificateurs :

$\mathcal{U} \neq \emptyset$	$\forall x. \varphi = \bigwedge_{x \in \mathcal{U}} \varphi$	$\exists x. \varphi = \bigvee_{x \in \mathcal{U}} \varphi$
$\mathcal{U} = \emptyset$	$\forall x. \varphi = \top$	$\exists x. \varphi = \perp$

- ▶ Problème : Expansion infinie si  $\mathcal{U}$  est infini requiert raisonnement symbolique
- ▶  $\mathcal{M} \models \varphi$  :  $\varphi$  est valide pour l'interprétation donnée par le modèle  $\mathcal{M}$

# Sémantique

## Relation d'équivalence

$\varphi = \psi$  si et seulement si  $\varphi$  et  $\psi$  sont valides pour les mêmes modèles

$$\forall \mathcal{M}. (\mathcal{M} \models \varphi) \leftrightarrow (\mathcal{M} \models \psi)$$

$$\forall x. \varphi = \varphi$$

$$x \notin VL(\varphi) \wedge \mathcal{U} \neq \emptyset$$

$$\exists x. \varphi = \varphi$$

$$x \notin VL(\varphi) \wedge \mathcal{U} \neq \emptyset$$

$$\forall x. \varphi = \forall y. [y/x] \varphi$$

$y$  inutilisée

$$\exists x. \varphi = \exists y. [y/x] \varphi$$

$y$  inutilisée

$$\forall x. (\forall y. \varphi) = \forall y. (\forall x. \varphi)$$

$$\exists x. (\exists y. \varphi) = \exists y. (\exists x. \varphi)$$

$$\neg(\forall x. \varphi) = \exists x. (\neg \varphi)$$

$$\neg(\exists x. \varphi) = \forall x. (\neg \varphi)$$

$$\forall x. (\varphi \rightarrow \psi) = (\exists x. \varphi) \rightarrow \psi$$

$$x \notin VL(\psi)$$

$$\forall x. (\varphi \rightarrow \psi) = \varphi \rightarrow (\forall x. \psi)$$

$$x \notin VL(\varphi)$$

$$\exists x. (\varphi \rightarrow \psi) = (\forall x. \varphi) \rightarrow \psi$$

$$x \notin VL(\psi)$$

$$\exists x. (\varphi \rightarrow \psi) = \varphi \rightarrow (\exists x. \psi)$$

$$x \notin VL(\varphi)$$

$$\forall x. (\varphi \wedge \psi) = (\forall x. \varphi) \wedge (\forall x. \psi)$$

$$\forall x. (\varphi \vee \psi) = (\forall x. \varphi) \vee \psi$$

$$x \notin VL(\psi)$$

$$\exists x. (\varphi \vee \psi) = (\exists x. \varphi) \vee (\exists x. \psi)$$

$$\exists x. (\varphi \wedge \psi) = (\exists x. \varphi) \wedge \psi$$

$$x \notin VL(\psi)$$

# Sémantique

## Formes normales

- Forme normale prénexe : les règles précédentes permettent de remonter les quantificateurs en tête de formule

$$(\forall x. \varphi) \rightarrow (\exists x. \psi)$$



# Sémantique

## Formes normales

- Forme normale prénexe : les règles précédentes permettent de remonter les quantificateurs en tête de formule

$$\begin{aligned} & (\forall x. \varphi) \rightarrow (\exists x. \psi) \\ &= (\forall x. \varphi) \rightarrow (\exists y. [y/x] \psi) \end{aligned}$$

# Sémantique

## Formes normales

- Forme normale prénexe : les règles précédentes permettent de remonter les quantificateurs en tête de formule

$$\begin{aligned} & (\forall x. \varphi) \rightarrow (\exists x. \psi) \\ &= (\forall x. \varphi) \rightarrow (\exists y. [y/x] \psi) \\ &= \forall x. \varphi \rightarrow (\exists y. [y/x] \psi) \end{aligned}$$

# Sémantique

## Formes normales

- Forme normale prénexe : les règles précédentes permettent de remonter les quantificateurs en tête de formule

$$\begin{aligned} & (\forall x. \varphi) \rightarrow (\exists x. \psi) \\ &= (\forall x. \varphi) \rightarrow (\exists y. [y/x] \psi) \\ &= \forall x. \varphi \rightarrow (\exists y. [y/x] \psi) \\ &= \forall x. \exists y. \varphi \rightarrow [y/x] \psi \end{aligned}$$

# Sémantique

## Formes normales

- Forme normale prénexe : les règles précédentes permettent de remonter les quantificateurs en tête de formule

$$\begin{aligned}(\forall x. \varphi) &\rightarrow (\exists x. \psi) \\&= (\forall x. \varphi) \rightarrow (\exists y. [y/x] \psi) \\&= \forall x. \varphi \rightarrow (\exists y. [y/x] \psi) \\&= \forall x. \exists y. \varphi \rightarrow [y/x] \psi\end{aligned}$$

- Transformation de Skolem :
  - Soit  $\vec{y} = VL(\forall x_1 \dots \forall x_n. \exists z. \varphi)$
  - Soit  $f_n^{SK}$  un nouveau symbole fonctionnel d'arité  $n = |\vec{x}| + |\vec{y}|$

$$\forall x_1 \dots \forall x_n. \exists z. \varphi = \forall x_1 \dots \forall x_n. [f_n^{SK}(\vec{x}, \vec{y})/z] \varphi \text{ avec } \vec{x} = \langle x_1, \dots, x_n \rangle$$

- La transformation de Skolem permet de remplacer les quantificateurs existentiels dans une forme normale prénexe par des termes exprimant la dépendance de la variable quantifiée avec les autres variables

$$\forall x. \exists y. \varphi \rightarrow [y/x] \psi$$

# Sémantique

## Formes normales

- Forme normale prénexe : les règles précédentes permettent de remonter les quantificateurs en tête de formule

$$\begin{aligned}(\forall x. \varphi) &\rightarrow (\exists x. \psi) \\&= (\forall x. \varphi) \rightarrow (\exists y. [y/x] \psi) \\&= \forall x. \varphi \rightarrow (\exists y. [y/x] \psi) \\&= \forall x. \exists y. \varphi \rightarrow [y/x] \psi\end{aligned}$$

- Transformation de Skolem :

- Soit  $\vec{y} = VL(\forall x_1 \dots \forall x_n. \exists z. \varphi)$
- Soit  $f_n^{SK}$  un nouveau symbole fonctionnel d'arité  $n = |\vec{x}| + |\vec{y}|$

$$\forall x_1 \dots \forall x_n. \exists z. \varphi = \forall x_1 \dots \forall x_n. [f_n^{SK}(\vec{x}, \vec{y})/z] \varphi \text{ avec } \vec{x} = \langle x_1, \dots, x_n \rangle$$

- La transformation de Skolem permet de remplacer les quantificateurs existentiels dans une forme normale prénexe par des termes exprimant la dépendance de la variable quantifiée avec les autres variables

$$\begin{aligned}\forall x. \exists y. \varphi &\rightarrow [y/x] \psi \\&= \forall x. [f^{SK}(x, \vec{z})/y] \varphi \rightarrow [y/x] \psi \text{ avec } \vec{z} = VL(\forall x. \exists y. \varphi \rightarrow [y/x] \psi)\end{aligned}$$

# Sémantique

## Formes normales

- Forme normale prénexe : les règles précédentes permettent de remonter les quantificateurs en tête de formule

$$\begin{aligned}(\forall x. \varphi) &\rightarrow (\exists x. \psi) \\&= (\forall x. \varphi) \rightarrow (\exists y. [y/x] \psi) \\&= \forall x. \varphi \rightarrow (\exists y. [y/x] \psi) \\&= \forall x. \exists y. \varphi \rightarrow [y/x] \psi\end{aligned}$$

- Transformation de Skolem :

- Soit  $\vec{y} = VL(\forall x_1 \dots \forall x_n. \exists z. \varphi)$
- Soit  $f_n^{SK}$  un nouveau symbole fonctionnel d'arité  $n = |\vec{x}| + |\vec{y}|$

$$\forall x_1 \dots \forall x_n. \exists z. \varphi = \forall x_1 \dots \forall x_n. [f_n^{SK}(\vec{x}, \vec{y})/z] \varphi \text{ avec } \vec{x} = \langle x_1, \dots, x_n \rangle$$

- La transformation de Skolem permet de remplacer les quantificateurs existentiels dans une forme normale prénexe par des termes exprimant la dépendance de la variable quantifiée avec les autres variables

$$\begin{aligned}\forall x. \exists y. \varphi &\rightarrow [y/x] \psi \\&= \forall x. [f^{SK}(x, \vec{z})/y] \varphi \rightarrow [y/x] \psi \text{ avec } \vec{z} = VL(\forall x. \exists y. \varphi \rightarrow [y/x] \psi) \\&= \forall x. \varphi \rightarrow [f^{SK}(x, \vec{z})/x] \psi\end{aligned}$$

# Déduction naturelle

## Logique constructive

### ► Quantificateur universel :

#### ► Introduction

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x. \varphi} \quad I_{\forall} \ (x \notin VL(\Gamma))$$

#### ► Élimination

$$\frac{\Gamma \vdash \forall x. \varphi}{\Gamma \vdash [t/x] \varphi} \quad E_{\forall}$$

# Déduction naturelle

## Logique constructive

### ► Quantificateur universel :

#### ► Introduction

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x. \varphi} I_{\forall} \quad (x \notin VL(\Gamma))$$

#### ► Élimination

$$\frac{\Gamma \vdash \forall x. \varphi}{\Gamma \vdash [t/x] \varphi} E_{\forall}$$

### ► Quantificateur existentiel :

#### ► Introduction

$$\frac{\Gamma \vdash [t/x] \varphi}{\Gamma \vdash \exists x. \varphi} I_{\exists}$$

#### ► Élimination par Skolemisation

$$\frac{\Gamma \vdash \exists x. \varphi}{\Gamma \vdash [f(\vec{x})/x] \varphi} E_{\exists}^{SK} \quad (\vec{x} = VL(\Gamma) \cup VL(\exists x. \varphi))$$

#### ► Élimination par Modus Ponens

$$\frac{\Gamma \vdash \exists x. \varphi \quad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi} E_{\exists}^{MP} \quad (x \notin VL(\Gamma) \cup VL(\psi))$$



# Déduction naturelle

## Exemple

- Montrer que  $\forall x. \varphi \rightarrow (\neg \exists x. \neg \varphi)$  est un théorème.

$$\frac{\frac{\frac{}{\exists x. \neg \varphi, \forall x. \varphi \vdash \forall x. \varphi} \text{Hyp}}{\forall x. \varphi, \exists x. \neg \varphi \vdash [f(\vec{x})/x] \varphi} E_{\forall}}{\frac{\frac{\frac{}{\forall x. \varphi, \exists x. \neg \varphi \vdash \exists x. \neg \varphi} \text{Hyp}}{\forall x. \varphi, \exists x. \neg \varphi \vdash [f(\vec{x})/x] \neg \varphi} E_{\exists}^{SK} \ (\vec{x} = VL(\forall x. \varphi, \exists x. \neg \varphi) \cup VL(\exists x. \neg \varphi))}{\frac{\frac{\frac{}{\forall x. \varphi, \exists x. \neg \varphi \vdash \perp} I_{\neg}}{\forall x. \varphi \vdash \neg \exists x. \neg \varphi} I_{\perp}}{\vdash \forall x. \varphi \rightarrow (\neg \exists x. \neg \varphi)} I_{\rightarrow}}$$

# Logique des prédicats

## Conclusion

La logique des prédicats **d'ordre supérieur** est :

- ▶ Suffisamment expressive pour les mathématiques et l'informatique
- ▶ Complète et consistante sémantiquement
- ▶ Pas d'axiomatique complète et consistante (théorème de Gödel d'incomplétude de l'arithmétique)
- ▶ Indécidable mécaniquement
- ▶ Etude de fragments complets, consistants, décidables, semi-décidables, ...

La logique des prédicats **du premier ordre** est :

- ▶ Complète, consistante et correcte axiomatiquement
- ▶ Semi-décidable mécaniquement par énumération des théorèmes
- ▶ **Mais** Typage par prédicats coûteux et Raisonnement par égalité complexe
- ▶ Introduction du typage et des équations : Spécifications algébriques

# Spécifications algébriques

## Typage des constantes et opérateurs

- ▶ Soit  $\mathcal{S}$  un ensemble dénombrable de symboles, les sortes utilisées pour distinguer les termes possédant les mêmes caractéristiques
- ▶ Les termes sont séparés selon leur sorte :  $\mathcal{T} = \bigcup_{s \in \mathcal{S}} \mathcal{T}_s$
- ▶ La stratification des termes également
- ▶ Les constantes également :  $\mathcal{C} = \bigcup_{s \in \mathcal{S}} \mathcal{C}_s$
- ▶ Les variables également :  $\mathcal{V} = \bigcup_{s \in \mathcal{S}} \mathcal{V}_s$
- ▶ L'arité des fonctions prend en compte la sorte des paramètres et du résultat

$$\forall n \in \mathbb{N}. \mathcal{F}_n = \bigcup_{s \in \mathcal{S}, \forall i \in [1, \dots, n]. s_i \in \mathcal{S}} \mathcal{F}_{(s_1 \times \dots \times s_n) \mapsto s}$$

- ▶ L'arité est donc étendue pour intégrer les sortes

# Structure de termes

## Vision ensembliste

- ▶ Soit  $\mathcal{S}$  (resp.  $\mathcal{C}$ , resp.  $\mathcal{F}$ ) un ensemble dénombrable de sortes (resp. constantes, resp. fonctions)
- ▶ L'ensemble des termes  $\mathcal{T}$  partitionné selon les sortes est le plus petit ensemble tel que :
  - ▶  $\forall s \in \mathcal{S}. \forall c \in \mathcal{C}_s. c \in \mathcal{T}_s$
  - ▶  $\forall s_1, \dots, s_n, s \in \mathcal{S}. \forall f \in \mathcal{F}_{s_1 \times \dots \times s_n \mapsto s}. \forall t_1 \in \mathcal{T}_{s_1} \dots \forall t_n \in \mathcal{T}_{s_n}. f(t_1, \dots, t_n) \in \mathcal{T}_s$

# Structure de termes

## Vision ensembliste

- ▶ Soit  $\mathcal{S}$  (resp.  $\mathcal{C}$ , resp.  $\mathcal{F}$ ) un ensemble dénombrable de sortes (resp. constantes, resp. fonctions)
- ▶ L'ensemble des termes  $\mathcal{T}$  partitionné selon les sortes est le plus petit ensemble tel que :
  - ▶  $\forall s \in \mathcal{S}. \forall c \in \mathcal{C}_s. c \in \mathcal{T}_s$
  - ▶  $\forall s_1, \dots, s_n, s \in \mathcal{S}. \forall f \in \mathcal{F}_{s_1 \times \dots \times s_n \rightarrow s}. \forall t_1 \in \mathcal{T}_{s_1} \dots \forall t_n \in \mathcal{T}_{s_n}. f(t_1, \dots, t_n) \in \mathcal{T}_s$
- ▶ Exemple : Les entiers naturels de Peano

$$\begin{aligned} nat &\in \mathcal{S} \\ zero &\in \mathcal{C}_{nat} \\ successeur &\in \mathcal{F}_{nat \rightarrow nat} \end{aligned}$$

L'ensemble des termes est la plus petite solution de l'équation :

$$\mathcal{T}_{nat} = \{zero\} \cup \{successeur(n) \mid n \in \mathcal{T}_{nat}\}$$

- ▶ Ces définitions peuvent être stratifiées pour éliminer les paradoxes

# Structure de termes avec variable

## Vision ensembliste

- ▶ Soit  $\mathcal{V}$  un ensemble dénombrable de variables
- ▶ L'ensemble des termes  $\mathcal{T}[\mathcal{V}]$  avec variables partitionné selon les sortes est le plus petit ensemble tel que :
  - ▶  $\forall s \in \mathcal{S}. \forall c \in \mathcal{C}_s. c \in \mathcal{T}[\mathcal{V}]_s$
  - ▶  $\forall s \in \mathcal{S}. \forall x \in \mathcal{V}_s. x \in \mathcal{T}[\mathcal{V}]_s$
  - ▶  $\forall s_1, \dots, s_n, s \in \mathcal{S}. \forall f \in \mathcal{F}_{s_1 \times \dots \times s_n \rightarrow s}. \forall t_1 \in \mathcal{T}[\mathcal{V}]_{s_1}, \dots, t_n \in \mathcal{T}[\mathcal{V}]_{s_n}. f(t_1, \dots, t_n) \in \mathcal{T}[\mathcal{V}]_s$
- ▶ Notons qu'une substitution est une fonction de  $\mathcal{V}$  vers  $\mathcal{T}[\mathcal{V}]$  qui associe à une variable d'une sorte un terme de la même sorte
- ▶ Ces définitions peuvent être stratifiées pour éliminer les paradoxes

# Structure de termes

## Vision déductive

- La construction des termes de sorte  $s$  correspond aux règles d'introduction de  $s$  :

$$\frac{\displaystyle \frac{c \in \mathcal{C}_s}{c \in \mathcal{T}[\mathcal{V}]_s} I_s^c \quad \frac{x \in \mathcal{V}_s}{x \in \mathcal{T}[\mathcal{V}]_s} I_s^x}{\displaystyle \frac{n \in \mathbb{N}^* \quad f \in \mathcal{F}_{s_1 \times \dots \times s_n \mapsto s} \quad t_1 \in \mathcal{T}[\mathcal{V}]_{s_1} \quad t_n \in \mathcal{T}[\mathcal{V}]_{s_n}}{f(t_1, \dots, t_n) \in \mathcal{T}[\mathcal{V}]_s} I_s^f}$$

# Structure de termes

## Vision déductive

- La construction des termes de sorte  $s$  correspond aux règles d'introduction de  $s$  :

$$\frac{\begin{array}{c} \frac{c \in \mathcal{C}_s}{c \in \mathcal{T}[\mathcal{V}]_s} I_s^c \quad \frac{x \in \mathcal{V}_s}{x \in \mathcal{T}[\mathcal{V}]_s} I_s^x \\[10pt] n \in \mathbb{N}^* \quad \frac{f \in \mathcal{F}_{s_1 \times \dots \times s_n \rightarrow s} \quad t_1 \in \mathcal{T}[\mathcal{V}]_{s_1} \quad t_n \in \mathcal{T}[\mathcal{V}]_{s_n}}{f(t_1, \dots, t_n) \in \mathcal{T}[\mathcal{V}]_s} I_s^f \end{array}}{}$$

- Exemple : Les entiers naturels de Peano

$$\frac{}{zero \in \mathcal{T}[\mathcal{V}]_{nat}} I_{nat}^{zero}$$
$$\frac{n \in \mathcal{T}[\mathcal{V}]_{nat}}{successeur(n) \in \mathcal{T}[\mathcal{V}]_{nat}} I_{nat}^{successeur}$$



# Modélisation de l'arithmétique

## Formulation classique

- ▶ Entiers naturels (arithmétique de Peano) :  $\mathbb{N}$  modélisé par
  - ▶ *zero*  $\in \mathcal{C}_0$  ( $\bar{0} = \emptyset$ )
  - ▶ *successeur*  $\in \mathcal{F}_1$  ( $\overline{n+1} = \{\bar{n}\} \cup \bar{n}$ )

# Modélisation de l'arithmétique

## Formulation classique

- ▶ Entiers naturels (arithmétique de Peano) :  $\mathbb{N}$  modélisé par
  - ▶  $zero \in \mathcal{C}_0$  ( $\bar{0} = \emptyset$ )
  - ▶  $successeur \in \mathcal{F}_1$  ( $\overline{n+1} = \{\bar{n}\} \cup \bar{n}$ )
- ▶ Entiers relatifs :  $\mathbb{Z}$  modélisé par  $\mathbb{N}^2$  avec
  - ▶  $(n, 0)$  modélise  $\mathbb{Z}^+$  (représentant classe équivalence)
  - ▶  $(0, n)$  modélise  $\mathbb{Z}^-$  (représentant classe équivalence)

# Modélisation de l'arithmétique

## Formulation classique

- ▶ Entiers naturels (arithmétique de Peano) :  $\mathbb{N}$  modélisé par
  - ▶  $zero \in \mathcal{C}_0$  ( $\bar{0} = \emptyset$ )
  - ▶  $successeur \in \mathcal{F}_1$  ( $\overline{n+1} = \{\bar{n}\} \cup \bar{n}$ )
- ▶ Entiers relatifs :  $\mathbb{Z}$  modélisé par  $\mathbb{N}^2$  avec
  - ▶  $(n, 0)$  modélise  $\mathbb{Z}^+$  (représentant classe équivalence)
  - ▶  $(0, n)$  modélise  $\mathbb{Z}^-$  (représentant classe équivalence)
- ▶ Nombres rationnels :  $\mathbb{Q}$  modélisé par  $\mathbb{Z} \times \mathbb{N}^*$  avec  $(a, b) \in \mathbb{Z} \times \mathbb{N}^*$  tel que  $a \wedge b = 1$  modélise  $a/b \in \mathbb{Q}$  (représentant classe équivalence)

# Modélisation de l'arithmétique

## Formulation classique

- ▶ Entiers naturels (arithmétique de Peano) :  $\mathbb{N}$  modélisé par
  - ▶  $zero \in \mathcal{C}_0$  ( $\bar{0} = \emptyset$ )
  - ▶  $successeur \in \mathcal{F}_1$  ( $\overline{n+1} = \{\bar{n}\} \cup \bar{n}$ )
- ▶ Entiers relatifs :  $\mathbb{Z}$  modélisé par  $\mathbb{N}^2$  avec
  - ▶  $(n, 0)$  modélise  $\mathbb{Z}^+$  (représentant classe équivalence)
  - ▶  $(0, n)$  modélise  $\mathbb{Z}^-$  (représentant classe équivalence)
- ▶ Nombres rationnels :  $\mathbb{Q}$  modélisé par  $\mathbb{Z} \times \mathbb{N}^*$  avec  $(a, b) \in \mathbb{Z} \times \mathbb{N}^*$  tel que  $a \wedge b = 1$  modélise  $a/b \in \mathbb{Q}$  (représentant classe équivalence)
- ▶ Nombres réels (coupure de Dedekind) :  $\mathbb{R}$  modélisé par  $\mathcal{P}(\mathbb{Q})^2$  avec  $x \in \mathbb{R} \setminus \mathbb{Q}$  modélisé par la coupure  
( $\{q \in \mathbb{Q} \mid q < x\}, \{q \in \mathbb{Q} \mid x < q\}$ )  
( $\aleph_0 = |\mathbb{N}| = \omega, \aleph_1 = 2^{\aleph_0} = |\mathbb{R}|$ )

# Modélisation de l'arithmétique

## Formulation classique

- ▶ Entiers naturels (arithmétique de Peano) :  $\mathbb{N}$  modélisé par
  - ▶  $zero \in \mathcal{C}_0$  ( $\bar{0} = \emptyset$ )
  - ▶  $successeur \in \mathcal{F}_1$  ( $\overline{n+1} = \{\bar{n}\} \cup \bar{n}$ )
- ▶ Entiers relatifs :  $\mathbb{Z}$  modélisé par  $\mathbb{N}^2$  avec
  - ▶  $(n, 0)$  modélise  $\mathbb{Z}^+$  (représentant classe équivalence)
  - ▶  $(0, n)$  modélise  $\mathbb{Z}^-$  (représentant classe équivalence)
- ▶ Nombres rationnels :  $\mathbb{Q}$  modélisé par  $\mathbb{Z} \times \mathbb{N}^*$  avec  $(a, b) \in \mathbb{Z} \times \mathbb{N}^*$  tel que  $a \wedge b = 1$  modélise  $a/b \in \mathbb{Q}$  (représentant classe équivalence)
- ▶ Nombres réels (coupure de Dedekind) :  $\mathbb{R}$  modélisé par  $\mathcal{P}(\mathbb{Q})^2$  avec  $x \in \mathbb{R} \setminus \mathbb{Q}$  modélisé par la coupure  
( $\{q \in \mathbb{Q} \mid q < x\}, \{q \in \mathbb{Q} \mid x < q\}$ )  
( $\aleph_0 = |\mathbb{N}| = \omega, \aleph_1 = 2^{\aleph_0} = |\mathbb{R}|$ )
- ▶ Arithmétique décidable de Presburger : arithmétique linéaire (pas de multiplications entre variables)

# Equations sur les termes

## Propriétés algébriques

- ▶ L'égalité  $=$  est une relation (prédicat) d'équivalence définie dans la plupart des structures (appelées égalitaires) :
  - ▶ Réflexive :  $\forall x. x = x$
  - ▶ Symétrique :  $\forall x. \forall y. x = y \rightarrow y = x$
  - ▶ Transitive :  $\forall x. \forall y. \forall z. x = y \wedge y = z \rightarrow x = z$
- ▶ Mais, en logique des prédicats elle est traitée comme un prédicat quelconque dont les propriétés doivent être formalisées
- ▶ Les logiques équationnelles lui accordent un rôle particulier

# Equations sur les termes

## Propriétés algébriques

- ▶ L'égalité  $=$  est une relation (prédicat) d'équivalence définie dans la plupart des structures (appelées égalitaires) :
  - ▶ Réflexive :  $\forall x. x = x$
  - ▶ Symétrique :  $\forall x. \forall y. x = y \rightarrow y = x$
  - ▶ Transitive :  $\forall x. \forall y. \forall z. x = y \wedge y = z \rightarrow x = z$
- ▶ Mais, en logique des prédicats elle est traitée comme un prédicat quelconque dont les propriétés doivent être formalisées
- ▶ Les logiques équationnelles lui accordent un rôle particulier
- ▶ Une équation gardée sur la sorte  $s \in \mathcal{S}$  est de la forme :

$$\forall x_1 \in \mathcal{T}_{s_1}, \dots, \forall x_n \in \mathcal{T}_{s_n}. \varphi \rightarrow (G = D)$$

avec  $G \in \mathcal{T}[\mathcal{V}]_s$ ,  $D \in \mathcal{T}[\mathcal{V}]_s$ ,  $\varphi$  une formule bien formée de la logique des prédicats et  $\{x_1, \dots, x_n\} = VL(\varphi) \cup VL(G) \cup VL(D)$

- ▶ Elle est notée  $\varphi \rightarrow (G = D)$

# Equations sur les termes

## Exemple

- Définition de l'addition pour les entiers de Peano :

$$\begin{aligned} & \text{somme} \in \mathcal{F}_{\text{nat} \times \text{nat} \rightarrow \text{nat}} \\ & \forall m \in \mathcal{T}_{\text{nat}}. \text{somme}(\text{zero}, m) = m \\ & \left[ \begin{array}{l} \forall n \in \mathcal{T}_{\text{nat}}, \\ \forall m \in \mathcal{T}_{\text{nat}}, \\ \text{somme}(\text{successeur}(n), m) = \text{successeur}(\text{somme}(n, m)) \end{array} \right] \end{aligned}$$



# Equations sur les termes

## Exemple

- Définition de l'addition pour les entiers de Peano :

$$\begin{aligned} & \text{somme} \in \mathcal{F}_{\text{nat} \times \text{nat} \rightarrow \text{nat}} \\ & \forall m \in \mathcal{T}_{\text{nat}}. \text{somme}(\text{zero}, m) = m \\ & \left[ \begin{array}{l} \forall n \in \mathcal{T}_{\text{nat}}, \\ \forall m \in \mathcal{T}_{\text{nat}}, \\ \text{somme}(\text{successeur}(n), m) = \text{successeur}(\text{somme}(n, m)) \end{array} \right] \end{aligned}$$

- Calcul d'une addition par réécriture en utilisant les équations :

$$\begin{aligned} & \text{somme}(\text{successeur}(\text{successeur}(\text{zero})), \text{successeur}(\text{zero})) \\ &= \text{successeur}(\text{somme}(\text{successeur}(\text{zero}), \text{successeur}(\text{zero}))) \\ &= \text{successeur}(\text{successeur}(\text{somme}(\text{zero}, \text{successeur}(\text{zero})))) \\ &= \text{successeur}(\text{successeur}(\text{successeur}(\text{zero}))) \end{aligned}$$

# Spécification algébrique

## Sémantique

- ▶ Une spécification algébrique est définie par un ensemble dénombrable de sortes, un ensemble dénombrable de constantes, un ensemble dénombrables de fonctions et un ensemble dénombrables d'équations
- ▶ La sémantique d'une spécification algébrique est donnée par une interprétation comme pour la logique des prédicats
- ▶ L'interprétation d'une spécification algébrique doit valider toutes les équations
- ▶ Une interprétation particulière appelée sémantique initiale s'appuie sur la structure d'algèbre de termes partitionnée en classes d'équivalence selon les équations
- ▶ Les formules de logique équationnelle sont des formules de logique des prédicats du premier ordre exploitant l'opérateur  $=$  sur les termes, c'est-à-dire contenant des équations

# Déduction naturelle

## Logique équationnelle

$$\frac{}{\Gamma \vdash t = t} \text{ Réflexivité} \qquad \frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash t_2 = t_1} \text{ Symétrie}$$

$$\frac{\Gamma \vdash t_1 = t_2 \quad \Gamma \vdash t_2 = t_3}{\Gamma \vdash t_1 = t_3} \text{ Transitivité}$$

$$\frac{f \in \mathcal{F}_{s_1 \times \dots \times s_n \mapsto s} \quad \Gamma \vdash t_1 = t'_1 \quad \Gamma \vdash t_n = t'_n}{\Gamma \vdash f(t_1, \dots, t_n) = f(t'_1, \dots, t'_n)} \text{ Congruence}$$

$$\frac{\Gamma \vdash t_1 = t_2}{\Gamma \vdash \sigma t_1 = \sigma t_2} \text{ Substitution} \qquad \frac{\Gamma \vdash t_1 = t_2 \quad \Gamma \vdash [t_2/t_1]\varphi}{\Gamma \vdash \varphi} \text{ Réécriture}$$

- ▶ En déduction naturelle “pure”, seules Réflexivité et Réécriture sont nécessaires (resp. introduction et élimination de l'égalité)
- ▶ La substitution  $[t_2/t_1]\varphi$  est une généralisation de la substitution d'une variable au remplacement d'un terme par un autre terme. Les contraintes sur les variables libres sont similaires.

# Preuve par induction

## Formulation classique

### ► Rappel: Récurrence simple

$$(\forall n \in \mathbb{N}. \varphi) \leftrightarrow \begin{cases} [0/n] \varphi \\ \wedge \forall m \in \mathbb{N}. ([m/n] \varphi \rightarrow [m+1/n] \varphi) \end{cases}$$

# Preuve par induction

## Formulation classique

- Rappel: Récurrence simple

$$(\forall n \in \mathbb{N}. \varphi) \leftrightarrow \begin{cases} [0/n] \varphi \\ \wedge \forall m \in \mathbb{N}. ([m/n] \varphi \rightarrow [m+1/n] \varphi) \end{cases}$$

- Rappel: Récurrence généralisée

$$(\forall n \in \mathbb{N}. \varphi) \leftrightarrow \begin{cases} [0/n] \varphi \\ \wedge \forall p \in \mathbb{N}. ((\forall q \in \mathbb{N}. q < p \rightarrow [q/n] \varphi) \rightarrow [p/n] \varphi) \end{cases}$$

# Preuve par induction

## Formulation classique

- Rappel: Récurrence simple

$$(\forall n \in \mathbb{N}. \varphi) \leftrightarrow \begin{cases} [0/n] \varphi \\ \wedge \forall m \in \mathbb{N}. ([m/n] \varphi \rightarrow [m+1/n] \varphi) \end{cases}$$

- Rappel: Récurrence généralisée

$$(\forall n \in \mathbb{N}. \varphi) \leftrightarrow \begin{cases} [0/n] \varphi \\ \wedge \forall p \in \mathbb{N}. ((\forall q \in \mathbb{N}. q < p \rightarrow [q/n] \varphi) \rightarrow [p/n] \varphi) \end{cases}$$

- Induction bien fondée: la relation d'ordre strict  $< : \mathcal{E} \times \mathcal{E}$  est bien fondée s'il n'existe pas de chaîne de  $\mathcal{E}$  infiniment décroissante

$$(\forall x \in \mathcal{E}. \varphi) \leftrightarrow \forall y \in \mathcal{E}. ((\forall z \in \mathcal{E}. z < y \rightarrow [z/x] \varphi) \rightarrow [y/x] \varphi)$$

- Exemple :  $<$  est bien fondée sur  $\mathbb{N}$ . Une telle relation sur les termes peut être définie par plongement sur  $\mathbb{N}$ .

# Preuve par induction

## Formulation déductive

- Récurrence et induction sont des règles d'élimination
- Rappel : Récurrence simple

$$\frac{\Gamma \vdash n \in \mathbb{N} \quad \Gamma \vdash [0/n] \varphi \quad \Gamma, m \in \mathbb{N}, [m/n] \varphi \vdash [m+1/n] \varphi}{\Gamma \vdash \varphi} E_{nat}^{RS}$$

# Preuve par induction

## Formulation déductive

- Récurrence et induction sont des règles d'élimination

- Rappel : Récurrence simple

$$\frac{\Gamma \vdash n \in \mathbb{N} \quad \Gamma \vdash [0/n] \varphi \quad \Gamma, m \in \mathbb{N}, [m/n] \varphi \vdash [m+1/n] \varphi}{\Gamma \vdash \varphi} E_{nat}^{RS}$$

- Rappel : Récurrence généralisée

$$\frac{\Gamma \vdash n \in \mathbb{N} \quad \Gamma \vdash [0/n] \varphi \quad \Gamma, p \in \mathbb{N}, (\forall q \in \mathbb{N}. q < p \rightarrow [q/n] \varphi) \vdash [p/n] \varphi}{\Gamma \vdash \varphi} E_{nat}^{RG}$$



# Preuve par induction

## Formulation déductive

- Récurrence et induction sont des règles d'élimination

- Rappel : Récurrence simple

$$\frac{\Gamma \vdash n \in \mathbb{N} \quad \Gamma \vdash [0/n] \varphi \quad \Gamma, m \in \mathbb{N}, [m/n] \varphi \vdash [m+1/n] \varphi}{\Gamma \vdash \varphi} E_{nat}^{RS}$$

- Rappel : Récurrence généralisée

$$\frac{\Gamma \vdash n \in \mathbb{N} \quad \Gamma \vdash [0/n] \varphi \quad \Gamma, p \in \mathbb{N}, (\forall q \in \mathbb{N}. q < p \rightarrow [q/n] \varphi) \vdash [p/n] \varphi}{\Gamma \vdash \varphi} E_{nat}^{RG}$$

- Induction bien fondée

$$\frac{\Gamma \vdash x \in \mathcal{E} \quad \Gamma, y \in \mathcal{E}, (\forall z \in \mathcal{E}. z < y \rightarrow [z/x] \varphi) \vdash [y/x] \varphi}{\Gamma \vdash \varphi} E_{nat}^{BF}$$

# Preuve par induction

## Formulation classique

- Induction structurelle:

$$\begin{array}{c} \forall s \in \mathcal{S}. \\ (\forall t \in \mathcal{T}_s. \varphi) \\ \Leftrightarrow \\ \left( \begin{array}{c} \forall c \in \mathcal{C}_s. [c/t] \varphi \\ \wedge \\ \forall f \in \mathcal{F}_{s_1 \times \dots \times s_n \mapsto s}. \forall t_1 \in \mathcal{T}_{s_1}, \dots, \forall t_n \in \mathcal{T}_{s_n}. \\ \left( \bigwedge_{j \in [1 \dots n]} (s_j = s \rightarrow [t_j/t] \varphi) \right) \rightarrow [f(t_1, \dots, t_n)/t] \varphi \end{array} \right) \end{array}$$

- Remarque : Hypothèse d'induction uniquement lorsque  $s_i = s$

# Preuve par induction

## Formulation classique

- Induction structurelle:

$$\begin{array}{c} \forall s \in \mathcal{S}. \\ (\forall t \in \mathcal{T}_s. \varphi) \\ \Leftrightarrow \\ \forall c \in \mathcal{C}_s. [c/t] \varphi \\ \wedge \\ \forall f \in \mathcal{F}_{s_1 \times \dots \times s_n \mapsto s}. \forall t_1 \in \mathcal{T}_{s_1}, \dots, \forall t_n \in \mathcal{T}_{s_n}. \\ \left( \bigwedge_{j \in [1..n]} (s_j = s \rightarrow [t_j/t] \varphi) \right) \rightarrow [f(t_1, \dots, t_n)/t] \varphi \end{array}$$

- Remarque : Hypothèse d'induction uniquement lorsque  $s_i = s$
- Exemple : Entier naturel de Peano

$$\begin{array}{c} (\forall t \in \mathcal{T}_{nat}. \varphi) \\ \Leftrightarrow \\ ([zero/t] \varphi \wedge \forall p \in \mathcal{T}_{nat}. [p/t] \varphi \rightarrow [successeur(p)/t] \varphi) \end{array}$$

- Identique dans ce cas à la récurrence simple

# Preuve par induction structurelle

## Exemple

- ▶ Exemple :  $\forall n_1 \in T_{nat}. \forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(n_1, n_2)$

# Preuve par induction structurelle

## Exemple

- ▶ Exemple :  $\forall n_1 \in T_{nat}. \forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(n_1, n_2)$
- ▶ Induction sur  $n_1$  :
  - ▶  $n_1 = \text{zero}$  : Prouvons  $\forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(\text{zero}, n_2)$
  - ▶  $n_1 = \text{successeur}(n_3)$  avec  $n_3 \in T_{nat}$  et  $\forall n_2 \in T_{nat}. \exists r_2 \in T_{nat}. r_2 = \text{somme}(n_3, n_2)$  :  
Prouvons  $\forall n_2 \in T_{nat}. \exists r_1 \in T_{nat}. r_1 = \text{somme}(\text{successeur}(n_3), n_2)$

# Preuve par induction structurelle

## Exemple

- ▶ Exemple :  $\forall n_1 \in T_{nat}. \forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(n_1, n_2)$
- ▶ Induction sur  $n_1$  :
  - ▶  $n_1 = \text{zero}$  : Prouvons  $\forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(\text{zero}, n_2)$   
Nous avons  $\text{somme}(\text{zero}, n_2) = n_2$   
Prenons donc  $r = n_2$
  - ▶  $n_1 = \text{successeur}(n_3)$  avec  $n_3 \in T_{nat}$  et  
 $\forall n_2 \in T_{nat}. \exists r_2 \in T_{nat}. r_2 = \text{somme}(n_3, n_2)$  :  
Prouvons  $\forall n_2 \in T_{nat}. \exists r_1 \in T_{nat}. r_1 = \text{somme}(\text{successeur}(n_3), n_2)$

# Preuve par induction structurelle

## Exemple

- ▶ Exemple :  $\forall n_1 \in T_{nat}. \forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(n_1, n_2)$
- ▶ Induction sur  $n_1$  :
  - ▶  $n_1 = \text{zero}$  : Prouvons  $\forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(\text{zero}, n_2)$   
Nous avons  $\text{somme}(\text{zero}, n_2) = n_2$   
Prenons donc  $r = n_2$
  - ▶  $n_1 = \text{successeur}(n_3)$  avec  $n_3 \in T_{nat}$  et  $\forall n_2 \in T_{nat}. \exists r_2 \in T_{nat}. r_2 = \text{somme}(n_3, n_2)$  :  
Prouvons  $\forall n_2 \in T_{nat}. \exists r_1 \in T_{nat}. r_1 = \text{somme}(\text{successeur}(n_3), n_2)$   
Nous avons  $\text{somme}(\text{successeur}(n_3), n_2) = \text{successeur}(\text{somme}(n_3, n_2)) = \text{successeur}(r_2)$   
Prenons donc  $r_1 = \text{successeur}(r_2)$

# Preuve par induction

## Formulation déductive

- ▶ Induction structurelle:
- ▶ Prenons  $\{c_1, \dots, c_p\} = \mathcal{C}_s$
- ▶ Prenons  $\{f_1, \dots, f_q\} = \bigcup_{\substack{n \in \mathbb{N} \\ s_1, \dots, s_n \in \mathcal{S}}} \mathcal{F}_{s_1 \times \dots \times s_n \mapsto s}$
- ▶ La règle ci-dessous possède  $p + q$  prémisses  $c_i \in \mathcal{C}_s$  et  $f_j \in \mathcal{F}_{s_1 \times \dots \times s_{n_j} \mapsto s}$

$$\frac{\begin{array}{l} \Gamma \vdash [c_1/t] \varphi \qquad \Gamma, t_i \in \mathcal{T}_{s_i}, (s_i = s \rightarrow [t_i/t] \varphi) \vdash [f_{n,1}(t_1, \dots, t_n)/t] \varphi \\ \vdots \qquad \qquad \qquad \vdots \\ \Gamma \vdash [c_p/t] \varphi \qquad \Gamma, t_i \in \mathcal{T}_{s_i}, (s_i = s \rightarrow [t_i/t] \varphi) \vdash [f_{n,q}(t_1, \dots, t_n)/t] \varphi \end{array}}{\Gamma \vdash \varphi} E_s$$

- ▶ Remarque : Hypothèse d'induction uniquement lorsque  $s_k = s$



# Preuve par induction

## Formulation déductive

- ▶ Induction structurelle:
- ▶ Prenons  $\{c_1, \dots, c_p\} = \mathcal{C}_s$
- ▶ Prenons  $\{f_1, \dots, f_q\} = \bigcup_{\substack{n \in \mathbb{N} \\ s_1, \dots, s_n \in \mathcal{S}}} \mathcal{F}_{s_1 \times \dots \times s_n \mapsto s}$
- ▶ La règle ci-dessous possède  $p + q$  prémisses  $c_i \in \mathcal{C}_s$  et  $f_j \in \mathcal{F}_{s_1 \times \dots \times s_{n_j} \mapsto s}$

$$\frac{\begin{array}{c} \Gamma \vdash [c_1/t] \varphi \qquad \Gamma, t_i \in \mathcal{T}_{s_i}, (s_i = s \rightarrow [t_i/t] \varphi) \vdash [f_{n,1}(t_1, \dots, t_n)/t] \varphi \\ \vdots \qquad \qquad \qquad \vdots \\ \Gamma \vdash [c_p/t] \varphi \qquad \Gamma, t_i \in \mathcal{T}_{s_i}, (s_i = s \rightarrow [t_i/t] \varphi) \vdash [f_{n,q}(t_1, \dots, t_n)/t] \varphi \end{array}}{\Gamma \vdash \varphi} E_s$$

- ▶ Remarque : Hypothèse d'induction uniquement lorsque  $s_k = s$
- ▶ Exemple : Entier naturel de Peano

$$\frac{\Gamma \vdash t \in \mathcal{T}_{nat} \quad \Gamma \vdash [zero/t] \varphi \quad \Gamma, p \in \mathcal{T}_{nat}, [p/t] \varphi \vdash [successeur(p)/t] \varphi}{\Gamma \vdash \varphi} E_{nat}$$

- ▶ Identique à la récurrence simple

# Induction structurelle et D  duction naturelle

## Exemple

► Exemple :  $\forall n_1 \in T_{nat}. \forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(n_1, n_2)$

$$\frac{}{n_1 \in T_{nat} \vdash n_1 \in T_{nat}} \text{Hyp}$$

$$\vdots$$

$$\frac{}{n_1 \in T_{nat} \vdash \forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(\text{zero}, n_2)}$$

$$\vdots$$

$$n_1 \in T_{nat},$$

$$n_3 \in T_{nat}$$

$$\left[ \begin{array}{l} \forall n_2 \in T_{nat}, \\ \exists r_2 \in T_{nat}, \\ r_2 = \text{somme}(n_3, n_2) \end{array} \right]$$

$$\vdash \left[ \begin{array}{l} \forall n_2 \in T_{nat}, \\ \exists r \in T_{nat}, \\ r = \text{somme}(\text{successeur}(n_3), n_2) \end{array} \right]$$

$$\left[ \begin{array}{l} \forall n_2 \in T_{nat}, \\ \exists r_2 \in T_{nat}, \\ r_2 = \text{somme}(n_3, n_2) \end{array} \right] \vdash \left[ \begin{array}{l} \forall n_2 \in T_{nat}, \\ \exists r \in T_{nat}, \\ r = \text{somme}(\text{successeur}(n_3), n_2) \end{array} \right]$$

$$E_{nat}$$

$$\frac{}{n_1 \in T_{nat} \vdash \forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(n_1, n_2)}$$

$$\frac{}{\vdash \forall n_1 \in T_{nat}. \forall n_2 \in T_{nat}. \exists r \in T_{nat}. r = \text{somme}(n_1, n_2)} \text{I}_{\forall}$$

# Preuves de programmes fonctionnels

- ▶ Un programme fonctionnel pur est semblable à une fonction mathématique
- ▶ Pas de modification de l'état de la mémoire (effet de bord)
- ▶ Chaque appel avec les mêmes valeurs de paramètres calcule le même résultat
- ▶ Langages appropriés : Haskell, Coq, Isabelle, ... ; restrictions d'OCaML, de  $F^\sharp$ , de Lisp, de Scheme, de Clojure, ...
- ▶ La spécification du programme est une propriété des valeurs du résultat, en fonction des valeurs des paramètres
- ▶ Les données typées sont représentées par des termes
- ▶ Les programmes sont des ensembles d'équations sur les termes

# Preuves de programmes impératifs

Spécification par les triplets de Hoare

- ▶ Un programme impératif modifie l'état de la mémoire à chaque étape de son exécution
  - ▶ Semblable à une fonction mathématique dont le paramètre est l'état initial de la mémoire et le résultat est l'état final de la mémoire
  - ▶ Chaque étape de calcul est une fonction de l'état précédent de la mémoire vers l'état suivant de la mémoire

# Preuves de programmes impératifs

## Spécification par les triplets de Hoare

- ▶ Un programme impératif modifie l'état de la mémoire à chaque étape de son exécution
  - ▶ Semblable à une fonction mathématique dont le paramètre est l'état initial de la mémoire et le résultat est l'état final de la mémoire
  - ▶ Chaque étape de calcul est une fonction de l'état précédent de la mémoire vers l'état suivant de la mémoire
- ▶ Spécification d'un programme impératif
  - ▶ Post-condition  $\psi$  : Propriétés attendues sur l'état de la mémoire à la fin de l'exécution
  - ▶ Pré-condition  $\varphi$  : En fonction des propriétés de l'état de la mémoire au début de l'exécution
  - ▶ Notation : Triplet de Hoare  $\{\varphi\} P \{\psi\}$

# Exemple de spécification formelle

Élévation au carré efficace

Syntaxe similaire à Ada mais plus compacte.

```
{0 ≤ N}  
x := 0;  
y := 0;  
while x <> N do  
    y := y + 2 * x + 1;  
    x := x + 1  
od  
{y = N2}
```

# Preuves de programmes impératifs

## Preuve de correction

- ▶ Chaque étape intermédiaire est annotée par une propriété de l'état de la mémoire
- ▶ Chaque instruction  $I$  est :
  - ▶ précédée d'une pré-condition  $\varphi$  (propriété de l'état de la mémoire avant l'exécution de l'instruction)
  - ▶ suivie d'une post-condition  $\psi$  (propriété de l'état de la mémoire après l'exécution de l'instruction)

# Preuves de programmes impératifs

## Preuve de correction

- ▶ Chaque étape intermédiaire est annotée par une propriété de l'état de la mémoire
- ▶ Chaque instruction  $I$  est :
  - ▶ précédée d'une pré-condition  $\varphi$  (propriété de l'état de la mémoire avant l'exécution de l'instruction)
  - ▶ suivie d'une post-condition  $\psi$  (propriété de l'état de la mémoire après l'exécution de l'instruction)
- ▶ Chaque instruction annotée doit satisfaire les règles de la logique de Hoare  $\{\varphi\} / \{\psi\}$ 
  - ▶ Correction partielle : **Si** la pré-condition  $\varphi$  est satisfaite avant l'exécution de l'instruction  $I$  **et** l'exécution se termine **alors** la post-condition  $\psi$  est satisfaite après l'exécution
  - ▶ Correction totale : **Si** la pré-condition  $\varphi$  est satisfaite avant l'exécution de l'instruction  $I$  **alors** l'exécution se termine **et** la post-condition  $\psi$  est satisfaite après l'exécution
- ▶ Les propriétés de l'état de la mémoire sont représentées par de la logique équationnelle (logique du premier ordre et spécifications algébriques)



# Exemple de preuve de correction partielle

Élévation au carré efficace

Utilisation d'un invariant  $y = x^2$

```
{0 ≤ N}  
{0 = 02}  
x := 0;  
{0 = x2}  
y := 0;  
{y = x2}  
while x <> N invariant y = x2 do  
  {y = x2 ∧ x <> N}  
  {y + 2 × x + 1 = (x + 1)2}  
  y := y + 2 * x + 1;  
  {y = (x + 1)2}  
  x := x + 1  
  {y = x2}  
od  
{y = x2 ∧ ¬(x <> N)}  
{y = N2}
```

# Logique de Floyd/Hoare

## Règles de déduction

$$\frac{}{\{\psi\} \text{ skip } \{\psi\}} \text{ skip}$$

$$\frac{}{\{[E/x]\psi\} x := E \{\psi\}} \text{ assign}$$

$$\frac{\{\varphi\} P \{\chi\} \quad \{\chi\} Q \{\psi\}}{\{\varphi\} P ; Q \{\psi\}} \text{ sequence}$$

$$\frac{\{\varphi \wedge C\} P \{\psi\} \quad \{\varphi \wedge \neg C\} Q \{\psi\}}{\{\varphi\} \text{ if } C \text{ then } P \text{ else } Q \text{ fi } \{\psi\}} \text{ conditional}$$

$$\frac{\{\varphi \wedge C\} P \{\varphi\}}{\{\varphi\} \text{ while } C \text{ invariant } \varphi \text{ do } P \text{ od } \{\varphi \wedge \neg C\}} \text{ partial loop}$$

$$\frac{\{\varphi \wedge C \wedge E \in \mathbb{N} \wedge V = E\} P \{\varphi \wedge E \in \mathbb{N} \wedge V > E\}}{\{\varphi \wedge E \in \mathbb{N}\} \text{ while } C \text{ invariant } \varphi \text{ variant } E \text{ do } P \text{ od } \{\varphi \wedge \neg C\}} \text{ total loop}$$

$$\frac{\varphi \rightarrow \chi \quad \{\chi\} P \{\psi\}}{\{\varphi\} P \{\psi\}} \text{ weaken}$$

$$\frac{\{\varphi\} P \{\chi\} \quad \chi \rightarrow \psi}{\{\varphi\} P \{\psi\}} \text{ strengthen}$$

# Exemple de preuve de correction totale

Élévation au carré efficace

Utilisation d'un variant :  $N - x$

```
{0 ≤ N}
{... ∧ (N - 0) ∈ ℕ}
x := 0;
{... ∧ (N - x) ∈ ℕ}
y := 0;
{... ∧ (N - x) ∈ ℕ}
while x <> N invariant y = x2 variant N - x do
    {... ∧ x <> N ∧ (N - x) ∈ ℕ ∧ V = N - x}
    {... ∧ (N - (x + 1)) ∈ ℕ ∧ N - (x + 1) < V}
    y := y + 2 * x + 1;
    {... ∧ (N - (x + 1)) ∈ ℕ ∧ N - (x + 1) < V}
    x := x + 1
    {... ∧ (N - x) ∈ ℕ ∧ N - x < V}
od
{...}
{...}
```

# Logique de Floyd/Hoare

## Règles de déduction

- ▶ Inaction : skip
  - ▶ Le contenu de la mémoire n'est pas modifié durant l'exécution
  - ▶ La pré-condition et la post-condition sont identiques

$$\frac{}{\{\psi\} \text{ skip } \{\psi\}} \text{ skip}$$

# Logique de Floyd/Hoare

## Règles de déduction

### ► Inaction : skip

- Le contenu de la mémoire n'est pas modifié durant l'exécution
- La pré-condition et la post-condition sont identiques

$$\frac{}{\{\psi\} \text{ skip } \{\psi\}} \text{ skip}$$

### ► Renforcement de la pré-condition (hypothèse plus forte)

$$\frac{\varphi \rightarrow \chi \quad \{\chi\} P \{\psi\}}{\{\varphi\} P \{\psi\}} \text{ weaken}$$

### ► Affaiblissement de la post-condition (conclusion plus faible)

$$\frac{\{\varphi\} P \{\chi\} \quad \chi \rightarrow \psi}{\{\varphi\} P \{\psi\}} \text{ strengthen}$$

# Logique de Floyd/Hoare

## Affectation et Plus faible pré-condition

Affectation :  $x := E$

- ▶ Seul le contenu de  $x$  est modifié
  - ▶ Mémoire avant :  $\langle x_1 \mapsto v_1, \dots, x_n \mapsto v_n \rangle$
  - ▶ Mémoire après :  $\langle x_1 \mapsto v'_1, \dots, x_n \mapsto v'_n \rangle$
  - ▶ Si  $x = x_i$  alors :
    - ▶  $v'_i = [v_1/x_1, \dots, v_n/x_n] E$
    - ▶ et  $\forall j \in [1, \dots, n]. (j \neq i) \rightarrow v'_j = v_j$
  - ▶ Donc :  $[v_1/x_1, \dots, v_n/x_n] [E/x] \psi = [v'_1/x_1, \dots, v'_n/x_n] \psi$

$$\frac{}{\{[E/x] \psi\} x := E \{\psi\}} \text{ assign}$$

- ▶ Il s'agit de calculer la plus faible pré-condition nécessaire pour établir la post-condition  $\psi$
- ▶ Travaux de Dijkstra
- ▶ Base des outils de mécanisation comme Why3

# Logique de Floyd/Hoare

## Séquence et Conditionnelle

- ▶ Séquence :  $P ; Q$

- ▶ L'état de la mémoire entre l'exécution de  $P$  et de  $Q$  est caractérisé par la propriété  $\chi$

$$\frac{\{\varphi\} P \{\chi\} \quad \{\chi\} Q \{\psi\}}{\{\varphi\} P ; Q \{\psi\}} \text{ sequence}$$

# Logique de Floyd/Hoare

## Séquence et Conditionnelle

### ► Séquence : $P ; Q$

- L'état de la mémoire entre l'exécution de  $P$  et de  $Q$  est caractérisé par la propriété  $\chi$

$$\frac{\{\varphi\} P \{\chi\} \quad \{\chi\} Q \{\psi\}}{\{\varphi\} P ; Q \{\psi\}} \text{ sequence}$$

- Calcul de la plus faible pré-condition

### ► Conditionnelle : $\text{if } C \text{ then } P \text{ else } Q \text{ fi}$

- L'état de la mémoire après l'exécution des branches  $P$  et de  $Q$  doit satisfaire la propriété  $\psi$
- L'état de la mémoire avant l'exécution des branches  $P$  et de  $Q$  est distingué par la valeur de la condition  $C$

$$\frac{\{\varphi \wedge C\} P \{\psi\} \quad \{\varphi \wedge \neg C\} Q \{\psi\}}{\{\varphi\} \text{if } C \text{ then } P \text{ else } Q \text{ fi } \{\psi\}} \text{ conditional}$$

- Calcul de la plus faible pré-condition



# Logique de Floyd/Hoare

## Boucle, Correction partielle

- ▶ Boucle : `while C invariant  $\varphi$  variant E do P od`
- ▶ Dépliage boucle :
  - ▶ Notons  $R = \text{while } C \text{ invariant } \varphi \text{ do } P \text{ od}$
  - ▶ Alors  $R$  satisfait  $R = \text{if } C \text{ then } P ; R \text{ else skip fi}$
  - ▶ Plus faible pré-condition est une disjonction infinie des séquences  $P^{(i)}$  issues des dépliages de  $R$
  - ▶ Besoin d'une propriété finie plus forte : l'invariant de boucle  $\varphi$

# Logique de Floyd/Hoare

## Boucle, Correction partielle

- ▶ Boucle : `while C invariant  $\varphi$  variant E do P od`
- ▶ Dépliage boucle :
  - ▶ Notons  $R = \text{while } C \text{ invariant } \varphi \text{ do } P \text{ od}$
  - ▶ Alors  $R$  satisfait  $R = \text{if } C \text{ then } P ; R \text{ else skip fi}$
  - ▶ Plus faible pré-condition est une disjonction infinie des séquences  $P^{(i)}$  issues des dépliages de  $R$
  - ▶ Besoin d'une propriété finie plus forte : l'invariant de boucle  $\varphi$
- ▶ Correction partielle :

$$\frac{\{\varphi \wedge C\} P \{\varphi\}}{\{\varphi\} \text{ while } C \text{ invariant } \varphi \text{ do } P \text{ od } \{\varphi \wedge \neg C\}} \quad \text{partial loop}$$

# Logique de Floyd/Hoare

## Boucle, Correction partielle

- ▶ Boucle : `while C invariant  $\varphi$  variant E do P od`
- ▶ Dépliage boucle :
  - ▶ Notons  $R = \text{while } C \text{ invariant } \varphi \text{ do } P \text{ od}$
  - ▶ Alors  $R$  satisfait  $R = \text{if } C \text{ then } P ; R \text{ else skip fi}$
  - ▶ Plus faible pré-condition est une disjonction infinie des séquences  $P^{(i)}$  issues des dépliages de  $R$
  - ▶ Besoin d'une propriété finie plus forte : l'invariant de boucle  $\varphi$

- ▶ Correction partielle :

$$\frac{\{\varphi \wedge C\} P \{\varphi\}}{\{\varphi\} \text{ while } C \text{ invariant } \varphi \text{ do } P \text{ od } \{\varphi \wedge \neg C\}} \text{ partial loop}$$

- ▶ Principe construction :

$$\frac{\frac{\{\varphi \wedge C\} P \{\varphi\} \quad \{\varphi\} R \{\varphi \wedge \neg C\}}{\{\varphi \wedge C\} P ; R \{\varphi \wedge \neg C\}} \text{ sequence} \quad \frac{}{\{\varphi \wedge \neg C\} \text{ skip } \{\varphi \wedge \neg C\}} \text{ skip}}{\{\varphi\} \text{ if } C \text{ then } P ; R \text{ else skip fi } \{\varphi \wedge \neg C\}} \text{ conditional}$$
$$\frac{\{\varphi\} \text{ if } C \text{ then } P ; R \text{ else skip fi } \{\varphi \wedge \neg C\}}{\{\varphi\} \text{ while } C \text{ invariant } \varphi \text{ do } P \text{ od } \{\varphi \wedge \neg C\}} \text{ unfold loop}$$

# Logique de Floyd/Hoare

## Boucle, Correction totale

- ▶ Boucle : while  $C$  invariant  $\varphi$  variant  $E$  do  $P$  od
- ▶ Correction totale : Combiner la correction partielle et une preuve de terminaison
- ▶ Induction bien fondée sur l'état de la mémoire
- ▶ L'état de la mémoire doit décroître strictement à chaque exécution du corps de la boucle
  - ▶ Variant :  $E \in \mathbb{N}$
  - ▶  $[v'_1/x_1, \dots, v'_n/x_n] E < [v_1/x_1, \dots, v_n/x_n] E$

- ▶ Correction totale :

$$\frac{\{\varphi \wedge C \wedge E \in \mathbb{N} \wedge V = E\} P \{\varphi \wedge E \in \mathbb{N} \wedge V > E\}}{\{\varphi \wedge E \in \mathbb{N}\} \text{ while } C \text{ invariant } \varphi \text{ variant } E \text{ do } P \text{ od } \{\varphi \wedge \neg C\}} \quad \text{total loop}$$

- ▶ Principe de construction identique à la correction partielle (variable  $V$  contient valeur du variant avant exécution du corps de la boucle)
- ▶ Il peut être nécessaire d'enrichir l'invariant avec  $C \rightarrow E \in \mathbb{N}$

# Logique de Floyd/Hoare

## Méthode de preuve

1. Recherche des candidats invariants et variants pour chaque boucle
2. Annotations des boucles
3. Remontée des plus faibles pré-conditions le long des séquences, affectations et branches de conditionnelle
4. Construction des obligations de preuve issues des affaiblissements et renforcements dans les boucles et les branches de conditionnelle
5. En cas d'échec dans la preuve des obligations, renforcement des invariants avec les propriétés manquantes pour réaliser ces preuves
6. Reprise à l'étape 2

# Exemple d'application de la méthode

Élévation au carré efficace

```
{0 ≤ N}  
x := 0;  
y := 0;  
while x <> N do  
    y := y + 2 * x + 1;  
    x := x + 1  
od  
{y = N2}
```

Étapes de boucle	x	y		

# Exemple d'application de la méthode

## Élévation au carré efficace

```
{0 ≤ N}  
x := 0;  
y := 0;  
while x <> N do  
    y := y + 2 * x + 1;  
    x := x + 1  
od  
{y = N2}
```

Étapes de boucle	x	y		
0	0	0		

# Exemple d'application de la méthode

Élévation au carré efficace

```
{0 ≤ N}  
x := 0;  
y := 0;  
while x <> N do  
    y := y + 2 * x + 1;  
    x := x + 1  
od  
{y = N2}
```

Étapes de boucle	x	y		
0	0	0		
1	1	1		



# Exemple d'application de la méthode

## Élévation au carré efficace

```
{0 ≤ N}  
x := 0;  
y := 0;  
while x <> N do  
    y := y + 2 * x + 1;  
    x := x + 1  
od  
{y = N2}
```

Étapes de boucle	x	y		
0	0	0		
1	1	1		
2	2	4		
3	3	9		
4	4	16		
⋮	⋮	⋮		

- Premier invariant candidat :
- Premier variant candidat :

# Exemple d'application de la méthode

## Élévation au carré efficace

```
{0 ≤ N}  
x := 0;  
y := 0;  
while x <> N do  
    y := y + 2 * x + 1;  
    x := x + 1  
od  
{y = N2}
```

Étapes de boucle	x	y	y = x <sup>2</sup>
0	0	0	⊤
1	1	1	⊤
2	2	4	⊤
3	3	9	⊤
4	4	16	⊤
⋮	⋮	⋮	⋮

- Premier invariant candidat :  $\varphi_0 : y = x^2$
- Premier variant candidat :

# Exemple d'application de la méthode

## Élévation au carré efficace

```
{0 ≤ N}  
x := 0;  
y := 0;  
while x <> N do  
    y := y + 2 * x + 1;  
    x := x + 1  
od  
{y = N2}
```

Étapes de boucle	x	y	$y = x^2$	$N - x$
0	0	0	⊤	N
1	1	1	⊤	N - 1
2	2	4	⊤	N - 2
3	3	9	⊤	N - 3
4	4	16	⊤	N - 4
⋮	⋮	⋮	⋮	⋮

- Premier invariant candidat :  $\varphi_0 : y = x^2$
- Premier variant candidat :  $N - x$

# Exemple d'application de la méthode

Élévation au carré efficace

$\{0 \leq N\}$

$x := 0;$

$y := 0;$

*while*  $x <> N$  *invariant*  $y = x^2$  *variant*  $N - x$  *do*

$y := y + 2 * x + 1;$

$x := x + 1$

*od*

$\{y = N^2\}$

# Exemple d'application de la méthode

## Élévation au carré efficace

$\{0 \leq N\}$

$x := 0;$

$y := 0;$

$\{y = x^2 \wedge (N - x) \in \mathbb{N}\}$

*while*  $x <> N$  *invariant*  $y = x^2$  *variant*  $N - x$  *do*

$\{y = x^2 \wedge x <> N \wedge (N - x) \in \mathbb{N} \wedge V = N - x\}$

$y := y + 2 * x + 1;$

$x := x + 1$

$\{y = x^2 \wedge (N - x) \in \mathbb{N} \wedge N - x < V\}$

*od*

$\{y = x^2 \wedge \neg(x <> N)\}$

$\{y = N^2\}$

# Exemple d'application de la méthode

## Élévation au carré efficace

$\{0 \leq N\}$

$x := 0;$

$y := 0;$

$\{y = x^2 \wedge (N - x) \in \mathbb{N}\}$

*while*  $x <> N$  *invariant*  $y = x^2$  *variant*  $N - x$  *do*

$\{y = x^2 \wedge x <> N \wedge (N - x) \in \mathbb{N} \wedge V = N - x\}$

$y := y + 2 * x + 1;$

$\{y = (x + 1)^2 \wedge (N - (x + 1)) \in \mathbb{N} \wedge N - (x + 1) < V\}$

$x := x + 1$

$\{y = x^2 \wedge (N - x) \in \mathbb{N} \wedge N - x < V\}$

*od*

$\{y = x^2 \wedge \neg(x <> N)\}$

$\{y = N^2\}$

# Exemple d'application de la méthode

## Élévation au carré efficace

$\{0 \leq N\}$

$x := 0;$

$y := 0;$

$\{y = x^2 \wedge (N - x) \in \mathbb{N}\}$

*while*  $x <> N$  *invariant*  $y = x^2$  *variant*  $N - x$  *do*

$\{y = x^2 \wedge x <> N \wedge (N - x) \in \mathbb{N} \wedge V = N - x\}$

$\{y + 2 \times x + 1 = (x + 1)^2 \wedge (N - (x + 1)) \in \mathbb{N} \wedge N - (x + 1) < V\}$

$y := y + 2 * x + 1;$

$\{y = (x + 1)^2 \wedge (N - (x + 1)) \in \mathbb{N} \wedge N - (x + 1) < V\}$

$x := x + 1$

$\{y = x^2 \wedge (N - x) \in \mathbb{N} \wedge N - x < V\}$

*od*

$\{y = x^2 \wedge \neg(x <> N)\}$

$\{y = N^2\}$

# Exemple d'application de la méthode

## Élévation au carré efficace

$\{0 \leq N\}$

$x := 0;$

$\{0 = x^2 \wedge (N - x) \in \mathbb{N}\}$

$y := 0;$

$\{y = x^2 \wedge (N - x) \in \mathbb{N}\}$

*while*  $x <> N$  *invariant*  $y = x^2$  *variant*  $N - x$  *do*

$\{y = x^2 \wedge x <> N \wedge (N - x) \in \mathbb{N} \wedge V = N - x\}$

$\{y + 2 \times x + 1 = (x + 1)^2 \wedge (N - (x + 1)) \in \mathbb{N} \wedge N - (x + 1) < V\}$

$y := y + 2 * x + 1;$

$\{y = (x + 1)^2 \wedge (N - (x + 1)) \in \mathbb{N} \wedge N - (x + 1) < V\}$

$x := x + 1$

$\{y = x^2 \wedge (N - x) \in \mathbb{N} \wedge N - x < V\}$

*od*

$\{y = x^2 \wedge \neg(x <> N)\}$

$\{y = N^2\}$



# Exemple d'application de la méthode

## Élévation au carré efficace

```
{0 ≤ N}  
{0 = 02 ∧ (N - 0) ∈ ℕ}  
x := 0;  
{0 = x2 ∧ (N - x) ∈ ℕ}  
y := 0;  
{y = x2 ∧ (N - x) ∈ ℕ}  
while x <> N invariant y = x2 variant N - x do  
  {y = x2 ∧ x <> N ∧ (N - x) ∈ ℕ ∧ V = N - x}  
  {y + 2 × x + 1 = (x + 1)2 ∧ (N - (x + 1)) ∈ ℕ ∧ N - (x + 1) < V}  
  y := y + 2 * x + 1;  
  {y = (x + 1)2 ∧ (N - (x + 1)) ∈ ℕ ∧ N - (x + 1) < V}  
  x := x + 1  
  {y = x2 ∧ (N - x) ∈ ℕ ∧ N - x < V}  
od  
{y = x2 ∧ ¬(x <> N)}  
{y = N2}
```

# Exemple d'application de la méthode

Élévation au carré efficace

$$0 \leq N \rightarrow 0 = 0^2 \wedge (N - 0) \in \mathbb{N}$$

$$(y = x^2 \wedge x <> N \wedge (N - x) \in \mathbb{N} \wedge V = N - x) \rightarrow (y + 2 \times x + 1 = (x + 1)^2)$$

$$(y = x^2 \wedge \neg(x <> N)) \rightarrow y = N^2$$

# Logique de Floyd/Hoare

## Bilan

- ▶ La logique de Floyd/Hoare est complète, consistante et correcte vis à vis de la sémantique des programmes
- ▶ Le calcul des plus faibles pré-conditions est complet, consistant et décidable
- ▶ Les caractéristiques de la logique de Hoare dépendent donc des caractéristiques de la logique utilisée pour exprimer les obligations de preuve
- ▶ Il existe de nombreux outils qui intègrent la logique de Floyd/Hoare dans les langages de programmation
  - ▶ SPARK Ada par AdaCore et Altran Praxis
  - ▶ CAVEAT et Frama-C par CEA
  - ▶ Why3 par LRI et INRIA
  - ▶ Boogie par MicroSoft Research
  - ▶ Spec# par MicroSoft Research
  - ▶  $F^*$  par MicroSoft Research

# Modélisation et Analyse des Informations Structurées

- ▶ Communication = Echange d'informations
- ▶ Besoins :
  - ▶ Représenter les informations possibles
  - ▶ Reconnaître une information
  - ▶ Exploiter une information
- ▶ Organisation stratifiée : information structurée

# Modélisation et Analyse des Informations Structurées

- ▶ Communication = Echange d'informations
- ▶ Besoins :
  - ▶ Représenter les informations possibles
  - ▶ Reconnaître une information
  - ▶ Exploiter une information
- ▶ Organisation stratifiée : information structurée
- ▶ Informatique : Science du traitement de l'information
- ▶ Computer science : Science de la « machine à calculer »

# Modélisation et Analyse des Informations Structurées

- ▶ Communication = Echange d'informations
- ▶ Besoins :
  - ▶ Représenter les informations possibles
  - ▶ Reconnaître une information
  - ▶ Exploiter une information
- ▶ Organisation stratifiée : information structurée
- ▶ Informatique : Science du traitement de l'information
- ▶ Computer science : Science de la « machine à calculer »
- ▶ Essentiel :
  - ▶ Description et manipulation de l'information (langage),
  - ▶ Traitement d'une information quelconque,
  - ▶ Traitement d'une manipulation quelconque
- ▶ D'où :
  - ▶ Description formelle du langage
  - ▶ Génération automatique des outils de manipulation

# Références bibliographiques

- ▶ Stern, Fondements mathématiques de l'informatique, McGraw-Hill, 1990.
- ▶ Hopcroft, Ullman, Introduction to automata theory, languages and computation, Addison-Wesley, 1979.
- ▶ Aho, Sethi, Ullman, Compilateurs : Principes, Techniques et Outils, InterEditions, 1989.
- ▶ Fisher, Leblanc, Crafting a compiler in ADA/in C, Benjamin Cummings, 1991.
- ▶ Wilhem, Maurer, Les compilateurs : Théorie, construction, génération, Masson, 1994.
- ▶ WWW Consortium : <http://www.w3c.org>.
- ▶ E-R. Harold, W-S. Means, XML in a nutshell, O'Reilly, 2001.
- ▶ JSON : <http://json.org>

## Exemple : fichier /etc/hosts

Informations brutes : caractères

```
# Ceci est un commentaire
```

```
\n
```

```
12.70.0.1\t halloween localhost
```

```
\n
```

```
# En voici un autre
```

```
\n
```

```
147.127.18.144 enseiht.enseiht.fr
```



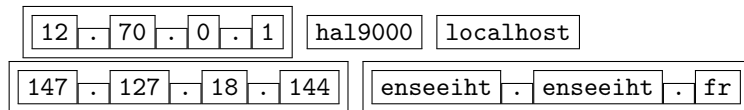
# Analyse lexicale

Informations élémentaires : commentaire, nombre, identificateur, .  
(unités lexicales)

# Ceci est un commentaire	12	.	70	.	0	.	1
hal9000	localhost	# En voici un autre	147	.	127	.	
18	.	144	enseeiht	.	enseeiht	.	fr

# Analyse syntaxique

Informations structurées : adresse IP, nom qualifié (unités syntaxiques)



# Analyse sémantique

Exploitation des informations : association nom qualifié/adresse IP  
(unités sémantiques)

hal9000

localhost

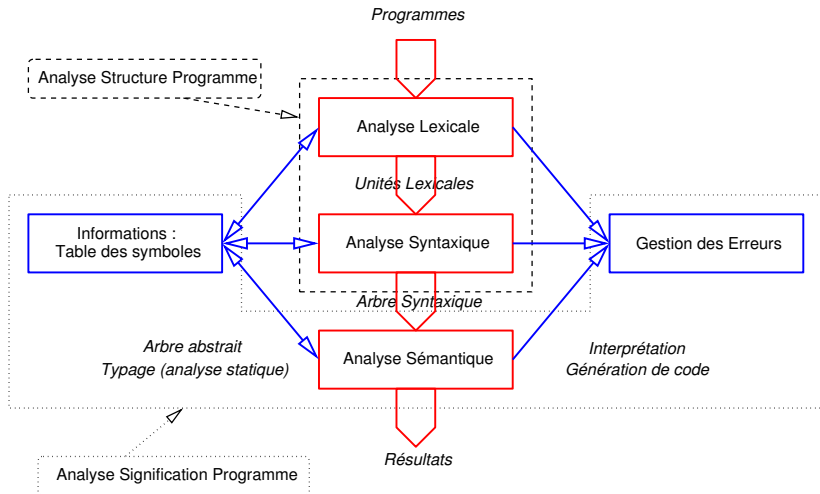
enseeiht . enseeiht . fr

12 . 70 . 0 . 1

12 . 70 . 0 . 1

147 . 127 . 18 . 144

# Structure d'un outil



# Définitions

- ▶ Caractère/Symbole : Unité élémentaire d'information
- ▶ Unité lexicale (lexème, mot) : Séquence de caractères
- ▶ Unité syntaxique (arbre syntaxique, syntème, phrase) : Arbre d'unités lexicales
- ▶ Unité sémantique : diverses (arbre abstrait, table des symboles, type, code généré, résultat évaluation, ...)

# Comment organiser les informations ?

- ▶ Objectif : Exploitation des informations
- ▶ Règle : Choisir le bon niveau de précision
- ▶ Unité lexicale : Bloc élémentaire d'information pertinente
- ▶ Unité syntaxique : Élément structurant de l'information

# Approche XML pour les informations structurées

- ▶ Signification : eXtensible Markup Language (langage à balises évolutif)
- ▶ Objectif : format universel pour décrire des documents arborescents structurés
- ▶ Evolutif : définition de nouveaux éléments (DTD, XML Schemas)

# Points importants

- ▶ Document bien formé : Structure arborescente
- ▶ Document valide : Respect arborescence particulière (famille de documents – DTD, XML Schemas, types de documents)
- ▶ Outil de description de données portables
- ▶ Approches :
  - ▶ Donnée : structure rigoureuse
  - ▶ Littéraire : structure partielle/libre
- ▶ Structure sémantique et pas figurative ( $\neq$  HTML)
- ▶ Important :
  - ▶ Ni un langage de programmation (peut représenter arbre abstrait)
  - ▶ Ni un protocole de transport (utilise HTTP)
  - ▶ Ni une base de donnée



# Historique

- ▶ Années 70 : Systèmes de Gestion de Données Techniques (SGDT)
- ▶ IBM Generalized Markup Language (GML) normalisé en 1986 (SGML)
- ▶ Application : Famille de données similaires représentée par une DTD
- ▶ Exemple : HTML (HyperText Markup Language), description pages WEB
- ▶ exclusivement figuratif (présentation sans sémantique) extension impose changement norme
- ▶ Solution : XML = SGML «allégé» proposé et standardisé par W3C

# Outils et Applications

- ▶ Outils d'analyse :
  - ▶ SAX : Sample Api for Xml parsing
  - ▶ DOM, DOM4J, JDOM : Document Object Model
- ▶ Format autre qu'XML :
  - ▶ XPath : filtrage de documents XML (extraction de parties)
  - ▶ CSS (Cascading Style Sheets) : feuilles de style hiérarchiques
  - ▶ DTD : Document Type Definition

# Outils et Applications (bis)

- ▶ Format XML :
  - ▶ XFragment : document composé de plusieurs autres documents
  - ▶ XMLQuery : filtrage de documents
  - ▶ XLinks : liens entre documents XML et non XML
  - ▶ XPointers : liens au sein d'un document
  - ▶ Namespace : organisation modulaire
  - ▶ XHTML : Version structurée de HTML
  - ▶ XSLT (eXtensible Stylesheet Language-Transformation) : transformations de documents XML
  - ▶ XSL-FO (XSL-Formating Objects) : aspect figuratif
  - ▶ XML Schemas : Description structure documents
  - ▶ SOAP (Sample Object Access Protocol), XML-RPC
  - ▶ SVG (Scalable Vector Graphics), RDF (Ressource Description Framework), SMIL (synchronisation, multimédia), MathML (formule mathématiques), ...

# Format d'un document XML

- ▶ Contrainte : Format textuel

- ▶ Balise :  
début `<ident>`  
fin `</ident>`

- ▶ Élément vide : `<ident />`  $\equiv$  `<ident></ident>`

- ▶ Identificateur : lettre, chiffre, `_`, `-`, `.`, `:` (espace nommage)

- ▶ Attribut : `ident = "chaîne"`

- ▶ Référence entité : `&ident;`

- ▶ Exemple caractères spéciaux :

<	>	&	"	'	...
<code>&amp;lt;</code>	<code>&amp;gt;</code>	<code>&amp;amp;</code>	<code>&amp;quot;</code>	<code>&amp;apos;</code>	...

# Exemple de documents

## ► Sans attributs :

```
<enseignant>
  <identite>
    <prenom>Marc</prenom>
    <nom>Pantel</nom>
  </identite>
  <discipline>Traduction des langages</discipline>
  <discipline>Sémantique formelle</discipline>
</enseignant>
```

## ► Avec attributs :

```
<enseignant>
  <identite prenom="Marc" nom="Pantel"/>
  <discipline nom="Traduction des langages"/>
  <discipline nom="Sémantique formelle"/>
</enseignant>
```

# Contenu du document

► Texte brut : `<![CDATA[texte]]>`

► Commentaires : `<!-- texte -->`

► Ordre applicatif : `<? cible texte ?>`

► Exemple : `<?php ...?>`

► Déclaration : ordre applicatif XML

<code>&lt;?xml</code>	<code>version="..."</code>	<code>encoding="..."</code>	<code>standalone="..."</code>	<code>?&gt;</code>
	$\underbrace{\hspace{1.5cm}}$	$\underbrace{\hspace{1.5cm}}$	$\underbrace{\hspace{1.5cm}}$	
	1.0	défaut UTF8	yes ou no	

# Document bien formé

- ▶ Chaque balise de début a une balise de fin
- ▶ Pas de recouvrement d'éléments distincts
- ▶ Une seule racine
- ▶ Valeurs des attributs entre `"..."`
- ▶ Unicité des attributs par éléments
- ▶ Pas de commentaires dans les balises
- ▶ Pas de caractères `<`, `>`, `&`, `"`, `'` dans le texte

# Représentation en JavaScript des informations structurées

- ▶ Signification : JavaScript Object Notation (représentation des objets JavaScript)
- ▶ Objectif : format texte pour échanger des données entre client et serveur en JavaScript
- ▶ Différence majeure avec XML : Pas de familles de documents (DTD/Schema – Typage de documents)



## Exemple de documents

```
{
  "prenom" : "Marc",
  "nom" : "Pantel",
  "naissance" : {
    "jour" : 4,
    "mois" : "mai",
    "annee" : 1966
  },
  "matiere" : [ "OMI", "TOB", "GLS", "STL", "SCP", "CL" ]
}
```

# Structure algébrique des langages

- ▶ Alphabet : Ensemble fini de symboles  $A = \{s_i\}$
- ▶ Mot : Suite de symbole  $s_1 s_2 \dots s_n$  (mot vide :  $\Lambda$ )
- ▶ Langage : Ensemble de mots ( Langage vide :  $\emptyset = \{\}$  )
- ▶ Opération de concaténation :  $\bullet : \text{Mot} \times \text{Mot} \mapsto \text{Mot}$
- ▶  $s_1 s_2 \dots s_n = (\dots (s_1 \bullet s_2) \bullet \dots) \bullet s_n$

# Structure algébrique des langages

- ▶ Alphabet : Ensemble fini de symboles  $A = \{s_i\}$
- ▶ Mot : Suite de symbole  $s_1 s_2 \dots s_n$  (mot vide :  $\Lambda$ )
- ▶ Langage : Ensemble de mots ( Langage vide :  $\emptyset = \{\}$  )
- ▶ Opération de concaténation :  $\bullet : \text{Mot} \times \text{Mot} \mapsto \text{Mot}$
- ▶  $s_1 s_2 \dots s_n = (\dots (s_1 \bullet s_2) \bullet \dots) \bullet s_n$
- ▶ Associatif :  
 $(\dots (s_1 \bullet s_2) \bullet \dots) \bullet s_n = s_1 \bullet s_2 \bullet \dots \bullet s_n = s_1 \bullet (\dots \bullet (s_{n-1} \bullet s_n) \dots)$
- ▶ Élément neutre : Suite vide  $\Lambda$
- ▶ Extension à des langages :  
 $X \bullet Y = \{x \bullet y \mid x \in X, y \in Y\}$
- ▶ d'où l'opération  $X^n$  définie par :  
 $X^0 = \{\Lambda\}$  et  $X^{n+1} = X \bullet X^n$

# Etoile de Kleene

- ▶ Objectif : Construire l'ensemble de tous les mots possibles
- ▶ Mot vide (0 symbole) :  $A^0 = \{\Lambda\}$  (  $\neq \emptyset = \{\}$  langage vide)
- ▶ Mots d'1 symbole :  $A^1 = A$
- ▶ Mots de 2 symboles :  $A^2 = A \bullet A$
- ▶ Mots de  $n$  symboles :  $A^n$
- ▶ d'où  $A^* = \bigcup_{i=0}^{\infty} A^i$
- ▶ Un langage sur  $A$  est un sous-ensemble de  $A^*$ .
- ▶ L'ensemble des langages sur  $A$  est  $\mathcal{P}(A^*)$ .
- ▶  $\Rightarrow$  Structure de  $A^*$  muni de  $\bullet$  : Monoïde libre engendré par  $A$ .

# Opérateurs sur les langages

$$\overline{L} = \{m \in A^* \mid m \notin L\}$$

$$L_1 \cup L_2 = \{m \in A^* \mid m \in L_1 \vee m \in L_2\}$$

$$L_1 \cap L_2 = \{m \in A^* \mid m \in L_1 \wedge m \in L_2\} = \overline{\overline{L_1} \cup \overline{L_2}}$$

$$L_1 \setminus L_2 = \{m \in A^* \mid m \in L_1 \wedge m \notin L_2\} = L_1 \cap \overline{L_2}$$

$$L_1 \bullet L_2 = \{m_1 \bullet m_2 \in A^* \mid m_1 \in L_1 \wedge m_2 \in L_2\}$$

$$L^0 = \{\Lambda\} \neq \emptyset = \{\}$$

$$L^{n+1} = L \bullet L^n$$

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

$$L^+ = \bigcup_{i=1}^{\infty} L^i = L \bullet L^* = L^* \setminus \{\Lambda\} \text{ si } \Lambda \notin L$$

# Expressions régulières

- ▶ Objectif : Description formelle de langages (langage de description de langages)
- ▶ Moyen : Langage vide, mot vide, symboles de l'alphabet, opérateurs sur les langages

# Expressions régulières

- ▶ Objectif : Description formelle de langages (langage de description de langages)
- ▶ Moyen : Langage vide, mot vide, symboles de l'alphabet, opérateurs sur les langages

- ▶ Constantes :

Signification	Notation
Langage contenant le mot composé du symbole $s$ de $A$	$s$
Langage contenant les mots composés des symboles $s_m$ à $s_{m+n}$ dans l'ordre de $A$	$[s_m - s_{m+n}]$
Langage contenant le mot vide	$\Lambda$
Langage vide	$\emptyset$

# Expressions régulières

- Objectif : Description formelle de langages (langage de description de langages)
- Moyen : Langage vide, mot vide, symboles de l'alphabet, opérateurs sur les langages

- Constantes :

Signification	Notation
Langage contenant le mot composé du symbole $s$ de $A$	$s$
Langage contenant les mots composés des symboles $s_m$ à $s_{m+n}$ dans l'ordre de $A$	$[s_m - s_{m+n}]$
Langage contenant le mot vide	$\Lambda$
Langage vide	$\emptyset$

- Opérateurs :

Signification	Notation	notation alternative
Concaténation de langages	$e_1 e_2$	$e_1 \bullet e_2$
Choix entre deux langages	$e_1 \mid e_2$	$e_1 + e_2$
Option	$e?$	
Répétition d'un langage	$e^*$	
	$e^+$	
	$e\{n_1, n_2\}$	
Complément d'un langage	$[\wedge s_m - s_{m+n}]$	



# Langage associé

$$\begin{aligned}L(\emptyset) &= \{\} \\L(\wedge) &= \{\wedge\} \\L(s) &= \{s\} \\L(e_1 \mid e_2) &= L(e_1) \cup L(e_2) \\L(e_1 e_2) &= L(e_1) \bullet L(e_2) \\L(e^\star) &= L(e)^\star = \bigcup_{i=0}^{\infty} L(e)^i \\L(e^+) &= L(e) \bullet L(e)^\star = \bigcup_{i=1}^{\infty} L(e)^i \\L(e?) &= \{\wedge\} \cup L(e) \\L(e\{n_1, n_2\}) &= \bigcup_{i=n_1}^{n_2} L(e)^i \\L([s_m - s_{m+n}]) &= \{s_i\}_{i \in [m, \dots, m+n]} \\L([\wedge s_m - s_{m+n}]) &= A \setminus \{s_i\}_{i \in [m, \dots, m+n]}\end{aligned}$$

# Propriétés des opérateurs

$$\emptyset e = e \emptyset = \emptyset$$

$$e \mid \emptyset = \emptyset \mid e = e$$

$$e_1 (e_2 e_3) = (e_1 e_2) e_3$$

$$e_1 (e_2 \mid e_3) = (e_1 e_2) \mid (e_1 e_3)$$

$$e_1 \mid e_2 = e_2 \mid e_1$$

$$e \star = \Lambda \mid e \mid$$

$$e \star e \star = e \star$$

$$e = e \star \Leftrightarrow e = e e$$

$$(e_1 \star e_2 \star) \star = (e_1 \mid e_2) \star = (e_1 \star \mid e_2 \star) \star$$

$$(e_1 \star e_2) \star (e_1 \star) = (e_1 \mid e_2) \star = e_1 \star (e_2 (e_1 \star)) \star$$

$$\Lambda e = e \Lambda = e$$

$$e \mid e = e$$

$$e_1 \mid (e_2 \mid e_3) = (e_1 \mid e_2) \mid e_3$$

$$(e_1 \mid e_2) e_3 = (e_1 e_2) \mid (e_1 e_3)$$

$$\emptyset \star = \Lambda \star = \Lambda$$

$$e \mid = e e \star = e \star e$$

$$e \star \star = e \star$$

$$e e \star = e \star \Leftrightarrow \Lambda \in L(e)$$

# Exemples

- ▶ Nombres entiers naturels :  $\text{nombre} = [0-9]^+$
- ▶ Notation usuelle :  $\text{entier} = ([1-9][0-9]^*) \mid 0$
- ▶ Nombres entiers relatifs :  $\text{relatif} = ('+' \mid '-' )? \{ \text{nombre} \}$
- ▶ Nombres décimaux :  
 $\text{decimal} = \{ \text{relatif} \} ( \. \{ \text{nombre} \} )? ( [eE] \{ \text{relatif} \} )?$
- ▶ Et ensuite : Expressions arithmétiques
- ▶ Limites : Expressions parenthésées (association des parenthèses – langages de Dick)

# Grammaire

- ▶ Quadruplet  $(A, V, S, P)$  tel que :
  - ▶  $A$  : ens. fini symboles terminaux (alphabet)
  - ▶  $V$  : ens. fini symboles non-terminaux (  $A \cap V = \emptyset$  )
  - ▶  $S \in V$  : axiome
  - ▶  $P \subseteq ((A \cup V)^* \bullet V \bullet (A \cup V)^*) \times (A \cup V)^*$  :  
ensemble fini de règles de production notées  $\alpha \rightarrow \beta$  ( $\beta$  est la production de  $\alpha$ )
- ▶ Notations :
  - ▶  $a, b, \dots \in A$
  - ▶  $A, B, \dots \in V$
  - ▶  $\alpha, \beta, \dots \in (A \cup V)^*$

## Exemple : Expressions Arithmétiques

$$A = \{ + \ * \ - \ c \ i \ , \ ( \ ) \ [ \ ] \}$$

$$V = \{ E, L_E, S_E \}$$

$$S = E$$

$$P = \left\{ \begin{array}{l} E \rightarrow E + E \\ E \rightarrow E * E \\ E \rightarrow - E \\ E \rightarrow c \\ E \rightarrow i \\ E \rightarrow i [ L_E ] \\ E \rightarrow ( E ) \\ L_E \rightarrow E S_E \\ S_E \rightarrow , E S_E \\ S_E \rightarrow \Lambda \end{array} \right\}$$

# Dérivation

Langage décrit par une grammaire

- ▶ Relation  $\alpha \Rightarrow \beta$  si et seulement si 
$$\left\{ \begin{array}{l} \alpha = \delta_1 \gamma_1 \delta_2 \\ \beta = \delta_1 \gamma_2 \delta_2 \\ \gamma_1 \rightarrow \gamma_2 \in P \\ \delta_i, \gamma_i \in (A \cup V)^* \end{array} \right.$$
- ▶ Fermeture réflexive transitive :  $\alpha \Rightarrow^* \beta$
- ▶  $\alpha$  se dérive en  $\beta$  en utilisant  $P$
- ▶ Notation :  $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_{n-1} \Rightarrow \beta$
- ▶  $L(G) = \{m \in A^* \mid S \Rightarrow^* m\}$
- ▶ Exemple : Dérivation pour  $E \Rightarrow^* 1 + 2 * 3$

$$\begin{aligned} E &\Rightarrow \underbrace{E * E} \\ \alpha &= E \\ \beta &= E * E \\ \delta_1 &= \Lambda \\ \gamma_1 &= E \\ \delta_2 &= \Lambda \\ \gamma_2 &= E * E \end{aligned}$$

# Dérivation

Langage décrit par une grammaire

- ▶ Relation  $\alpha \Rightarrow \beta$  si et seulement si 
$$\left\{ \begin{array}{l} \alpha = \delta_1 \gamma_1 \delta_2 \\ \beta = \delta_1 \gamma_2 \delta_2 \\ \gamma_1 \rightarrow \gamma_2 \in P \\ \delta_i, \gamma_i \in (A \cup V)^* \end{array} \right.$$
- ▶ Fermeture réflexive transitive :  $\alpha \Rightarrow^* \beta$
- ▶  $\alpha$  se dérive en  $\beta$  en utilisant  $P$
- ▶ Notation :  $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_{n-1} \Rightarrow \beta$
- ▶  $L(G) = \{m \in A^* \mid S \Rightarrow^* m\}$
- ▶ Exemple : Dérivation pour  $E \Rightarrow^* 1 + 2 * 3$

$$\begin{array}{ll} E \Rightarrow \underbrace{E * E} & \Rightarrow \underbrace{E + E * E} \\ \alpha = E & \alpha = E * E \\ \beta = E * E & \beta = E + E * E \\ \delta_1 = \Lambda & \delta_1 = \Lambda \\ \gamma_1 = E & \gamma_1 = E \\ \delta_2 = \Lambda & \delta_2 = *E \\ \gamma_2 = E * E & \gamma_2 = E + E \end{array}$$

# Dérivation

Langage décrit par une grammaire

- ▶ Relation  $\alpha \Rightarrow \beta$  si et seulement si 
$$\left\{ \begin{array}{l} \alpha = \delta_1 \gamma_1 \delta_2 \\ \beta = \delta_1 \gamma_2 \delta_2 \\ \gamma_1 \rightarrow \gamma_2 \in P \\ \delta_i, \gamma_i \in (A \cup V)^* \end{array} \right.$$
- ▶ Fermeture réflexive transitive :  $\alpha \Rightarrow^* \beta$
- ▶  $\alpha$  se dérive en  $\beta$  en utilisant  $P$
- ▶ Notation :  $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_{n-1} \Rightarrow \beta$
- ▶  $L(G) = \{m \in A^* \mid S \Rightarrow^* m\}$
- ▶ Exemple : Dérivation pour  $E \Rightarrow^* 1 + 2 * 3$

$E \Rightarrow E * E$	$\Rightarrow E + E * E$	$\Rightarrow 1 + E * E$
$\alpha = E$	$\alpha = E * E$	$\alpha = E + E * E$
$\beta = E * E$	$\beta = E + E * E$	$\beta = 1 + E * E$
$\delta_1 = \Lambda$	$\delta_1 = \Lambda$	$\delta_1 = \Lambda$
$\gamma_1 = E$	$\gamma_1 = E$	$\gamma_1 = E$
$\delta_2 = \Lambda$	$\delta_2 = *E$	$\delta_2 = +E * E$
$\gamma_2 = E * E$	$\gamma_2 = E + E$	$\gamma_2 = 1$



# Dérivation

Langage décrit par une grammaire

- ▶ Relation  $\alpha \Rightarrow \beta$  si et seulement si 
$$\left\{ \begin{array}{l} \alpha = \delta_1 \gamma_1 \delta_2 \\ \beta = \delta_1 \gamma_2 \delta_2 \\ \gamma_1 \rightarrow \gamma_2 \in P \\ \delta_i, \gamma_i \in (A \cup V)^* \end{array} \right.$$
- ▶ Fermeture réflexive transitive :  $\alpha \Rightarrow^* \beta$
- ▶  $\alpha$  se dérive en  $\beta$  en utilisant  $P$
- ▶ Notation :  $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_{n-1} \Rightarrow \beta$
- ▶  $L(G) = \{m \in A^* \mid S \Rightarrow^* m\}$
- ▶ Exemple : Dérivation pour  $E \Rightarrow^* 1 + 2 * 3$

$E \Rightarrow E * E$	$\Rightarrow E + E * E$	$\Rightarrow 1 + E * E$	$\Rightarrow 1 + 2 * E$
$\alpha = E$	$\alpha = E * E$	$\alpha = E + E * E$	$\alpha = 1 + E * E$
$\beta = E * E$	$\beta = E + E * E$	$\beta = 1 + E * E$	$\beta = 1 + 2 * E$
$\delta_1 = \Lambda$	$\delta_1 = \Lambda$	$\delta_1 = \Lambda$	$\delta_1 = 1 +$
$\gamma_1 = E$	$\gamma_1 = E$	$\gamma_1 = E$	$\gamma_1 = E$
$\delta_2 = \Lambda$	$\delta_2 = *E$	$\delta_2 = +E * E$	$\delta_2 = *E$
$\gamma_2 = E * E$	$\gamma_2 = E + E$	$\gamma_2 = 1$	$\gamma_2 = 2$

# Dérivation

Langage décrit par une grammaire

- ▶ Relation  $\alpha \Rightarrow \beta$  si et seulement si  $\left\{ \begin{array}{l} \alpha = \delta_1 \gamma_1 \delta_2 \\ \beta = \delta_1 \gamma_2 \delta_2 \\ \gamma_1 \rightarrow \gamma_2 \in P \\ \delta_i, \gamma_i \in (A \cup V)^* \end{array} \right.$
- ▶ Fermeture réflexive transitive :  $\alpha \Rightarrow^* \beta$
- ▶  $\alpha$  se dérive en  $\beta$  en utilisant  $P$
- ▶ Notation :  $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_{n-1} \Rightarrow \beta$
- ▶  $L(G) = \{m \in A^* \mid S \Rightarrow^* m\}$
- ▶ Exemple : Dérivation pour  $E \Rightarrow^* 1 + 2 * 3$

$E$	$\Rightarrow E * E$	$\Rightarrow E + E * E$	$\Rightarrow 1 + E * E$	$\Rightarrow 1 + 2 * E$	$\Rightarrow 1 + 2 * 3$
$\alpha = E$	$\alpha = E * E$	$\alpha = E + E * E$	$\alpha = 1 + E * E$	$\alpha = 1 + 2 * E$	$\alpha = 1 + 2 * 3$
$\beta = E * E$	$\beta = E + E * E$	$\beta = 1 + E * E$	$\beta = 1 + 2 * E$	$\beta = 1 + 2 * 3$	
$\delta_1 = \Lambda$	$\delta_1 = \Lambda$	$\delta_1 = \Lambda$	$\delta_1 = 1 +$	$\delta_1 = 1 + 2 *$	
$\gamma_1 = E$	$\gamma_1 = E$	$\gamma_1 = E$	$\gamma_1 = E$	$\gamma_1 = E$	
$\delta_2 = \Lambda$	$\delta_2 = *E$	$\delta_2 = +E * E$	$\delta_2 = *E$	$\delta_2 = \Lambda$	
$\gamma_2 = E * E$	$\gamma_2 = E + E$	$\gamma_2 = 1$	$\gamma_2 = 2$	$\gamma_2 = 3$	

# Stratégies de dérivation

- ▶ Un même mot peut être engendré par plusieurs dérivation distinctes
- ▶ Analyse déterministe requiert une stratégie lors de la reconnaissance de ce mot
- ▶ Dérivation la plus à gauche (Leftmost) : Dérivation du non-terminal le plus à gauche (Analyseur LL : Left to right/Leftmost)

$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow 1 + E * E \Rightarrow 1 + 2 * E \Rightarrow 1 + 2 * 3$$

- ▶ Dérivation la plus à droite (Rightmost) : Dérivation du non-terminal le plus à droite (Analyseur LR : Left to right/Rightmost)

$$E \Rightarrow E * E \Rightarrow E * 3 \Rightarrow E + E * 3 \Rightarrow E + 2 * 3 \Rightarrow 1 + 2 * 3$$

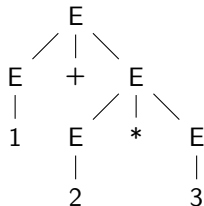
# Arbres de dérivation (syntaxique)

- ▶ Objectif : Représenter la structure du mot induite par les règles de production lors d'une dérivation
- ▶ Feuilles de l'arbre : Terminaux composant le mot
- ▶ Racine de l'arbre : Axiome
- ▶ Nœuds de l'arbre : Non-terminaux apparaissant dans la dérivation
- ▶ Branches de l'arbre : Règles de production

# Arbres de dérivation (syntaxique)

- ▶ Objectif : Représenter la structure du mot induite par les règles de production lors d'une dérivation
- ▶ Feuilles de l'arbre : Terminaux composant le mot
- ▶ Racine de l'arbre : Axiome
- ▶ Nœuds de l'arbre : Non-terminaux apparaissant dans la dérivation
- ▶ Branches de l'arbre : Règles de production

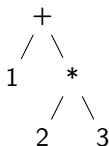
Exemple :  
Arbre de dérivation  
pour  $1 + 2 * 3$



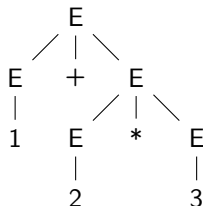
# Arbres de dérivation (syntaxique)

- ▶ Objectif : Représenter la structure du mot induite par les règles de production lors d'une dérivation
- ▶ Feuilles de l'arbre : Terminaux composant le mot
- ▶ Racine de l'arbre : Axiome
- ▶ Nœuds de l'arbre : Non-terminaux apparaissant dans la dérivation
- ▶ Branches de l'arbre : Règles de production

Attention :  $\neq$  Arbre abstrait

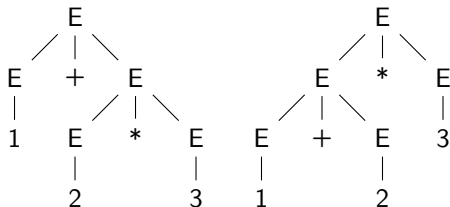


Exemple :  
Arbre de dérivation  
pour 1 + 2 \* 3



# Grammaire et langage ambigu

- ▶ Une grammaire est ambiguë s'il existe plusieurs arbres de dérivation distincts pour un même mot
- ▶ Exemple : Arbres de dérivation pour  $1 + 2 * 3$



- ▶ Un langage est ambigu si toutes les grammaires le représentant sont ambiguës

# Typologie des grammaires

## Travaux de Noam Chomsky

Type	Productions	Nom
0	$\alpha \rightarrow \beta$	
1	$\gamma X \delta \rightarrow \beta$ $\alpha \rightarrow \beta,  \alpha  <  \beta , \alpha \rightarrow \Lambda$	Contextuel Taille croissante
2	$X \rightarrow \beta$	Non contextuel Algébrique
$LR(k)$	$X \rightarrow \beta$ déterministe	
3	$X \rightarrow aY, X \rightarrow \Lambda$ $X \rightarrow Ya, X \rightarrow \Lambda$	Régulier à droite Régulier à gauche

Type	Analyse	Isomorphe
0	Indécidable, Coûteuse	Machine de Turing
1	Décidable, Coûteuse	
2	Décidable, Indéterministe	Automate fini à pile indéterministe
$LR(k)$	Décidable, Performante	Automate fini à pile déterministe
3	Décidable, Performante	Automate fini



# Notation BNF

- ▶ Introduction du choix dans les productions (factorisation)
- ▶ Non-terminaux :  $\langle X \rangle$
- ▶ Productions :  $X \rightarrow \alpha$  notée :  $\langle X \rangle ::= \alpha$
- ▶ Choix :  $\{X \rightarrow \alpha_i\}_{i \in [1, \dots, n]}$  noté :  $\langle X \rangle ::= \alpha_1 \mid \dots \mid \alpha_n$

# Notation EBNF

- ▶ Introduction des répétitions dans les productions
- ▶ Répétition 0 ou plus de  $\alpha$  :  $\{\alpha\}$
- ▶  $\alpha$  optionnel :  $[\alpha]$
- ▶ Associativité/Priorité :  $(\alpha)$

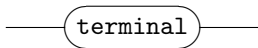
## Exemple : Expressions Arithmétiques

```
<Expression> ::=  
  (  
    <Expression> ( + | * ) <Expression>  
  |  
    - <Expression>  
  |  
    CONSTANCE  
  |  
    IDENTIFICATEUR  
    [  
      \[  
        <Expression> { VIRGULE <Expression> }  
      \]  
    ]  
  |  
    \ ( <Expression> \ )  
  )
```

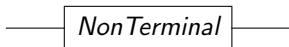
# Notation de Conway

- ▶ Forme graphique plus intuitive

- ▶ Terminal :

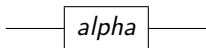


- ▶ Non terminal :



- ▶ Production  $X \rightarrow \alpha$  :

$X$

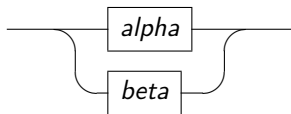


- ▶ Séquence  $\alpha\beta$  :

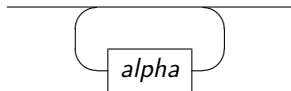


# Notation de Conway (bis)

- Choix entre  $\alpha$  et  $\beta$  :





- Répétition 0 ou plus de  $\alpha$  :





# XML : Document Type Definition

- ▶ Objectif : Définir la structure arborescente d'un document
- ▶ Définition des éléments (balises) et de leur contenu (structure)
- ▶ Inspiré de la notation EBNF : mélange règles de production  
expression régulière
- ▶ Typage d'un document :
- ▶ DTD externe : `<!DOCTYPE ident SYSTEM "url">`
- ▶ DTD interne : `<!DOCTYPE ident [  ... ]>`  
éléments
- ▶ Définition éléments : `<!ELEMENT ident  (...) >`  
structure

# Structure d'un élément

- ▶ Séquence d'autres éléments

▶ Avec :	Répétition	<code>*</code> , <code>+</code> , <code>?</code>
	Choix	<code>...   ...</code>
	Associativité/Priorité	<code>( ... )</code>
	Feuille texte	<code>#PCDATA</code>
	Feuille vide	<code>EMPTY</code>
	Feuille quelconque	<code>ANY</code>

- ▶ Problème : Récursivité entre éléments interdites par certains outils (utilisation d'entités)



# Exemples

```
<!DOCTYPE PERSONNEL [  
  <!ELEMENT enseignant (identité,displine+)>  
  <!ELEMENT identité ((prénom,nom) | (nom,prénom))>  
  <!ELEMENT prénom #PCDATA>  
  <!ELEMENT nom #PCDATA>  
  <!ELEMENT discipline #PCDATA>  
>
```

```
<enseignant>  
  <identite>  
    <prenom>Marc</prenom>  
    <nom>Pantel</nom>  
  </identite>  
  <discipline>Traduction des langages</discipline>  
  <discipline>Sémantique formelle</discipline>  
</enseignant>
```

# Structure d'un attribut

► Déclaration : 

<code>&lt;!ATTLIST</code>	$\underbrace{\textit{ident}}$	$\underbrace{\textit{ident type mode}}$	<code>&gt;</code>
	élément	répétition	

► Mode d'attributs :  
Obligatoire : 

<code>#REQUIRED</code>
------------------------

  
Optionnel : 

<code>#IMPLIED</code>
-----------------------

  
Constant : 

<code>#FIXED</code>
---------------------

# Structure d'un attribut (bis)

► Type d'attributs :

CDATA	texte brut
NMTOKEN	identificateur XML
NMTOKENS	suite d'identificateurs XML
ENUMERATION	valeurs possibles
ID	unique
IDREF	référence ID
IDREFS	suite de références ID
ENTITY	nom d'entité
ENTITIES	suite de noms d'entité
NOTATION	type MIME

# Exemple

```
<!DOCTYPE PERSONNEL [  
  <!ELEMENT enseignant (identité,displine+)>  
  <!ELEMENT identité #EMPTY>  
  <!ELEMENT discipline #EMPTY>  
  
  <!ATTLIST identité  
    nom CDATA #REQUIRED  
    prénom CDATA #REQUIRED  
  >  
  <!ATTLIST discipline  
    nom CDATA #REQUIRED  
  >  
>  
  
<enseignant>  
  <identite prenom="Marc" nom="Pantel"/>  
  <discipline nom="Traduction des langages"/>  
  <discipline nom="Sémantique formelle"/>  
</enseignant>
```

# Entité

- ▶ Macro-définitions substituées «à la demande» (récursivité possible)
- ▶ Utilisation externe DTD :

- ▶ Définition simple :

<code>&lt;!ENTITY <i>ident</i> ' ...'&gt;</code>
<code>&lt;!ENTITY <i>ident</i> " ..."&gt;</code>

- ▶ Accès simple : `&ident ;`

- ▶ Utilisation interne DTD :

- ▶ Définition paramétrée :

<code>&lt;!ENTITY % <i>ident</i> ' ...'&gt;</code>
<code>&lt;!ENTITY % <i>ident</i> " ..."&gt;</code>

- ▶ Accès paramétré : `%ident ;`

# Exemple

```
<!DOCTYPE ARBRE [  
    <!ENTITY %arbre; "(noeud|feuille)">  
    <!ELEMENT noeud (%arbre;,étiquette,%arbre;)>  
    <!ELEMENT feuille (étiquette)>  
    <!ELEMENT étiquette #PCDATA>  
>
```

- Problème : Entités paramétrées mal prise en compte par certains outils pour les DTD internes