Logique du premier ordre : introduction à la notion de système de preuve

Prérequis : Maitriser la syntaxe des formules du premier ordre, avoir une compréhension limpide des notions d'interprétation, de modèle, de validité, de cohérence et de conséquence logique en logique du premier ordre (aussi appelée logique des prédicats).

Objectif : Comprendre ce qu'est une règle d'inférence, une preuve formelle, un système de preuve complet, les conséquences concrètes de la complétude et de l'indécidabilité de la logique du premier ordre.

Non objectif : Le but n'est pas d'être capable de réaliser des preuves formelles, ni de programmer des systèmes de preuve, ni même d'utiliser des prouveurs existants.

Par Olivier Bailleux, maître de conférences HDR en informatique.

olivier.bailleux@u-bourgogne.fr

Version de juillet 2017.

UNIVERSITÉ DE BOURGOGNE

Introduction

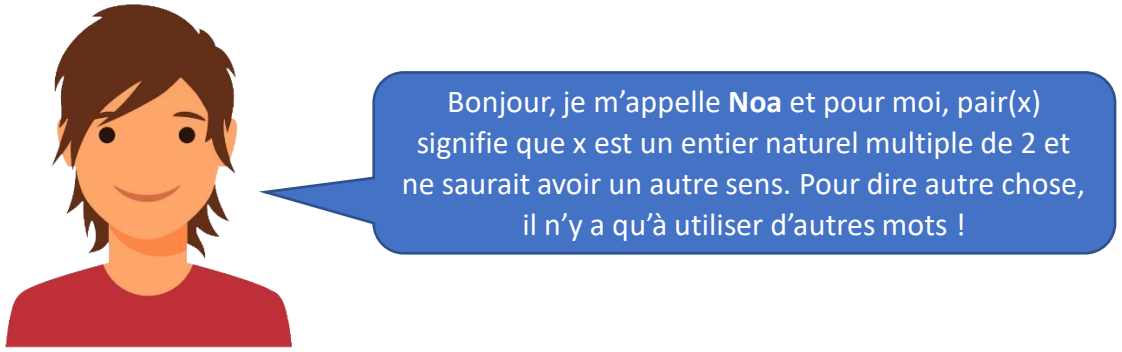
Les identificateurs de couleur rouge sont des *symboles de prédicats* et ceux de couleur verte sont des *symboles fonctionnels*.

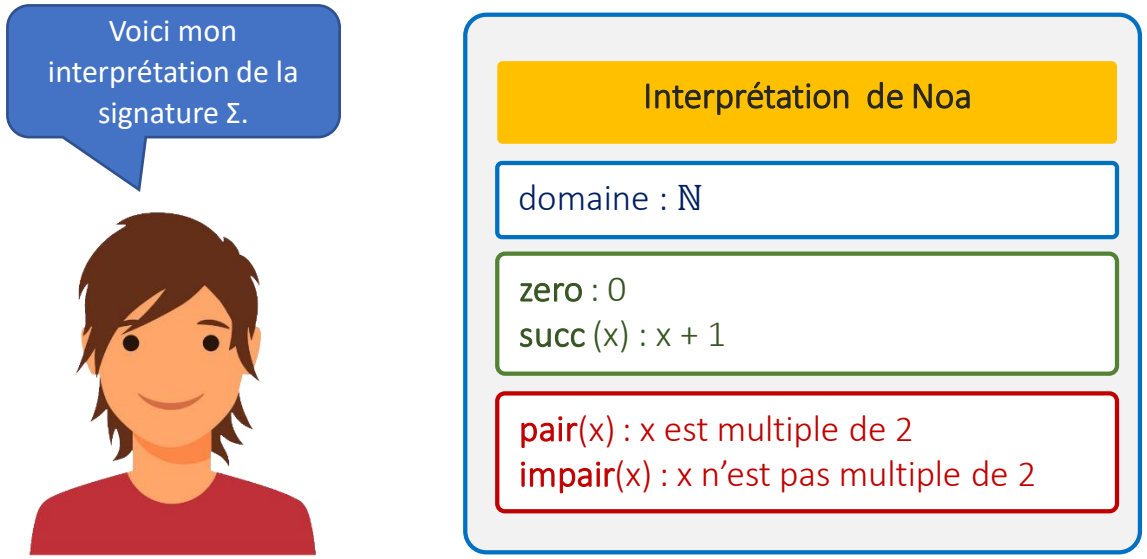
Considérons les trois formules suivantes comme des connaissances déjà établies concernant les nombres pairs et impairs.

K1	$\text{pair}(\text{zero})$	0 est un nombre pair
K2	$\forall x\ [\text{pair}(x) \rightarrow \text{impair}(\text{succ}(x))]$	Le successeur de tout nombre pair est impair.
K3	$\forall x\ [\text{impair}(x) \rightarrow \text{pair}(\text{succ}(x))]$	Le successeur de tout nombre impair est pair

Ces formules énoncent des propriétés des nombres pairs et impairs dès lors qu'on donne aux symboles fonctionnels **zero** et **succ** et aux symboles de prédicat **pair** et **impair** un sens particulier et qu'on considère que les valeurs possibles des variables et des termes fonctionnels sont des entiers naturels, i.e., 0, 1, 3, etc. Ces conventions représentent une *interprétation* particulière de la signature Σ = ({pair/1, impair/1}, {zero/0, succ/1}).

Du point de vue de **Noa**, qui s'intéresse aux propriétés des entiers naturels pairs et impairs, cette signature s'interprète d'une manière unique que nous appellerons *interprétation standard.*





Pour Noa, il n'y a qu'une seule façon de « comprendre » ce que disent les trois formules K1, K2, et K3, qui est basée sur cette interprétation standard pour laquelle ces trois formules énoncent des propriétés connues des nombres pairs et impairs. Ces trois formules sont *satisfaites* par cette interprétation standard

Introduction

Mais du point de vue de **Lucie**, qui est spécialiste de logique, la signature Σ a une infinité d'interprétations possibles et l'interprétation de Noa n'est qu'un cas particulier.

K1	$\text{pair}(\text{zero})$	0 est un nombre pair
K2	$\forall x\ [\text{pair}(x) \rightarrow \text{impair}(\text{succ}(x))]$	Le successeur de tout nombre pair est impair.
K3	$\forall x\ [\text{impair}(x) \rightarrow \text{pair}(\text{succ}(x))]$	Le successeur de tout nombre impair est pair

Bonjour, je m'appelle **Lucie** et pour moi, le symbole **pair** peut désigner une infinité de fonctions différentes, qui n'ont pas toutes les mêmes ensembles de départ et d'arrivée.

Voici deux exemples d'interprétations de la signature Σ qui, du point de vue de la logique, sont tout à fait légales.

Mais pourquoi aller imaginer des trucs aussi exotiques ?

Interprétation Lucie1

domaine : $\mathbb{N}$

zero : 0
succ (x) : x + 1

pair(x) : x est multiple de 2
impair(x) : x est multiple de 3

Interprétation Lucie2

domaine : $\mathbb{R}$

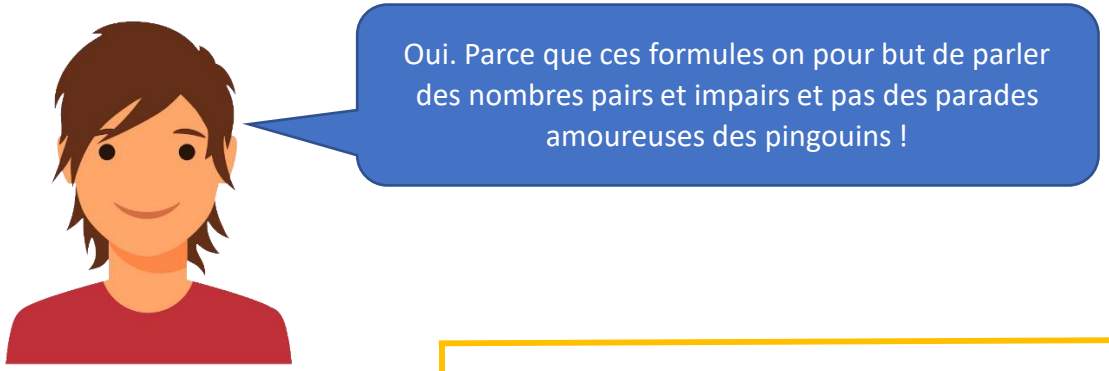
zero : 1
succ (x) : x + 1/x

pair(x) : x > 2
impair(x) : x > 3

Introduction

La signature Σ = ({pair/1, impair/1}, {zero/0, succ/1}) a une infinité d'interprétations possibles. Les interprétations Noa, Lucie1 et Lucie2 ne sont que trois exemples qui donnent des sens différents aux symboles fonctionnels et au symboles de prédicats.
On peut remarquer que l'interprétation Noa, aussi appelée interprétation standard, est la seule parmi ces trois exemples à satisfaire les formules K1, K2 et K3.

K1	$\text{pair}(\text{zero})$	0 est un nombre pair
K2	$\forall x\ [\text{pair}(x) \rightarrow \text{impair}(\text{succ}(x))]$	Le successeur de tout nombre pair est impair.
K3	$\forall x\ [\text{impair}(x) \rightarrow \text{pair}(\text{succ}(x))]$	Le successeur de tout nombre impair est pair



	K1	K2	K3
Noa	vrai	vrai	vrai
Lucie1	vrai	faux	faux
Lucie2	faux	faux	vrai

Interprétation de Noa

domaine : $\mathbb{N}$

zero : 0
succ (x) : x + 1

pair(x) : x est multiple de 2
impair(x) : x n'est pas multiple de 2

Interprétation Lucie1

domaine : $\mathbb{N}$

zero : 0
succ (x) : x + 1

pair(x) : x est multiple de 2
impair(x) : x est multiple de 3

Interprétation Lucie2

domaine : $\mathbb{R}$

zero : 1
succ (x) : x + 1/x

pair(x) : x > 2
impair(x) : x > 3

Introduction

En logique, pour faire une déduction, on a pas besoin de connaître le sens des symboles fonctionnels et des symboles de prédicats, parce-que la déduction devra être correcte *quelle que soit l'interprétation de ces symboles*.

Dans notre exemple, on veut montrer que la formule D est une *conséquence logique* de l'ensemble de formules {K1, K2, K3}, c'est à dire que toute interprétation qui satisfait les trois formules K1, K2 et K3 satisfait la formule D. On peut dire aussi que tout modèle de K1, K2 et K3 est un *modèle* de D, puisque toute interprétation satisfaisant une formule est, par définition, un modèle de cette formule.

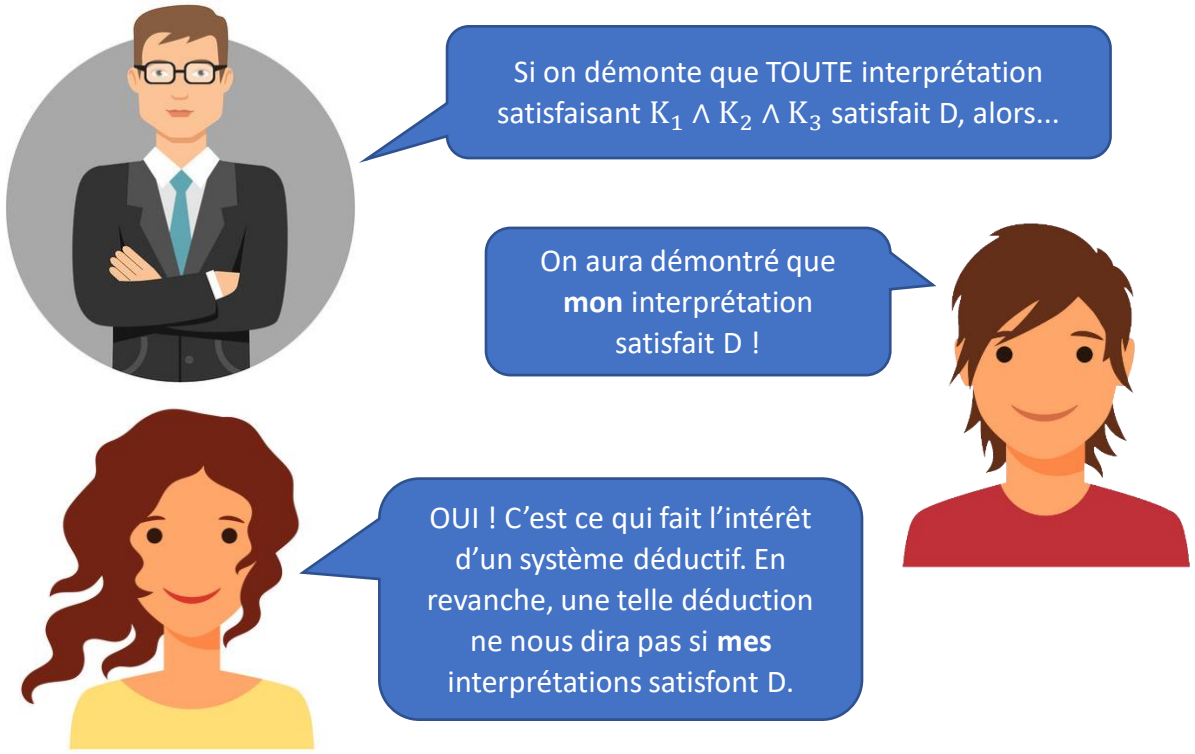
Cette propriété se note $\mathrm{K}_1 \wedge \mathrm{K}_2 \wedge \mathrm{K}_3 \vDash \mathrm{D}$ ou $\{\mathrm{K}_1, \mathrm{K}_2, \mathrm{K}_3\} \vDash \mathrm{D}$.

Dans la suite, on notera K la formule $\mathrm{K}_1 \wedge \mathrm{K}_2 \wedge \mathrm{K}_3$.

K1	$\mathrm{pair}(\mathrm{zero})$	0 est un nombre pair
K2	$\forall x\ [\mathrm{pair}(x) \rightarrow \mathrm{impair}(\mathrm{succ}(x))]$	Le successeur de tout nombre pair est impair.
K3	$\forall x\ [\mathrm{impair}(x) \rightarrow \mathrm{pair}(\mathrm{succ}(x))]$	Le successeur de tout nombre impair est pair

$\{\mathrm{K}_1, \mathrm{K}_2, \mathrm{K}_3\} \vDash \mathrm{D}$

D	$\forall x\ [\mathrm{pair}(x) \rightarrow \mathrm{pair}(\mathrm{succ}(\mathrm{succ}(x)))]$	Le successeur du successeur de tout nombre pair est pair.



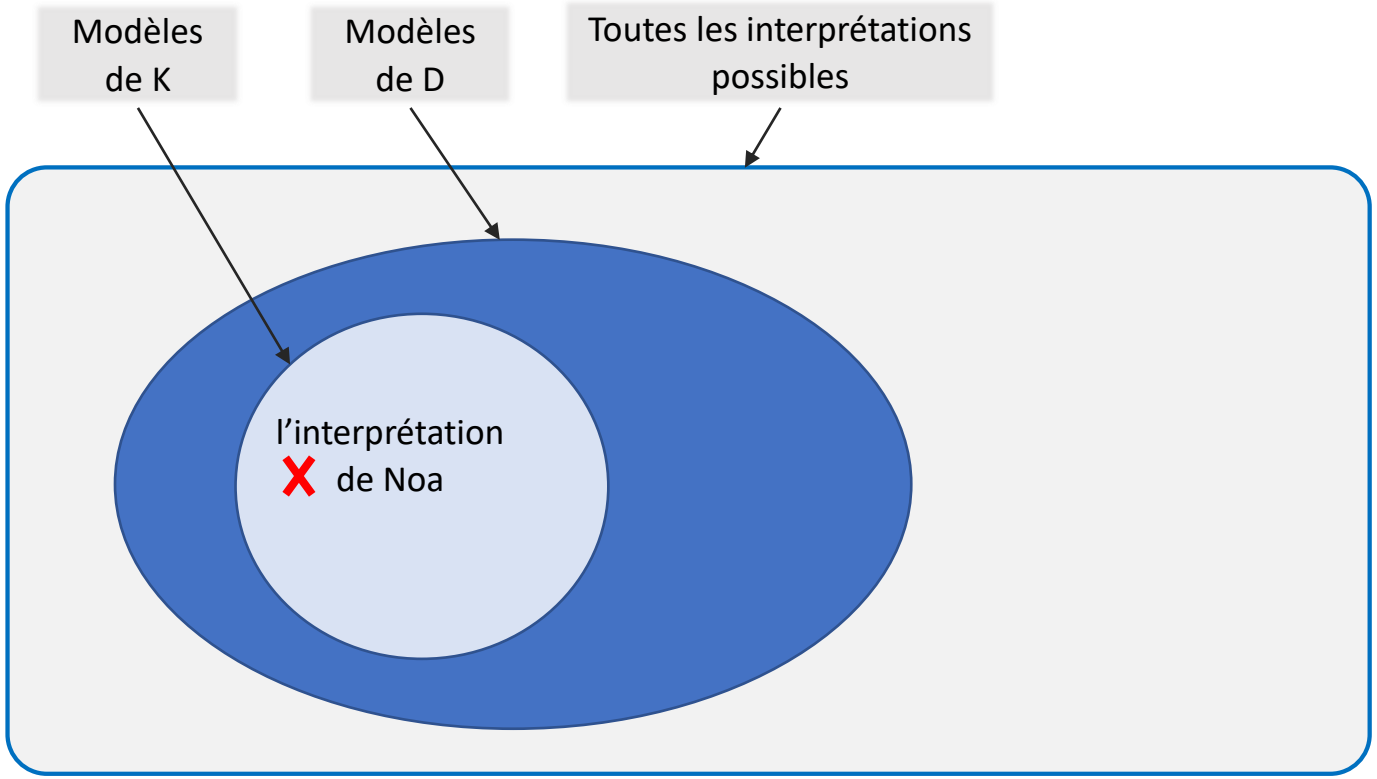
Introduction

Illustrons par un petit dessin le sens de la relation $K \vDash D$ que nous voulons prouver.

K1	$\text{pair}(\text{zero})$	0 est un nombre pair
K2	$\forall x\ [\text{pair}(x) \rightarrow \text{impair}(\text{succ}(x))]$	Le successeur de tout nombre pair est impair.
K3	$\forall x\ [\text{impair}(x) \rightarrow \text{pair}(\text{succ}(x))]$	Le successeur de tout nombre impair est pair

$$\mathrm{K} = \mathrm{K}_1 \wedge \mathrm{K}_2 \wedge \mathrm{K}_3 \vDash \mathrm{D}$$

D	$\forall x\ [\text{pair}(x) \rightarrow \text{pair}(\text{succ}(\text{succ}(x)))]$	Le successeur du successeur de tout nombre pair est pair.



On suppose que le fait que l'interprétation de Noa satisfait K est déjà établi. On veut montrer que tout modèle de K satisfait D, c'est à dire que l'ensemble des modèles de D inclus l'ensemble des modèles de K.

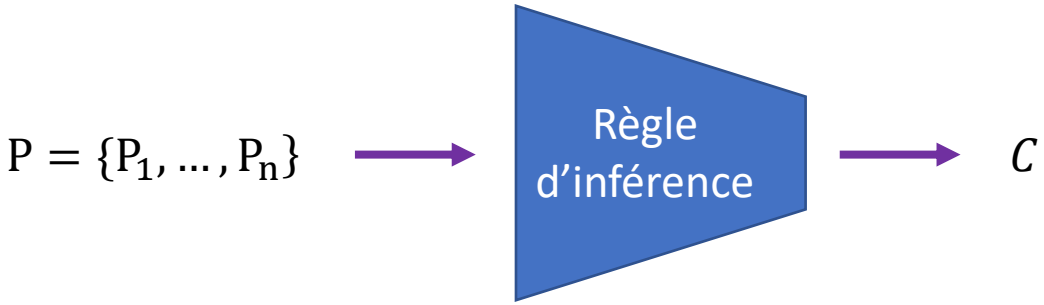
Comme on sait par ailleurs que les interprétations Lucie1 et Lucie2 ne sont pas des modèles de K, on les ignore. Elles ne sont pas concernées par les déductions réalisées à partir de K. Vous pouvez utiliser vos connaissances mathématiques pour situer ces deux interprétations sur le dessin.

(Il y en a une dans la partie bleu foncé et une dans la partie grise.)

Qu’est ce qu’une preuve formelle ?

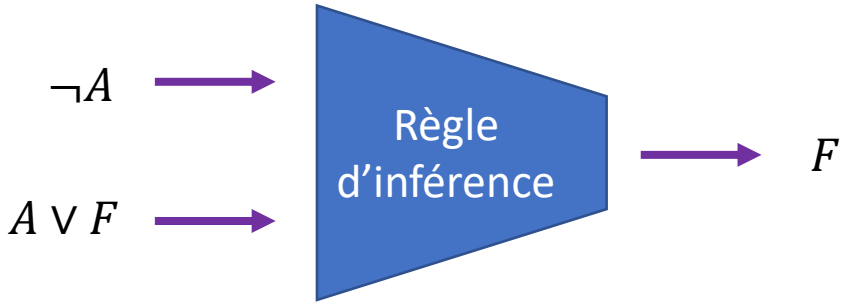
Règle d’inférence

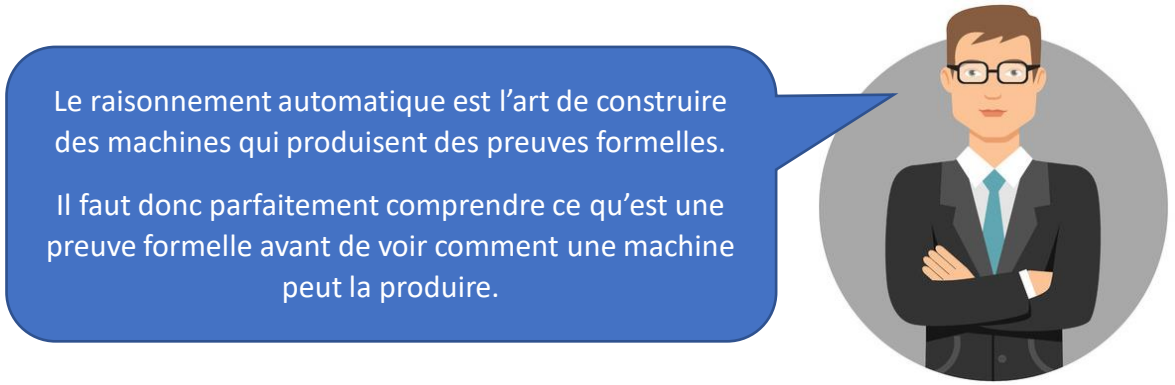
Une *règle d’inférence* est une règle pouvant être exécutée par une machine, permettant de produire, à partir d’un ensemble P de formules appelées *prémisses*, une nouvelle formule C appelée *conclusion,* qui est conséquence logique de P (en considérant que les variables libres des formules sont quantifiées universellement).



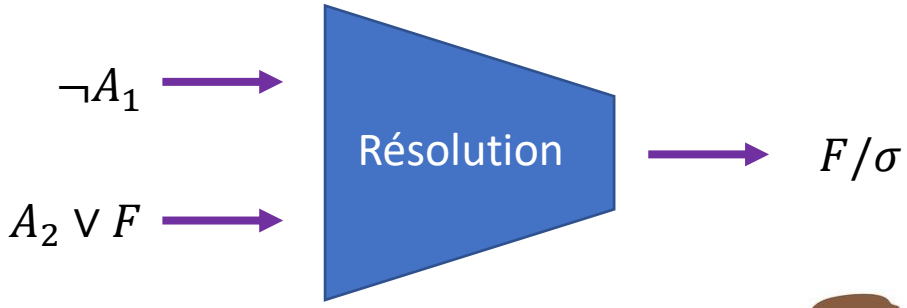
$$P_1 \wedge \cdots \wedge P_n \vDash C$$

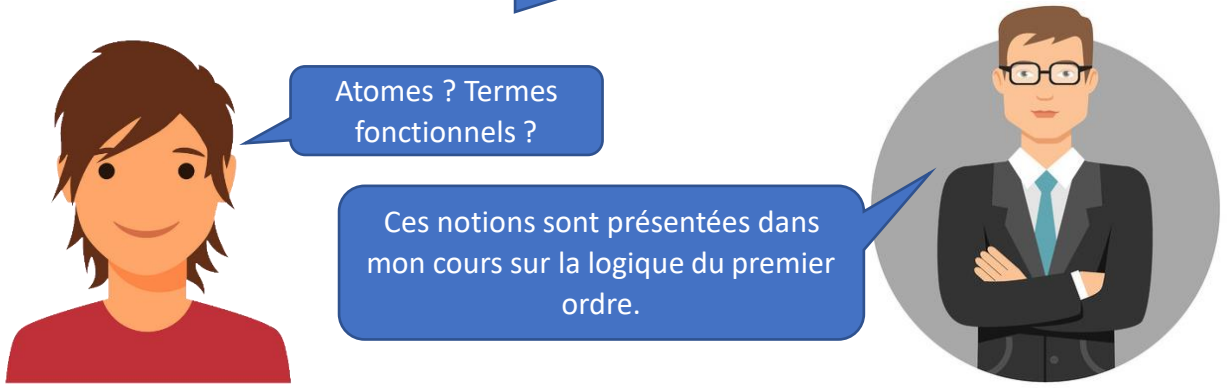
Voici un exemple de règle d’inférence :





Une variante de cette règle appelée *Résolution* accepte pour prémisses deux formules $\neg A_1$ et $\mathrm{A}_2 \vee F$ telles que $\neg A_1$ et A_2 sont deux atomes *unifiables*, c’est à dire qu’il existe une substitution σ des variables par des termes fonctionnels qui transforme A_1 et A_2 en un même atome A.

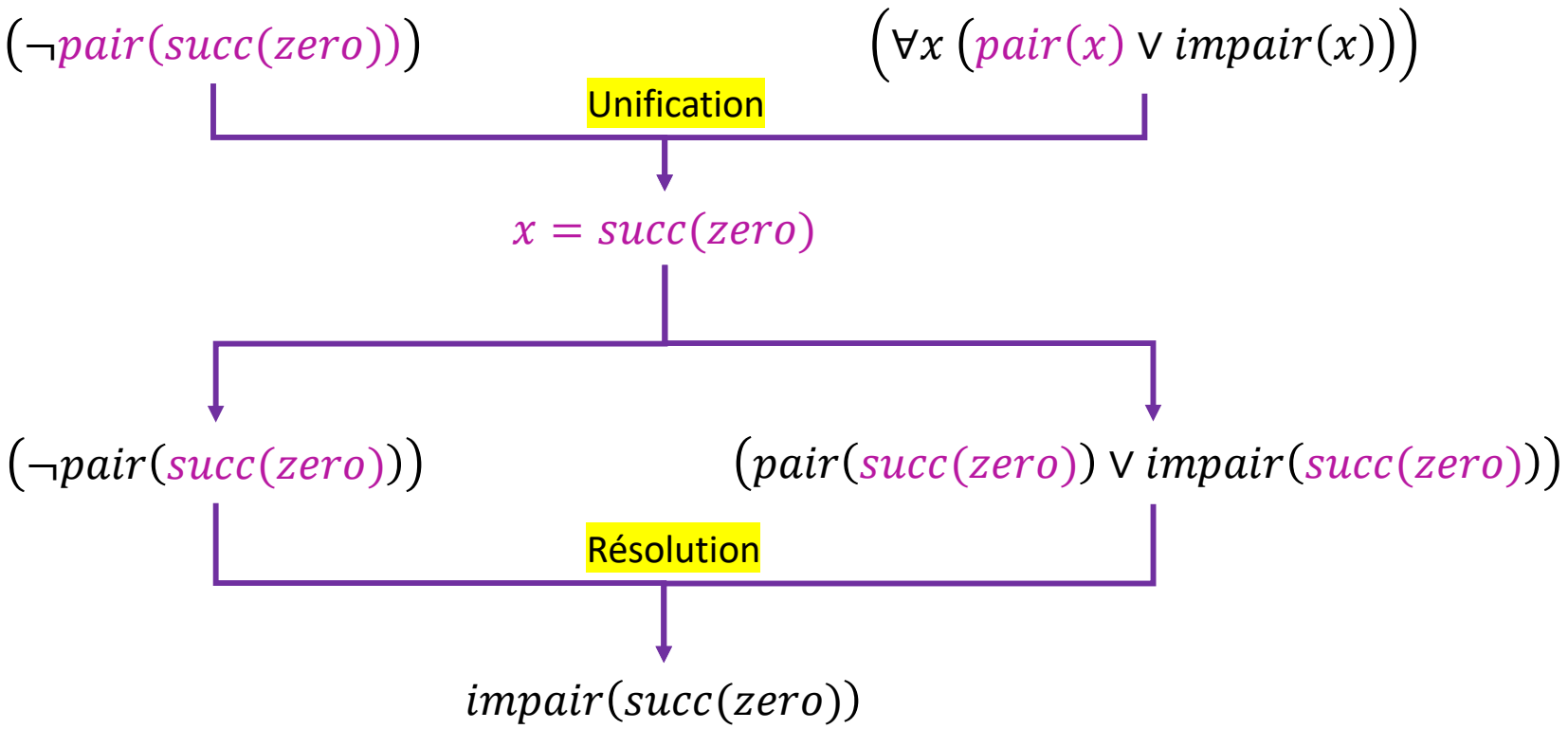




Qu’est ce qu’une preuve formelle ?

Règle d’inférence

Cet exemple a juste vocation à concrétiser la notion de règle d’inférence. Les deux transformations utilisées (unification et Résolution) sont facilement réalisables par des algorithmes.



Qu'est ce qu'une preuve formelle ?

Dérivation

Soit R un ensemble de règles d'inférence, K un ensemble de formules et D une formule. On appelle *dérivation* produisant D à partir de K toute suite de formules $H_1, \dots, H_n$ vérifiant les deux propriétés suivantes :

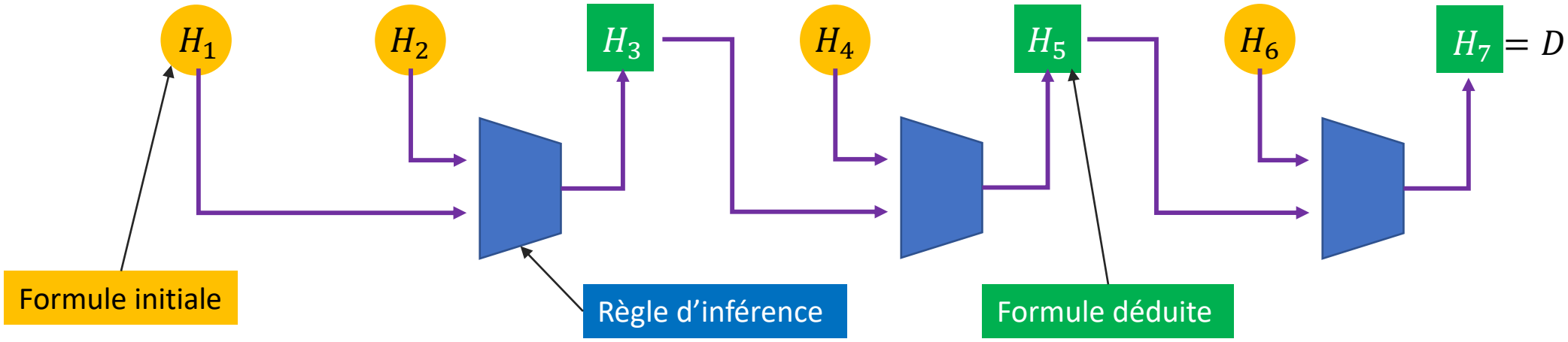
1. Chaque élément H_i est soit une formule de K, soit une formule produite à l'aide d'une règle de R avec comme prémisses des formules apparaissant avant H_i dans la suite, c'est-à-dire des formules de $\{H_1, \dots, H_{i-1}\}$.

2. $H_n = D$.

On applique des règles d'inférence à des formules initiales et / ou à des formules déjà déduites, donc la formule D est conséquence logique de l'ensemble K des formules initiales

Et en plus des formules initiales, il est possible d'utiliser des formules valides appelé axiomes logiques !

Exemple de dérivation d'une formule D à partir d'un ensemble de formules $K = \{H_1, H_2, H_4, H_6\}$



Qu'est ce qu'un système de preuve ?

Preuve formelle

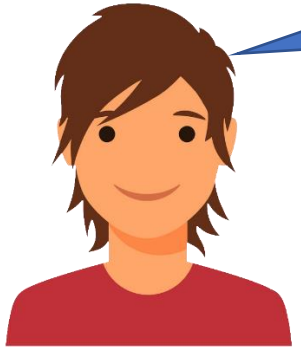
On appelle *système de preuve* un ensemble de règles d'inférence et de formules valides appelées axiomes logiques. Toute dérivation réalisée à l'aide d'un tel système est appelée *preuve formelle*.

Si K est l'ensemble des formules initiales et D la formule produite, alors on note $K \vdash D$. Comme les conclusions des règles d'inférences utilisées sont des conséquences logiques de leurs prémisses (sous réserve de quantifier universellement les variables libres), si $K \vdash D$ alors $K \models D$, autrement dit, D est conséquence logique de K.

Attendez attendez, c'est quoi cette histoire « d'axiomes logiques » ?

Si A est une formule valide, donc toujours vraie, et F une formule quelconque, alors tout modèle de F ∧ A est modèle de F donc toute conséquence logique de F ∧ A est conséquence logique de F.

On peut donc enrichir un système déductif en lui permettant d'utiliser des formules valides, appelées axiomes logiques comme prémisses de ses règles d'inférence pour augmenter ses capacités de déduction.



Bon, donc un système de preuve permet de produire des conséquences logiques.

si $K \vdash D$ alors $K \models D$

Mais existe-t-il des systèmes de preuves permettant de produire *toutes* les conséquences logiques ?

si $K \models D$ alors $K \vdash D$

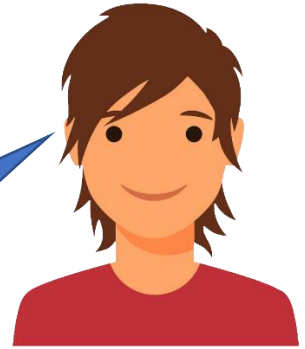
Un système de preuve est dit *complet* si et seulement si pour tout ensemble K de formules et toute formule D conséquence logique de K, il existe une preuve formelle $K \vdash D$ basée sur les règles d'inférences et les axiomes logiques de ce système de preuve.

En 1930, Kurt Gödel démontre qu'il existe des systèmes de preuve complets en logique du premier ordre.

C'est le théorème de *complétude* de Gödel.

Qu'est ce qu'un système de preuve ?

Système de preuve complet



Ah ! Très bien. Parce qu'on a toujours pas prouvé que le successeur du successeur d'un nombre pair est pair. C'est le moment de dégainer un de ces fameux « systèmes de preuve complet » !

Il existe plusieurs systèmes déductifs complets en logique du premier ordre, mais ils ne sont pas tous très « faciles » à utiliser. Un des plus simples n'utilise aucun axiome et juste une règle d'inférence, la Résolution vue précédemment.

Par contre, ce système comporte deux limitations :

Primo, ce système de preuve n'est complet que pour la *réfutation*, c'est à dire qu'il garantit de dériver la formule ⊥, c'est à dire faux si l'ensemble des formule de départ est incohérent. MAIS ce n'est pas un problème car prouver que $K \vDash D$ revient à prouver que $K \wedge \neg D \vDash \bot$.

Secundo, il faut que les formules initiales soient des *clauses*. Une clause est une disjonction d'atomes dont toutes le variables sont quantifiées par des quantificateurs universels placés en début de formule. MAIS ce n'est pas un problème car il existe des algorithmes permettant de convertir toute formule F en une clause équi-cohérente c'est à dire que cette clause est cohérente si et seulement si F est cohérente.

K1	$\text{pair}(\text{zero})$	0 est un nombre pair
K2	$\forall x\ [\text{pair}(x) \rightarrow \text{impair}(\text{succ}(x))]$	Le successeur de tout nombre pair est impair.
K3	$\forall x\ [\text{impair}(x) \rightarrow \text{pair}(\text{succ}(x))]$	Le successeur de tout nombre impair est pair

D	$\forall x\ [\text{pair}(x) \rightarrow \text{pair}(\text{succ}(\text{succ}(x)))]$	Le successeur du successeur de tout nombre pair est pair.

Voici les clauses équivalentes à K1, K2 et K3 et ¬D (les quantificateurs universels sont implicites) :

K1	$\text{pair}(\text{zero})$
K2	$\neg\ \text{pair}(x) \vee \text{impair}(\text{succ}(x))$
K3	$\neg\ \text{impair}(x) \vee \text{pair}(\text{succ}(x))$
¬D	$\text{pair}(a), \neg\text{pair}(\text{succ}(\text{succ}(a)))$

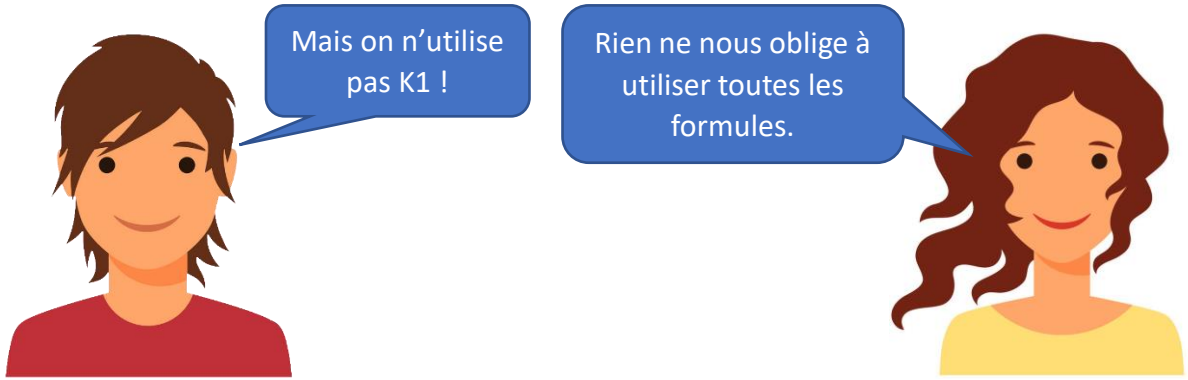
La négation de D est traduite par deux clauses dans lesquelles il n'y a pas de variable, mais un symbole fonctionnel *a* représentant une constante. Cette opération s'appelle la « skolémisation » en référence à son inventeur, le mathématicien Thoralf Skolem.

Qu'est ce qu'un système de preuve ?

Un exemple de preuve par Résolution

On va prouver que l'ensemble de clauses ci dessous est incohérent en utilisant l'unique règle d'inférence du système de preuve par Résolution.

K1	$\text{pair}(\text{zero})$
K2	$\neg\ \text{pair}(x) \vee \text{impair}(\text{succ}(x))$
K3	$\neg\ \text{impair}(x) \vee \text{pair}(\text{succ}(x))$
D1	$\text{pair}(\boldsymbol{a})$,
D2	$\neg\text{pair}(\text{succ}(\text{succ}(\boldsymbol{a})))$



Etape 1 : de K2 et D1 je déduis H1 : $\text{impair}(\text{succ}(\text{a}))$ après unification $x = a$.

$$\neg\ \text{pair}(x) \vee \text{impair}(\text{succ}(x)) \qquad \text{pair}(\boldsymbol{a})$$
$$\text{impair}(\text{succ}(\boldsymbol{a}))$$

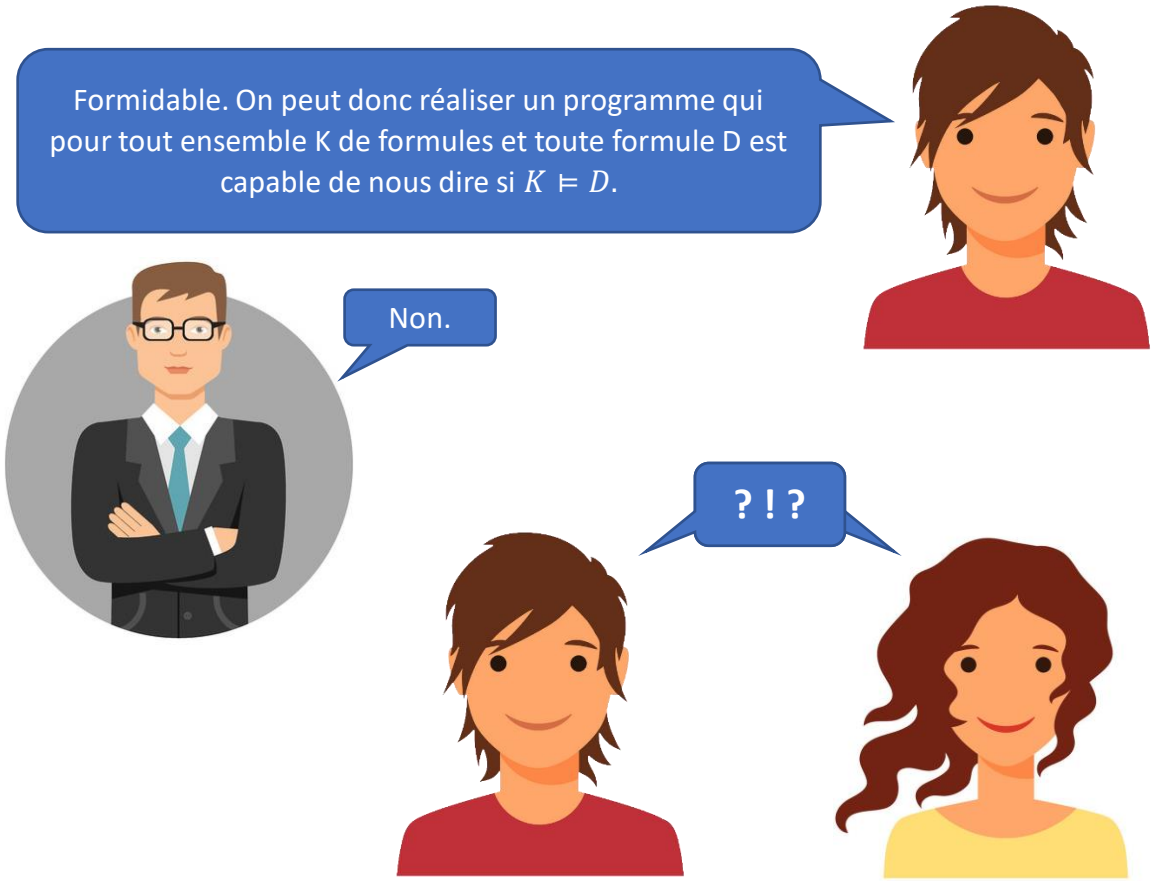
Etape 2 : de H1 et K3 je déduis H2 : $\text{pair}(\text{succ}(\text{succ}(\text{a})))$ après unification $x = \text{succ}(a)$.

$$\text{impair}(\text{succ}(\boldsymbol{a})) \qquad \neg\ \text{impair}(x) \vee \text{pair}(\text{succ}(x))$$
$$\text{pair}(\text{succ}(\text{succ}(\text{a})))$$

Etape 3 : de H2 et D2 je déduis la clause vide $\perp$.

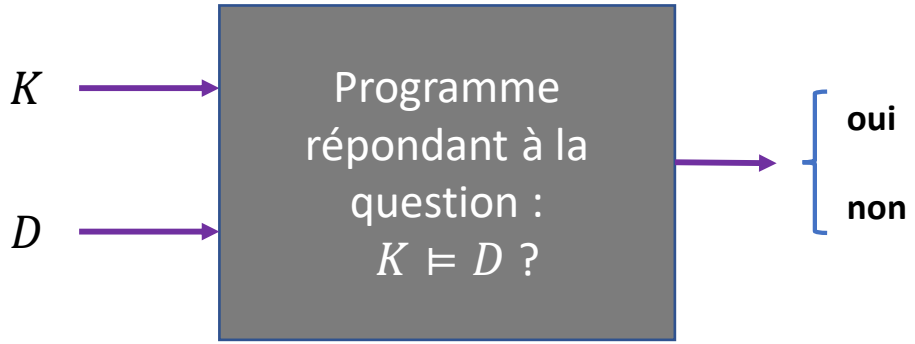
$$\text{pair}(\text{succ}(\text{succ}(\text{a}))) \qquad \neg\text{pair}(\text{succ}(\text{succ}(\text{a})))$$
$$\perp$$

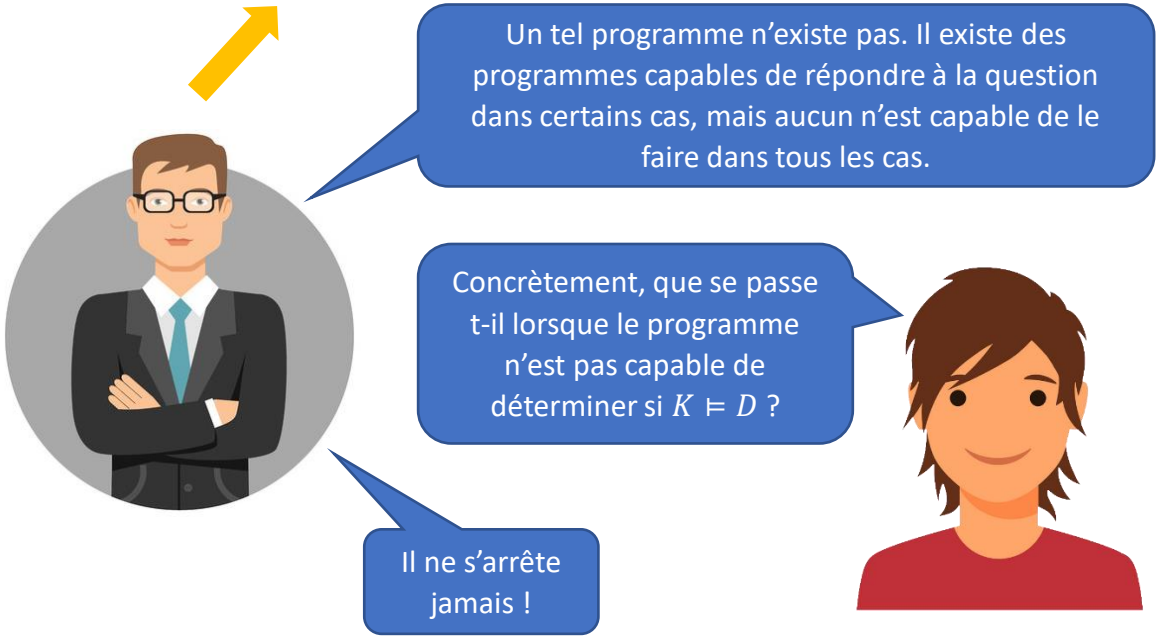
La logique du premier ordre est indécidable



On peut réaliser un programme qui, si on lui donne en entrée un ensemble K de formules et une formule D telle que $K \vDash D$, produit une preuve $K \vdash D$. L'exécution sera peut-être très longue, dans certains cas des milliards de milliards... d'années, mais si l'ordinateur tiens le coup et dispose de suffisamment de ressources, le programme s'arrêtera et produira une preuve formelle $K \vdash D$.

Par contre, il n'existe aucun programme qui, quel que soit l'ensemble K de formules et la formule D placés en entrée, s'arrête à coup sûr en donnant la réponse « oui » si $K \vDash D$ et « non » dans le cas contraire.





La logique du premier ordre est semi décidable

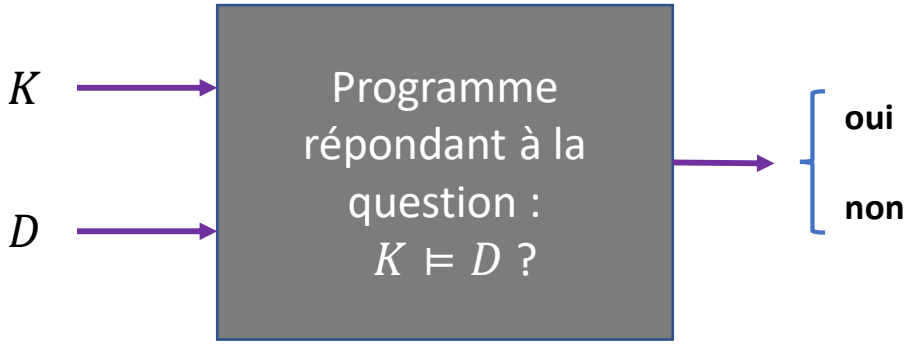
STOP. Ne nous emballons pas. Que signifie exactement « il n'existe pas de programme répondant à tous les coups à la question $K \vDash D$? Personne n'a encore réussi à réaliser un tel programme ?

C'est bien plus grave que ça : il a été prouvé mathématiquement qu'un tel programme n'existe pas.

Plus précisément, quand on parle d'indécidabilité, on parle de l'existence d'un programme sur machine de Turing qui répond à la question posée. Or toutes les machines actuellement connues et capables d'exécuter des programmes peuvent au mieux calculer ce qui est calculable par une machine de Turing.

On dit précisément que le problème de déterminer si une formule est conséquence logique d'une autre (ou d'un ensemble de formule, ce qui revient au même) est *semi-décidable*, pour exprimer la nuance qu'il existe des programmes capables de déterminer si la réponse est « oui », mais peuvent ne jamais s'arrêter si la réponse est « non ».

On sait réaliser des programmes tel que celui illustré ci-dessous qui ne donnent jamais de réponse fausse.

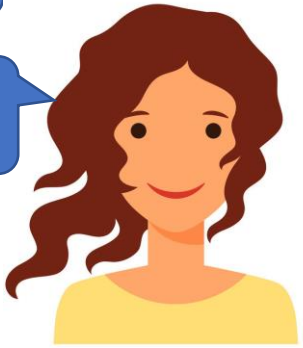


Lors de l'exécution d'un tel programme, trois cas sont possibles :

1. D est conséquence logique de K. Dans ce cas le programme s'arrête et répond « oui ».
2. D n'est pas conséquence logique de K, le programme s'arrête et répond « non ».
3. D n'est pas conséquence logique de K, le programme ne s'arrête jamais.

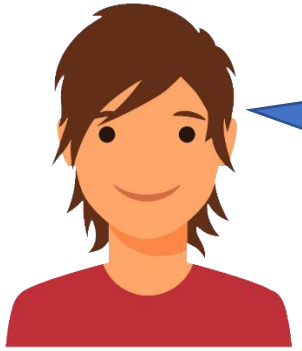
Mais tant que le programme ne s'est pas arrêté, on ne peut pas savoir dans quel cas on est.

Et on ne peut pas non plus savoir si le programme va s'arrêter ou pas.



En résumé

Voici un résumé des « bonnes » et « mauvaises » nouvelles à propos de l'automatisation du raisonnement en logique du premier ordre.



Bonnes nouvelles

Mauvaises nouvelles

Toute conséquence logique $K \vDash D$ peut être prouvée par une dérivation $K \vdash D$ dont chaque étape est réalisée à l'aide d'une règle d'inférence.

La recherche d'une telle dérivation, appelée *preuve formelle*, peut être automatisée, donc réalisée par un programme.

Complétude

Aucun programme ne peut déterminer dans tous les cas sur si une formule D n'est *pas* conséquence logique d'une formule K.

Certains programmes peuvent le faire dans certains cas, mais avec certaines formules en entrée, ils ne s'arrêtent jamais et il n'existe *aucun moyen* de détecter ce genre de situation (sinon il suffirait d'incorporer ce système de détection dans le programme pour l'arrêter).

Indécidabilité