Approches formelles pour la vérification de programmes

Catherine Dubois et Guillaume Burel (ENSIIE)

Master 2 CNS

But de la vérification formelle

Etant donnée une spécification formelle S d'un programme P,

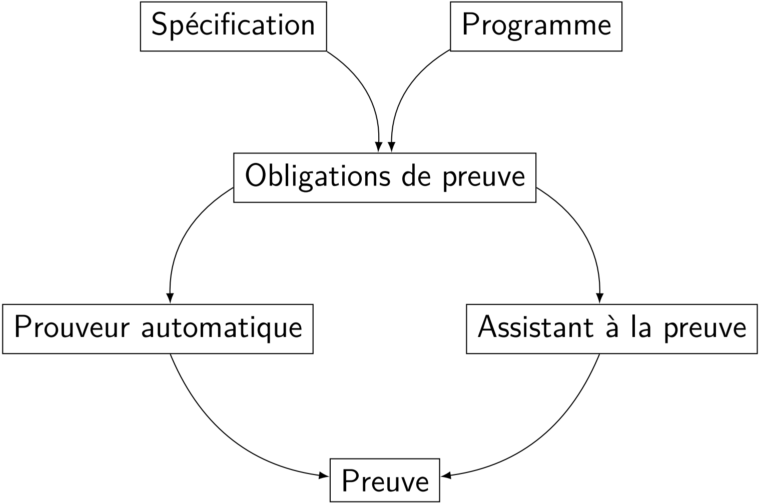
donner une preuve mathématique rigoureuse que toute exécution de P satisfait S

P est correct par rapport à S

Que faut-il pour vérifier formellement P ?

- une spécification formelle de P (rigoureuse, mathématique),
- une méthode de preuve de correction (logique de Hoare par exemple),
- des règles de preuve pour faire des preuves rigoureuses, mathématiques : un calcul.

Workflow


Spécification
Programme
Obligations de preuve
Prouveur automatique
Assistant à la preuve
Preuve

Preuves en logique du premier ordre

Logique propositionnelle

- Variables propositionnelles
  P, Q, R, $\ldots$
- Propositions atomiques
  $\top$, $\bot$
- Connecteurs
  - $\neg A$
  - $A \vee B$
  - $A \wedge B$
  - $A \Rightarrow B$

Exemple : $(P \wedge (\neg Q \Rightarrow (P \Rightarrow \bot))) \Rightarrow Q$

Logique du premier ordre

- Termes :
  - Variables X, Y, Z, ...
  - Symboles de fonction f, g, $+$, ...(y compris constantes)
- Propositions (= formules)
  - Prédicats : symboles de prédicat (p, q, est_pair, ...) appliqués à des termes
  - Connecteurs ($\neg$, $\vee$, $\wedge$, $\Rightarrow$)
  - Quantificateurs
    - universel $\forall x.\ A$
    - existentiel $\exists x.\ A$

Exemple : $\exists x.\ (porte(X, chapeau) \Rightarrow \forall Y.\ porte(Y, chapeau))$

Comparaison

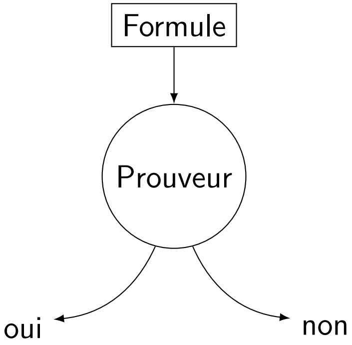
Logique propositionnelle :

- ⊖ peu expressif
  (même si encodages possibles)
- ⊕ décidable
  Ensemble des théorèmes co-NP complet

Logique du premier ordre :

- ⊕ universel
  peut exprimer des logiques plus riches
- ⊖ prouvabilité non décidable
  Ensemble des théorèmes récursivement énumérable

Prouveur automatique



Comment prouver automatiquement une formule ?

Preuve directe :

$(A_1 \wedge \cdots \wedge A_n) \Rightarrow G$

- Preuve en avant : part des A_i et infère de nouvelles formules jusqu'à obtenir G
- Preuve en arrière : part de G et applique des règles en fermant avec les A_i

Preuve par réfutation :

- Pour prouver A, on part de $\neg A$ et on essaie de trouver une contradiction
- Note : si A vaut $(A_1 \wedge \cdots \wedge A_n) \Rightarrow G$ cela revient à trouver une contradiction dans $\{A_1, \ldots, A_n, \neg G\}$

Quelle structure pour la recherche de preuve ?

Formule = arbre (avec lieurs)

Travailler directement avec cette structure arborescente ?

- Pas très efficace en pratique

Mieux :

- Transformer les formules en une structure plus facilement manipulable
  Par exemple listes, ensembles, etc.

Transformation de formules

Forme normale négative :

- on plonge les négations à l'intérieur des formules
- on élimine les doubles négations

$$\begin{aligned} \neg\neg A &\rightsquigarrow A \\ \neg(A \wedge B) &\rightsquigarrow (\neg A) \vee (\neg B) \\ \neg(\forall x.\ A) &\rightsquigarrow \exists x.\ (\neg A) \\ &\ldots \end{aligned}$$

Au final, seuls les prédicats peuvent avoir une négation

Transformation de formules

Forme prénexe :

- on fait remonter les quantifications en tête
- en faisant des renommages si besoin

$$\begin{array}{rl} (\forall x.\ A) \land B \rightsquigarrow \forall x.\ (A \land B) & \text{si } x \text{ n'est pas dans } B \\ (\forall x.\ A) \Rightarrow B \rightsquigarrow \exists x.\ (A \Rightarrow B) & \text{si } x \text{ n'est pas dans } B \\ \ldots & \end{array}$$

Au final, la formule est de la forme :

$$\mathcal{Q}_1 x_1.\ \ldots \mathcal{Q}_n x_n.\ A$$

avec $\mathcal{Q}_i \in \{\forall, \exists\}$ et A sans quantificateur

Transformation de formules

Skolémisation :

- on supprime les quantifications existentielles
- en introduisant de nouveaux symboles de fonction

$$
\begin{aligned}
&\forall x_1.\ \ldots \forall x_n.\ \exists y.\ A[x_1, \ldots, x_n, y] \\
&\quad \rightsquigarrow \forall x_1.\ \ldots \forall x_n.\ A[x_1, \ldots, x_n, f(x_1, \ldots, x_n)]
\end{aligned}
$$

Au final, la formule est de la forme :

$$\forall x_1.\ \ldots \forall x_n.\ A$$

avec A sans quantificateur

Transformation de formules

Forme normale conjonctive, ou clausale :

- on remplace les implications par des disjonctions (en poussant les négations si besoin)
- on distribue les disjonctions ($\vee$) sur les conjonctions ($\wedge$)

$$A \Rightarrow B \rightsquigarrow (\neg A) \vee B$$

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

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

Au final, la formule est de la forme :

$$\forall x_1. \ \ldots \forall x_n. \ C_1 \wedge \cdots \wedge C_m$$

où chaque C_i est de la forme $L_1 \vee \cdots L_{k_i}$
où chaque L_i est un prédicat ou sa négation

Exemple

On veut prouver

$$\exists Y.\, ((a(f(Y)) \wedge ((\exists Z.\, p(Z, g(Y))) \Rightarrow \neg a(Y))) \Rightarrow \forall X.\, \neg p(X, Y))$$

Exemple

On veut prouver

$$\exists Y.\, ((a(f(Y)) \wedge ((\exists Z.\, p(Z, g(Y))) \Rightarrow \neg a(Y))) \Rightarrow \forall X.\, \neg p(X, Y))$$

Négation :

$$\neg(\exists Y.\, ((a(f(Y)) \wedge ((\exists Z.\, p(Z, g(Y))) \Rightarrow \neg a(Y))) \Rightarrow \forall X.\, \neg p(X, Y)))$$

Exemple

On veut prouver

$$\exists Y.\,((a(f(Y)) \wedge ((\exists Z.\,p(Z,g(Y))) \Rightarrow \neg a(Y))) \Rightarrow \forall X.\,\neg p(X,Y))$$

Négation :

$$\neg(\exists Y.\,((a(f(Y)) \wedge ((\exists Z.\,p(Z,g(Y))) \Rightarrow \neg a(Y))) \Rightarrow \forall X.\,\neg p(X,Y)))$$

Forme normale négative :

$$\forall Y.\,(a(f(Y)) \wedge ((\exists Z.\,p(Z,g(Y))) \Rightarrow \neg a(Y)) \wedge \exists X.\,p(X,Y))$$

Forme prénexe :

$$\forall Y.\ \exists X.\ \forall Z.\ (a(f(Y)) \wedge (p(Z, g(Y)) \Rightarrow \neg a(Y)) \wedge p(X, Y))$$

Forme prénexe :

$$\forall Y.\ \exists X.\ \forall Z.\ (a(f(Y)) \wedge (p(Z, g(Y)) \Rightarrow \neg a(Y)) \wedge p(X, Y))$$

Forme de Skolem :

$$\forall Y.\ \forall Z.\ (a(f(Y)) \wedge (p(Z, g(Y)) \Rightarrow \neg a(Y)) \wedge p(sk(Y), Y))$$

Forme prénexe :

$$\forall Y.\ \exists X.\ \forall Z.\ (a(f(Y)) \wedge (p(Z, g(Y)) \Rightarrow \neg a(Y)) \wedge p(X, Y))$$

Forme de Skolem :

$$\forall Y.\ \forall Z.\ (a(f(Y)) \wedge (p(Z, g(Y)) \Rightarrow \neg a(Y)) \wedge p(sk(Y), Y))$$

Forme normale conjonctive :

$$\forall Y.\ \forall Z.\ (a(f(Y)) \wedge (\neg p(Z, g(Y)) \vee \neg a(Y)) \wedge p(sk(Y), Y))$$

Forme normale conjonctive et clauses

Litéral : prédicat ou négation de prédicat

Clause : ensemble de prédicats

À chaque formule de la forme $L_1 \vee \cdots L_k$
on associe la clause $\{L_1; \cdots; L_k\}$

À chaque forme normale conjonctive

$$\forall x_1. \ \ldots \forall x_n. \ C_1 \wedge \cdots \wedge C_m$$

on associe l'ensemble de clauses correspondant aux C_i

Exemple

$$\forall Y.\ \forall Z.\ (a(f(Y)) \land (\neg p(Z, g(Y)) \lor \neg a(Y)) \land p(sk(Y), Y))$$

$$\{a(f(Y))\}$$
$$\{\neg p(Z, g(Y)); \neg a(Y)\}$$
$$\{p(sk(Y), Y)\}$$

Exemple

$$\forall Y.\ \forall Z.\ (a(f(Y)) \wedge (\neg p(Z, g(Y)) \vee \neg a(Y)) \wedge p(sk(Y), Y))$$

$$\begin{array}{cc}
\{a(f(Y))\} & a(f(Y)) \\
\{\neg p(Z, g(Y)); \neg a(Y)\} & \neg p(Z, g(Y)) \vee \neg a(Y) \\
\{p(sk(Y), Y)\} & p(sk(Y), Y)
\end{array}$$

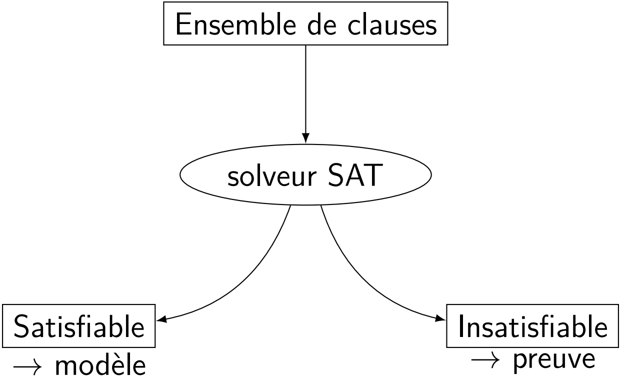
Exemple

$$\forall Y.\ \forall Z.\ (a(f(Y)) \wedge (\neg p(Z, g(Y)) \vee \neg a(Y)) \wedge p(sk(Y), Y))$$

$$\begin{array}{cc}
\{a(f(Y))\} & a(f(Y)) \\
\{\neg p(Z, g(Y)); \neg a(Y)\} & \neg p(Z, g(X)) \vee \neg a(X) \\
\{p(sk(Y), Y)\} & p(sk(V), V)
\end{array}$$

Satisfiabilité en logique propositionnelle

SAT



modèle = valuation 0 ou 1 des variables propositionnelles

Difficulté théorique

Problème NP-complet

- si algo polynomial, alors P=NP

Même si les clauses sont toutes de taille au plus trois

Mais...

Utilisations multiples

Depuis les années 90 (GRASP et Chaff), efficace en pratique

- applications industrielles

Big data

Solution (négative) au problème des triplets pythagoriciens :

- Séparer $\mathbb{N}$ en deux ensembles disjoints N_1 et N_2
- tel que ni N_1 ni N_2 ne contient un triplet pythagoricien (a, b, c) tel que $a^2 + b^2 = c^2$

preuve utilisant un SAT solver : 200 TB ! ! !

Utilisation des solveurs SAT

- Encodage du problème CNF
  - Représente le problème comme une instance de SAT
- Utilisation du SAT solveur comme oracle
  - boîte noire
- Intégration du SAT solveur
  - boîte blanche
  - utilisation des assignements partiels
  - dialogue

Format d'entrée

DIMACS

- À chaque variable propositionnelle on associe un entier > 0
- si A est associée à i, $\neg A$ est représentée par $-i$
- clause $=$ suite de littéraux terminée par 0

p cnf n k
1 -2 3 0
3 4 0
2 -1 4 0
$\vdots$

n nombre de variables propositionnelles

Glucose

- Gilles Audemard (Lens) et Laurent Simon (Bordeaux)
- Basé sur Minisat
- Libre

Au-delà de SAT

Besoin de raisonner à un niveau d'abstraction supérieur

Preuve de programme :

- arithmétique
- structures de données
- ...
- logique du premier ordre

Satisfiabilité modulo théorie

Problème : logique du premier ordre indécidable

On se restreint à la satisfiabilité dans une théorie donnée de formules sans quantificateurs
Décidable ou non en fonction de la théorie

Exemple

Théorie : arithmétique

$$
\begin{aligned}
& x \leq 3 \\
& \wedge (2x > 8 \vee x + y = 3) \\
& \wedge (y < -2 \vee x + y \neq 3)
\end{aligned}
$$

$$x_1 \wedge (\neg x_2 \vee x_3) \wedge (\neg x_4 \vee \neg x_3)$$

Coopération

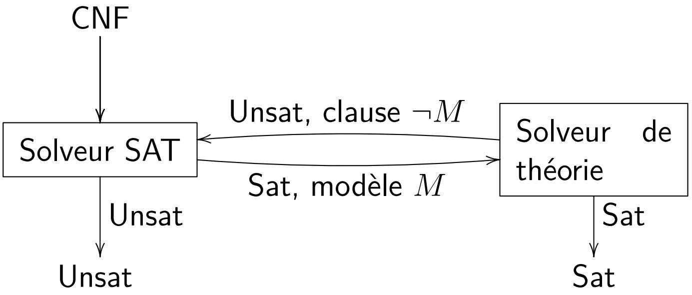
Solveur SAT

- gère la partie « logique » de la preuve

Solveur de théorie

- Entrée : conjonction de littéraux satisfaits
  (= modèle partiel)
- Sortie : satisfiable ou non pour la théorie

Schéma général



Quelles théories ?

Théorie = ensemble des formules qui restreignent les modèles à considérer

Le schéma marche pour n'importe quelle théorie

En pratique, certaines théories ont un intérêt, notamment pour la vérification de programme

Exemples de théories

- QF_UF : symboles de fonction non-interprétés
- (QF_)LIA : Arithmétique entière linéaire
  - pas de multiplication entre variables
  - arithmétique de Presburger, QF décidable
- (QF_)NIA : Arithmétique entière non linéaire
  - non décidable, même sans quantificateurs
- (QF_)NRA : Arithmétique réelle
  - QF décidable
- QF_AX : Tableaux
  - `read(write(`a`,` p`,` v`),` p`) =` v
    `read(write(`a`,` p`,` v`),` q`) =` v si $p \neq q$

Format d'entrée

SMTLIB

- On indique quelle(s) théorie(s) on utilise :
  ```
  (set-logic QF_LIA)
  ```
- On déclare les symboles :
  ```
  (declare-const x Int)
  ```
- On déclare les clauses :
  ```
  (assert (or (< y (-2)) (not (= (+ x y) 3))))
  ```
- On demande à résoudre le problème
  ```
  (check-sat)
  ```
- On quitte le solveur
  ```
  (exit)
  ```

Solveurs SMT

- Z3
  - Microsoft research (Leonardo De Moura, Nikolaj Bjørner, . . .)
- Alt-ergo
  - LRI (Orsay) (Sylvain Conchon, Évelyne Contejean, . . .)
- CVC4
  - Stanford University et University of Iowa (Cesare Tinelli, Clark Barrett, Andrew Reynolds,. . .)

Limites de l'approche SMT

Possibilité de donner des axiomes quantifiés en SMT

- remplacés par des instances bien choisie

Mais peu efficace

Passer à la logique du premier ordre complète

Résolution

Résolution propositionnelle :

$$\text{Resolution}\ \frac{P \vee C \qquad \neg P \vee D}{C \vee D}$$

À partir de l'ensemble de clauses de départ

- on essaie de dériver la clause vide $\square$
- auquel cas l'ensemble de départ était insatisfiable

Passage au premier ordre

Règle d'instanciation :

$$\text{Instantiation}\ \frac{C}{\sigma C}$$

σ substitution des variables de C par des termes

Quand faire les instanciations ?

- le minimum nécessaire pour pouvoir appliquer la règle de résolution

Résolution au premier ordre

$$\text{Resolution}\ \frac{P \vee C \qquad \neg Q \vee D}{\sigma(C \vee D)}$$

σ est l'unificateur le plus général de P et Q,
c'est-à-dire la substitution la plus générale telle que $\sigma P = \sigma Q$

Résolution au premier ordre

$$\text{Resolution}\ \frac{P \vee C \qquad \neg Q \vee D}{\sigma(C \vee D)}$$

σ est l'unificateur le plus général de P et Q,
c'est-à-dire la substitution la plus générale telle que $\sigma P = \sigma Q$

On a aussi besoin de

$$\text{Factoring}\ \frac{P \vee Q \vee C}{\sigma(P \vee C)}$$

Complétude

Résolution réfutationnellement complète :

- si A provable
  alors $\square$ peut être obtenue à partir de CNF$(\neg A)$

Complétude

Résolution réfutationnellement complète :

- si A provable
  alors $\Box$ peut être obtenue à partir de CNF($\neg A$)

Si A non prouvable :

- Peut s'arrêter après avoir généré toutes les clauses possibles
  - l'ensemble de clauses était satisfiable
- Peut ne jamais s'arrêter

En pratique, besoin de restreindre les applications de Resolution

Traitement de l'égalité

Equality Resolution $\dfrac{u \neq v \vee R}{\sigma(R)}\ \sigma = mgu(u, v)$

Negative Superposition $\dfrac{s = t \vee S \qquad u \neq v \vee R}{\sigma(u[t]_{\mathfrak{p}} \neq v \vee S \vee R)}$[1]

Positive Superposition $\dfrac{s = t \vee S \qquad u = v \vee R}{\sigma(u[t]_{\mathfrak{p}} = v \vee S \vee R)}$[1]

Equality Factoring $\dfrac{s = t \vee u = v \vee R}{\sigma(t \neq v \vee u = v \vee R)}\ \sigma = mgu(s, u)$

Avec des restrictions sur l'application des règles

1. $\sigma = mgu(u_{|\mathfrak{p}}, s)$

Format d'entrée

TPTP

Formula	syntaxe TPTP
$\neg p(X)$	`~p(X)`
$A \vee B$	`A | B`
$A \wedge B$	`A & B`
$A \Rightarrow B$	`A => B`
$\forall x.\ A$	`![X] : A`
$\exists x.\ A$	`?[X] : A`

```
fof(name, role, formula).
```

`role` : axiome, conjecture, lemme, etc.

Prouveur du premier ordre

- Vampire
  - U. de Manchester (Andrei Voronkov, Giles Reger, etc.)
- E
  - DHBW Stuttgart (Stefan Schulz)

Au delà de la preuve automatique

Prouveurs automatiques ne sont pas capable de tout prouver en pratique

- besoin de construire certaines preuves avec l'aide d'un utilisateur

Besoin d'un cadre rigoureux dans lequel construire la preuve

- déduction naturelle

Déduction naturelle

Idée : formaliser une façon « naturelle » de faire des raisonnements

Pour chaque connecteur/quantificateur, 2 règles :

- élimination : preuve en avant
  si on a A alors on a B
- introduction : preuve en arrière
  pour prouver A il suffit de prouver B

Conjonction

Élimination :
« On a A et B. On en déduit A (resp. B). »

$$\wedge\text{-e}_g \; \frac{A \wedge B}{A} \qquad\qquad \wedge\text{-e}_d \; \frac{A \wedge B}{B}$$

Conjonction

Élimination :
« On a A et B. On en déduit A (resp. B). »

$$\wedge\text{-e}_g \; \frac{A \wedge B}{A} \qquad\qquad \wedge\text{-e}_d \; \frac{A \wedge B}{B}$$

Introduction :
« Pour montrer A et B, montrons A, et montrons B »

$$\wedge\text{-i} \; \frac{A \qquad B}{A \wedge B}$$

Implication

Élimination :
« A implique B, or on a A, donc on a B. »

$$\Rightarrow\text{-e}\ \frac{A \Rightarrow B \qquad A}{B}$$

Modus ponens

Implication

Élimination :

« A implique B, or on a A, donc on a B. »

$$\Rightarrow\text{-e}\ \frac{A \Rightarrow B \qquad A}{B}$$

Modus ponens

« Supposons A, montrons B. »

$$\Rightarrow\text{-i}\ \frac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \Rightarrow B}$$

Exemple

$$\wedge\text{-i}\ \dfrac{\wedge\text{-e}_d\ \dfrac{A \wedge B}{B} \qquad \wedge\text{-e}_g\ \dfrac{A \wedge B}{A}}{B \wedge A}$$

Exemple

$$\dfrac{\wedge\text{-i}\ \dfrac{\wedge\text{-e}_d\ \dfrac{[A \wedge B]}{B} \qquad \wedge\text{-e}_g\ \dfrac{[A \wedge B]}{A}}{B \wedge A}}{(A \wedge B) \Rightarrow (B \wedge A)}\ \Rightarrow\text{-i}$$

Présentation à l'aide de séquents

Rendre explicite les axiomes utilisés dans la preuve

Jugements de la forme $A_1, \ldots, A_n \vdash B$
$A_1, \ldots, A_n$ axiomes, B conclusion

Présentation à l'aide de séquents

Rendre explicite les axiomes utilisés dans la preuve

Jugements de la forme $A_1, \ldots, A_n \vdash B$
$A_1, \ldots, A_n$ axiomes, B conclusion

Nouvelle présentation pour l'introduction de l'implication :

$$\Rightarrow\text{-i}\ \frac{A_1, \ldots, A_n, A \vdash B}{A_1, \ldots, A_n \vdash A \Rightarrow B}$$

Présentation à l'aide de séquents

Rendre explicite les axiomes utilisés dans la preuve

Jugements de la forme $A_1, \ldots, A_n \vdash B$
$A_1, \ldots, A_n$ axiomes, B conclusion

Nouvelle présentation pour l'introduction de l'implication :

$$\Rightarrow\text{-i}\ \frac{A_1, \ldots, A_n, A \vdash B}{A_1, \ldots, A_n \vdash A \Rightarrow B}$$

Comment fermer une preuve ?

- règle d'axiome

$$\widehat{\vdash}\ \frac{}{\Gamma, A \vdash A}$$

Exemple revisité

$$
\Rightarrow\text{-i}\ \dfrac{\wedge\text{-i}\ \dfrac{\wedge\text{-e}_d\ \dfrac{\widehat{\vdash}\ \dfrac{}{A \wedge B \vdash A \wedge B}}{A \wedge B \vdash B} \qquad \wedge\text{-e}_g\ \dfrac{\widehat{\vdash}\ \dfrac{}{A \wedge B \vdash A \wedge B}}{A \wedge B \vdash A}}{A \wedge B \vdash B \wedge A}}{\vdash (A \wedge B) \Rightarrow (B \wedge A)}
$$

Quantification universelle

Élimination :
« Pour tout x, A(x) est vrai. En particulier A(t) est vrai. »

$$\forall\text{-e}\ \frac{\Gamma \vdash \forall x.\ A[x]}{\Gamma \vdash A[t]}$$

Quantification universelle

Élimination :
« Pour tout x, A(x) est vrai. En particulier A(t) est vrai. »

$$\forall\text{-e}\ \frac{\Gamma \vdash \forall x.\ A[x]}{\Gamma \vdash A[t]}$$

Introduction :
« Soit x quelconque. Montrons A(x). »

$$\forall\text{-i}\ \frac{\Gamma \vdash A[x]}{\Gamma \vdash \forall x.\ A[x]}\ x \text{ non libre dans } \Gamma$$

Faux

Élimination :
« De faux on peut déduire n'importe quoi. »

$$\bot\text{-e}\ \frac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

Faux

Élimination :
« De faux on peut déduire n'importe quoi. »

$$\bot\text{-e}\ \frac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

Introduction :
Pas possible a priori

- pas de règle d'introduction

Négation

$\neg A$ encodable par $A \Rightarrow \bot$

Règles associées :

$$\neg\text{-e}\ \frac{\Gamma \vdash \neg A \qquad \Gamma \vdash A}{\Gamma \vdash \bot} \qquad\qquad \neg\text{-i}\ \frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}$$

Exemple

$$(\forall x.\ \neg P(x)) \Rightarrow \neg\forall x.\ P(x)$$

Exemple

$$(\forall x.\ \neg P(x)) \Rightarrow \neg\forall x.\ P(x)$$

$$
\Rightarrow\text{-i}\ \dfrac{\neg\text{-i}\ \dfrac{\neg\text{-e}\ \dfrac{\forall\text{-e}\ \dfrac{\widehat{\vdash}\ \dfrac{}{\Gamma \vdash \forall x.\ P(x)}}{\Gamma \vdash \neg P(t)} \qquad \forall\text{-e}\ \dfrac{\widehat{\vdash}\ \dfrac{}{\Gamma \vdash \forall x.\ P(x)}}{\Gamma \vdash P(t)}}{\Gamma \vdash \bot}}{\forall x.\ \neg P(x) \vdash \neg\forall x.\ P(x)}}{\vdash (\forall x.\ \neg P(x)) \Rightarrow \neg\forall x.\ P(x)}
$$

avec $\Gamma = \forall x.\ \neg P(x), \neg\forall x.\ P(x)$

Disjonction

Introduction :

« Pour montrer qu'on a A ou B, montrons qu'on a A. »

$$\vee\text{-i}_g \; \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \vee\text{-i}_d \; \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

Disjonction

Élimination :

« On a A ou B. On distingue les cas. Si on a A, on a C. Si on a B, on a C. Donc dans tous les cas on a C. »

$$\vee\text{-e}\ \frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}$$

Introduction :

« Pour montrer qu'on a A ou B, montrons qu'on a A. »

$$\vee\text{-i}_g\ \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \vee\text{-i}_d\ \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

Cas particulier d'élimination

$$\vee\text{-e}\ \dfrac{\widehat{\vdash}\ \dfrac{}{\Gamma, A \vee B \vdash A \vee B} \qquad \Gamma, A \vee B, A \vdash C \qquad \Gamma, A \vee B, B \vdash C}{\Gamma, A \vee B \vdash C}$$

Règle dérivée :

$$\vee\text{-d}\ \dfrac{\Gamma, A \vee B, A \vdash C \qquad \Gamma, A \vee B, B \vdash C}{\Gamma, A \vee B \vdash C}$$

Exemple

$$(A \vee B) \Rightarrow (B \vee A)$$

$$
\Rightarrow\text{-i}\;\dfrac{\vee\text{-d}\;\dfrac{\vee\text{-i}_d\;\dfrac{\widehat{\vdash}\;\dfrac{}{A \vee B, A \vdash A}}{A \vee B, A \vdash B \vee A} \qquad \vee\text{-i}_g\;\dfrac{\widehat{\vdash}\;\dfrac{}{A \vee B, B \vdash B}}{A \vee B, B \vdash B \vee A}}{A \vee B \vdash B \vee A}}{\vdash (A \vee B) \Rightarrow (B \vee A)}
$$

Quantification existentielle

Introduction :
« Pour montrer qu'il existe un x tel que A[x], montrons qu'on a A[t] pour un certain t. »

$$\exists\text{-i}\ \frac{\Gamma \vdash A[t]}{\Gamma \vdash \exists x.\ A[x]}$$

Quantification existentielle

Élimination :

« Il existe x tel que A(x). Si pour un x quelconque, en supposant A(x) on a C, alors dans tous les cas on a C. »

$$\exists\text{-e}\ \frac{\Gamma \vdash \exists x.\ A[x] \qquad \Gamma, A[x] \vdash C}{\Gamma \vdash C}\ x \text{ non libre dans } \Gamma, C$$

Introduction :

« Pour montrer qu'il existe un x tel que A[x], montrons qu'on a A[t] pour un certain t. »

$$\exists\text{-i}\ \frac{\Gamma \vdash A[t]}{\Gamma \vdash \exists x.\ A[x]}$$

Cas particulier d'élimination

$$\exists\text{-e}\ \dfrac{\widehat{\vdash}\ \dfrac{}{\Gamma, \exists x.\ A[x] \vdash \exists x.\ A[x]} \qquad \Gamma, \exists x.\ A[x], A[x] \vdash C}{\Gamma, \exists x.\ A[x] \vdash C}$$

Règle dérivée :

$$\exists\text{-d}\ \dfrac{\Gamma, \exists x.\ A[x], A[x] \vdash C}{\Gamma, \exists x.\ A[x] \vdash C}\ x \text{ non libre dans } \Gamma, C$$

Exemple

$$(\exists x.(P(f(x)) \wedge Q(x))) \Rightarrow \exists x.\ P(x)$$

$$\Rightarrow\text{-i}\ \dfrac{\exists\text{-d}\ \dfrac{\exists\text{-i}\ \dfrac{\wedge\text{-e}_g\ \dfrac{\overset{\frown}{\vdash}\ \dfrac{}{\exists x.(P(f(x)) \wedge Q(x)), P(f(x)) \wedge Q(x) \vdash P(f(x)) \wedge Q(x)}}{\exists x.(P(f(x)) \wedge Q(x)), P(f(x)) \wedge Q(x) \vdash P(f(x))}}{\exists x.(P(f(x)) \wedge Q(x)), P(f(x)) \wedge Q(x) \vdash \exists x.\ P(x)}}{\exists x.(P(f(x)) \wedge Q(x)) \vdash \exists x.\ P(x)}}{\vdash (\exists x.(P(f(x)) \wedge Q(x))) \Rightarrow \exists x.\ P(x)}$$

Correspondance de Curry–Howard–De Bruyn

Preuve en déduction naturelle = programme fonctionnel
Formule prouvée = type du programme

Correspondance de Curry–Howard–De Bruyn

Preuve en déduction naturelle = programme fonctionnel
Formule prouvée = type du programme

$A \Rightarrow B \equiv$ `'a -> 'b`

$$\Rightarrow\text{-e}\ \frac{\Gamma \vdash \texttt{f} : A \Rightarrow B \qquad \Gamma \vdash \texttt{g} : A}{\Gamma \vdash \texttt{f g} : B}$$

$$\Rightarrow\text{-i}\ \frac{\Gamma, \texttt{x} : A \vdash \texttt{t} : B}{\Gamma \vdash \texttt{fun x -> t} : A \Rightarrow B}$$

$A \wedge B \equiv$ `'a * 'b`

$$\wedge\text{-e}_g \; \frac{\Gamma \vdash \texttt{p} \;:\; A \wedge B}{\Gamma \vdash \texttt{fst p} \;:\; A} \qquad \wedge\text{-e}_d \; \frac{\Gamma \vdash \texttt{p} \;:\; A \wedge B}{\Gamma \vdash \texttt{snd p} \;:\; B}$$

$$\wedge\text{-i} \; \frac{\Gamma \vdash \texttt{u} \;:\; A \qquad \Gamma \vdash \texttt{v} \;:\; B}{\Gamma \vdash \texttt{(u, v)} \;:\; A \wedge B}$$

$A \wedge B \equiv$ `'a * 'b`

$$\wedge\text{-e}_g \; \frac{\Gamma \vdash \texttt{p} \;:\; A \wedge B}{\Gamma \vdash \texttt{fst p} \;:\; A} \qquad \wedge\text{-e}_d \; \frac{\Gamma \vdash \texttt{p} \;:\; A \wedge B}{\Gamma \vdash \texttt{snd p} \;:\; B}$$

$$\wedge\text{-i} \; \frac{\Gamma \vdash \texttt{u} \;:\; A \qquad \Gamma \vdash \texttt{v} \;:\; B}{\Gamma \vdash \texttt{(u, v)} \;:\; A \wedge B}$$

$A \vee B \equiv$ `Left of 'a | Right of 'b`

$$\vee\text{-e} \; \frac{\Gamma \vdash \texttt{s} \;:\; A \vee B \qquad \Gamma, \texttt{x} : A \vdash \texttt{u} \;:\; C \qquad \Gamma, \texttt{y} : B \vdash \texttt{v} \;:\; C}{\Gamma \vdash \texttt{match s with Left x -> u | Right y -> v} \;:\; C}$$

$$\vee\text{-i}_g \; \frac{\Gamma \vdash \texttt{u} \;:\; A}{\Gamma \vdash \texttt{Left u} \;:\; A \vee B} \qquad \vee\text{-i}_d \; \frac{\Gamma \vdash \texttt{v} \;:\; B}{\Gamma \vdash \texttt{Right v} \;:\; A \vee B}$$

Coq

- Assistant de preuve développé par Inria depuis les années 1980
- Basé sur le calcul des constructions inductives
- qui est une extension du calcul des constructions
- qui est une extension de la déduction naturelle à une logique plus riche (ordre supérieur, types dépendants)

Principes

Pour montrer un théorème,
on va construire le programme associé

Mais pas directement

- langage de tactiques

À chaque règle de la déduction naturelle

- une tactique

Tactiques

$\wedge$-e$_g$: `apply` H
$\wedge$-e$_d$: `apply` H
$\wedge$-i : `split`

$\Rightarrow$-e : `apply` H
$\Rightarrow$-i : `intro`

$\vee$-d : `destruct` H
$\vee$-i$_g$: `left`
$\vee$-i$_d$: `right`

$\forall$-e : `apply` H
$\forall$-i : `intro`

$\exists$-d : `destruct` H
$\exists$-i : `exists` t

Autres tactiques

`unfold` : déplie une définition

`rewrite` H : si H est de type $u = v$, remplace les occurences de u par v

`omega` : procédure de décision pour l'arithmétique de Presburger

`lia` : linear integer arithmetique

`ring` : normalisation de polynômes sur une structure d'anneau