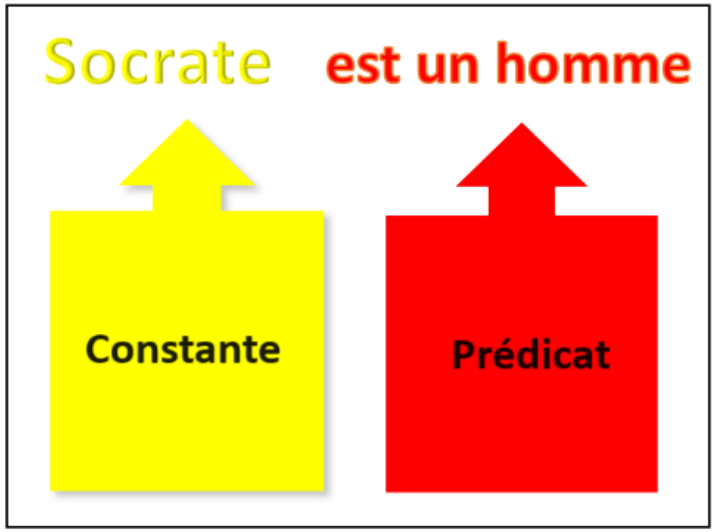
Logique du premier ordre : Raisonnement

Dr. NSENGE MPIA HERITIER, Ph.D

Précédemment

- Qu'est-ce que la logique des prédicats ?
- Syntaxe de la logique des prédicats
  - Constantes
  - Variables
  - Prédicats
  - Fonctions
  - Égalité
  - Quantificateurs
  - Variables liées et variables libres
  - Traduire un texte naturel en logique de premier ordre
- Équivalence de la logique des prédicats
  - En quoi l'équivalence diffère-t-elle entre la logique propositionnelle et la logique du premier ordre?
  - Focus sur les quantificateurs universels et existentiels
- Sémantique de la logique des prédicats
  - Interprétation
  - Domaine
  - Substitution
  - Satisfaisant/valide
  - Détermination de la vérité



$\forall$ $\exists$

Enoncé	Quand c'est Vrai	Quand c'est faux
$\forall x \in D, P(x)$	$P(x)$ est vrai pour tout x	Il y a un x pour lequel $P(x)$ est faux
$\exists x \in D, P(x)$	Il y a un x pour lequel $P(x)$ est vrai	$P(x)$ est faux pour tout x

Plan de la leçon

- Logique du premier ordre: Règles d'inférence
- Preuves de théorèmes en logique du premier
- Inférence par propositionnalisation
- Inférence sans propositionnalisation
- Logique du premier ordre: Inférence avec résolution
- Logique propositionnelle : Clauses de Horn et chaînage
- Inférence de la logique du premier ordre avec chaînage

Le défi des quantificateurs dans l'inférence de la logique du premier ordre

- Les quantificateurs sont une composante importante du pouvoir d'expression de la logique du premier ordre, mais ils perturbent l'inférence automatique.
  - rendent la structure des formules plus complexe
  - augmentent le nombre de règles d'inférence applicables à chaque étape d'une preuve (c'est-à-dire plus de possibilités dans une recherche de preuve).

Passage à l'inférence dans la logique du premier ordre

- Construit sur l'inférence en logique propositionnelle
  - Cependant, les choses deviennent un peu plus compliquées dans la logique du premier ordre
- Il est possible de réduire la base de connaissances de la logique du premier ordre à la logique propositionnelle, puis d'utiliser l'inférence propositionnelle (déjà abordée).
  - Mais selon le problème/domaine, cela peut s'avérer coûteux
- Il peut être plus efficace de construire des règles d'inférence qui fonctionnent directement avec les phrases de la logique du premier ordre
- Nous examinerons ici ces deux approches

Règles d'inférence de la logique propositionnelle

Modus Ponens

$$p \rightarrow q$$
$$p$$
$$\therefore q$$

Modus Tollens

$$p \rightarrow q$$
$$\neg q$$
$$\therefore \neg p$$

Résolution

$$p \vee q$$
$$\neg p \vee r$$
$$\therefore q \vee r$$

Simplification

$$p \wedge q$$
$$\therefore p$$

Addition

$$p$$
$$\therefore p \vee q$$

Syllogisme disjonctif

$$p \vee q$$
$$\neg p$$
$$\therefore q$$

Syllogisme hypothétique

$$p \rightarrow q$$
$$q \rightarrow r$$
$$\therefore p \rightarrow r$$

Conjonction

$$p$$
$$q$$
$$\therefore p \wedge q$$

Contradiction

$$\neg p \rightarrow F$$
$$\therefore p$$

Règles d'inférence de la logique du premier ordre

Règles d'inférence quantifiées

- Tout d'abord, nous apprenons les règles d'inférence pour les quantificateurs dans la logique du premier ordre.
  - Instanciation: convertir les phrases avec quantificateurs en règles sans quantificateurs
    - Permet de "propositionnaliser" n'importe quelle phrase de la logique du premier ordre ou de la base de connaissances.
- Généralisation - règles supplémentaires pour la logique de premier ordre

Règle d'inférence	Nom
$\dfrac{\forall x P(x)}{\therefore P(c)}$	Instanciation universelle
$\dfrac{\exists x P(x)}{\therefore P(c)}$ Pour certain c	Instanciation existentielle
$\dfrac{P(c)}{\therefore \forall x P(x)}$ Pour un c arbitraire	Généralisation universelle
$\dfrac{P(c)}{\therefore \exists x P(x)}$ Pour certain c	Généralisation existentielle

Règle d'inférence : Instanciation universelle (IU)

- **Aussi connue comme**: *Elimination universelle*
  - Toute variable universellement quantifiée, dans une phrase vraie, est remplacée (i.e. substitution) par tout terme approprié du domaine
    - Le résultat est une phrase vraie

- **Hypothèses :**
  - *c* est une constante dans le domaine de x
  - x est une variable qui n'est pas dans le champ d'application d'un quantificateur pour c
  - c'est-à-dire que x peut être remplacé par n'importe quel terme de base

$$\frac{\forall x P(x)}{\therefore P(c)}$$

- **Exemple :**

H1 • "Tous les chats n'attrapent pas les souris"
  - ∀x (Chat(x) → ¬Attrape(x))

H2 • "Tom est un chat"
  - Chat(Tom)

IU • "Si Tom est un chat, alors Tom ne peut pas attraper de souris"
  - Chat(Tom) → ¬Attrape(Tom)

Modus Ponens • "Tom ne peut pas attraper de souris"
  - ¬Attrape(Tom)

Règle d'inférence : Instanciation existentielle

Aussi connue comme: l'élimination existentielle, ou la skolemisation (c'est-à-dire la substitution par une constante skolem)

- Toute variable quantifiée existentiellement, dans une phrase vraie, est remplacée (i.e. substitution) par une nouvelle constante
  - Le résultat est une phrase vraie

- En logique mathématique, la skolémisation d'une formule du calcul des prédicats est une transformation de cette formule, qui, dans le cas d'une forme prénexe, consiste à éliminer toutes les occurrences de quantificateur existentiel en utilisant de nouveaux symboles de fonction (un par quantification existentielle), tout en conservant la satisfaisabilité de la formule.

- **Hypothèses :**
  - *C* est une toute nouvelle constante (qui n'apparaît ni dans cette phrase ni dans aucune autre phrase de la base Knowledge)
- Il est plus pratique de raisonner sur l'objet inconnu que de manipuler constamment le quantificateur existentiel.

$$\frac{\exists x P(x)}{\therefore P(c) \text{ Pour certain c}}$$

- **Exemple :**

H1 • "Il existe des chats qui ne peuvent pas attraper de souris"
  - ∃x (Chat(x) → ¬Attrape(x))

H2 • "SomeCat est un nom générique (placeholder) pour quelque chose qui est un chat"
  - Chat(SomeCat)

IU • "Si SomeCat est un chat, alors SomeCat ne peut pas attraper de souris"
  - Chat(SomeCat) → ¬Attrape(SomeCat)

Modus Ponens • "SomeCat ne peut pas attraper les souris"
  - ¬Attrape(SomeCat)

Généralisation universelle

- **Aussi connue comme**: Introduction universelle

- Toute phrase dont il est démontré qu'elle est vraie pour un *c* arbitraire peut être généralisée à l'aide d'un quantificateur universel
  - P(*c*) ne doit pas avoir été déduit de n'importe quel endroit de base de connaissance où *c* est une variable libre.

- Utile pour manipuler les phrases dans les preuves
- Exemple :
  - "x^2 est non négatif"
    - P(c) Ceci est vrai pour tout c du domaine
  - Nous pouvons donc ajouter un quantificateur universel
    - ∀x P(x)

$$\frac{P(c) \text{ Pour un c arbitraire}}{\therefore \forall x P(x)}$$

- Exemple de preuve dans la **Base de Connaissance**
  - **N.B**: On a qu'à démontrer que la première partie vraie

    $(\forall x)[P(x) \rightarrow Q(x)]$
  - Car si c'est vrai, on peut déduire l'autre partie

    $(\forall x)P(x) \rightarrow (\forall x)Q(x)$
  - La formule 6 a été donc ce qu'on devrait démontrer

- **Exemple** : $(\forall x)[P(x) \rightarrow Q(x)] \wedge (\forall x)P(x) \rightarrow (\forall x)Q(x)$
- Séquence de la preuve :

1.	$(\forall x)[P(x) \rightarrow Q(x)]$	Hypothèse
2.	$(\forall x)P(x)$	Hypothèse
3.	$P(x) \rightarrow Q(x)$	1, Instanciation universelle (IU)
4.	$P(x)$	2, IU: Aucune restriction sur IU concernant la réutilisation d'un nom
5.	$Q(x)$	3, 4, Modus Ponens
6.	$(\forall x)Q(x)$	5, Généralisation universelle

Généralisation existentielle

- **Aussi connue comme**: Introduction existentielle

$$\frac{P(c) \text{ Pour certain c}}{\therefore \exists x P(x)}$$

- Toute phrase dont on montre qu'elle est vraie pour un certain c peut également être généralisée à l'aide d'un quantificateur existentiel.
  - x ne peut pas avoir été une variable dans P(c) (c'est-à-dire P(x,c))

- Si tout le monde vend des actions, c'est que quelqu'un vend des actions.

- Exemple :
  - "x^2 est non négatif"
    - P(c) Ceci est vrai pour tout *c* du domaine
  - Nous pouvons donc ajouter un quantificateur existentiel
    - ∃x P(x)

Preuve Exemple 2

- **Prouver l'argument**
  - $(\forall y)[P(x) \rightarrow Q(x,y)] \rightarrow [P(x) \rightarrow (\forall y)Q(x,y)]$
- **En utilisant la méthode de déduction, nous pouvons déduire**
  - $(\forall y)[P(x) \rightarrow Q(x,y)] \wedge P(x) \rightarrow (\forall y)Q(x,y)$
- **Séquence de la preuve ：**

1. $(\forall y)[P(x) \rightarrow Q(x,y)]$	hypothèse
2. $P(x)$	hypothèse
3. $P(x) \rightarrow Q(x,y)$	1, instanciation universelle
4. $Q(x,y)$	2, 3, modus ponens
5. $(\forall y)Q(x,y)$	4, généralisation universelle

Généralisation vide de sens

- Le domaine est vide, ce qui rend l'énoncé "pour tout" vraie étant donné qu'il n'y a pas d'exemples.
  - Vérité dans un sens sans importance ou sans intérêt

- "Chaque collégien de ce cours a obtenu un A".
  - $\forall x\,(S(x) \rightarrow A(x))$, x est un ensemble vide $\therefore$ Vrai
  - Il n'y avait aucun collégien dans la classe, donc techniquement ceci est vrai

- "Certains enfants sont heureux, en fait tous les enfants sont heureux".
  - $\forall x\, H(x) \therefore \exists x\, H(x)$
  - Implication : si tous les enfants sont heureux, cela signifie que certains le sont.

Règles d'inférence de la logique du premier ordre

Modus Ponens Universel

$$(\forall x)P(x) \Rightarrow Q(x)$$

$$P(A) \text{ pour certain } A \in Dom(x)$$

$$\therefore Q(A)$$

Modus Tollens Universel

$$(\forall x)P(x) \Rightarrow Q(x)$$

$$\sim Q(A) \text{ pour certain } A \in Dom(x)$$

$$\therefore \sim P(A)$$

Autres règles d'inférence de la logique du premier ordre

	Syllogisme hypothétique	$\dfrac{\phi_1 \rightarrow \phi_2 \quad \phi_2 \rightarrow \phi_3}{\phi_1 \rightarrow \phi_3}$
Addition	Introduction de OU	$\dfrac{\phi_1}{\phi_1 \vee \phi_2}$
Syllogisme disjonctif	Elimination de OU	$\dfrac{\phi_1 \vee \phi_2 \quad \neg\phi_2}{\phi_1}$
Conjonction	Introduction de ET	$\dfrac{\phi_1 \quad \phi_2}{\phi_1 \wedge \phi_2}$
Simplification	Elimination de ET	$\dfrac{\phi_1 \wedge \phi_2}{\phi_1}$
	Résolution	$\dfrac{\phi_1 \vee \phi_2 \quad \neg\phi_1 \vee \phi_3}{\phi_2 \vee \phi_3}$

Théorèmes de preuve en logique du premier ordre

Conséquence logique (i.e. implication)

Définition (Conséquence logique)

Une formule G est une conséquence logique des formules $\{F_1, \cdots, F_n\}$ **ssi** pour toute interpretation I if $I \models F_1 \wedge \cdots \wedge F_n$ on constate que $I \models G$.

- Preuves de conséquences logiques :
  - Le processus de preuve est une recherche
  - Les règles d'inférence sont des opérateurs

- Le problème de l'inférence logique dans la logique du premier ordre est :
  - Semi décidable
    - Il existe des algorithmes qui disent oui à chaque phrase impliquée
    - Mais il n'existe pas d'algorithme qui dise non à toute phrase non impliquée
      - Lié à l'indécidabilité de la satisfiabilité/validité

Théorèmes de preuve

Les théorèmes suivants s'appliquent également à la logique du premier ordre

Théorème (Théorème de déduction)

Étant donné un ensemble de formules $\{F_1, \cdots, F_n\}$ et une formule G,

$F_1 \wedge \cdots \wedge F_n \models G$ **ssi** $\models F_1 \wedge \cdots \wedge F_n \rightarrow G$

De la logique propositionnelle $P \overset{val}{\models} Q$ si, et seulement si, $P \Rightarrow Q$ est une tautologie

Théorème (preuve par réfutation) Contradiction

Etant donné un ensemble de formules $\{F_1, \cdots, F_n\}$ et une formule G,

$\models F_1 \wedge \cdots \wedge F_n \rightarrow G$ SSI $F_1 \wedge \cdots \wedge F_n \wedge \neg G$ est incohérent

De la logique propositionnelle $KB \models q$ **si et seulement si** $KB \wedge \neg q$ **est insatisfaisant**

Systèmes de raisonnement

- Incluent les prouveurs de théorèmes, la programmation logique et les systèmes experts.
- La démonstration de théorèmes (c'est-à-dire la manipulation de phrases de la logique du premier ordre pour démontrer des preuves d'implication) peut être utile dans certains domaines plus complexes tels que les mathématiques
- Cependant, nous nous concentrerons ici sur des méthodologies de déduction/inférence "plus simples" mais plus restreintes, largement utilisées dans les systèmes experts
  - Propositionnalisation
  - Résolution
  - Chaînage avant
  - Chaînage arrière

Inférence par propositionnalisation

Approche simple : Propositionnalisation

- Définir les variables de la logique propositionnelle pour chaque formule atomique de la logique des prédicats pour toutes les interprétations

- Les quantificateurs devraient être énumérés
- Peut effectuer des déductions à l'aide de stratégies de logique propositionnelle déjà couvertes :

  - Vérification de la table de vérité (satisfiabilité)
  - Preuve par contradiction
    - Vérification de la table de vérité (insatisfiabilité)
    - Preuve de théorèmes : Détermination de l'insatisfaisabilité
  - Résolution (utilise la preuve par contradiction)

Instanciation dans les bases de connaissances

- Instanciation universelle: peut être appliquée plusieurs fois pour ajouter de nouvelles phrases
  - La nouvelle base de connaissances est logiquement équivalent à l'ancienne

- Instanciation existentielle : peut être appliquée une fois pour remplacer la phrase existentielle
  - La nouvelle base de connaissances n'est pas équivalente à l'ancienne
  - Mais est satisfaisable **ssi** l'ancienne base de connaissances était satisfaisable

Réduction à l'inférence propositionnelle

- Supposons que la base de connaissances contienne uniquement les prédicats suivants

$\forall x\ Président(x) \wedge$ Cupide(x) $\rightarrow Mauvais(x)$
$Président(Mobutu)$
$Cupide(Mobutu)$
$Frère(Museveni, Mobutu)$

- Pour conduire la propositionnalisation, il faut instancier la phrase universelle de toutes les manières possibles
  - Supposons que nous domaine ne contient que ces deux éléments $\upsilon = \{Mobutu, Museveni\}$,
    - *Par conséquent il faut remplacer le premier prédicat au tant de fois qu'avec les éléments de* υ

$Président$(Mobutu) $\wedge$ Cupide(Mobutu) $\rightarrow Mauvais(Mobutu)$
$Président$(Museveni) $\wedge$ Cupide(Museveni) $\rightarrow Mauvais(Museveni)$
$Président(Mobutu)$
$Cupide(Mobutu)$
$Frère(Museveni, Mobutu)$

Réduction à l'inférence propositionnelle (Cont.)

- Toute base de connaissances en logique du premier ordre peut être propositionnalisée de manière à préserver l'implication
- L'inférence avec propositionnalisation en logique du premier ordre :
  - Propositionnaliser la base de connaissances et la requête (c'est-à-dire la conclusion)
  - Appliquer l'inférence basée sur la résolution
  - Retourner le résultat
- Problème : avec les fonctions, il existe une infinité de termes de base (ce qui rend la recherche infinie)
  - par exemple, Père(Père(Père(Mobutu))), etc.

Réduction à l'inférence propositionnelle (Cont.)

- Théorème : Herbrand(1930). Si une phrase α est impliquée par une base de connaissances en logique du premier ordre, *elle est impliquée par un sous-ensemble fini* de la base de connaissances propositionnalisée

- Idée: Pour $\mathrm{n} = 0$ à ∞ $faire$
  - créer une base de connaissances propositionnelle en l'instanciant avec des termes de profondeur *n*
  - voir si α est impliqué par cette base de connaissances
- Le but est ici de propositionnaliser une simple version de la base de connaissances en se basant sur les termes profonds imbriqués d'abord. Si en ce cas l'implication est atteinte, alors on vient de créer une nouvelle base de connaissances qui explore la profondeur suivante et ainsi de suite
- Prenons l'exemple où le domaine ne contient que $\upsilon = \{Mobutu, Museveni\}$
- La requête (implication) $Mauvais(x)$?
- Etant donné la formule ci-contre, on aura les résultats du slide qui suit:

$\forall x\ Président(x) \wedge \mathrm{Cupide(x)} \rightarrow Mauvais(x)$
$Père(x)$
$Président(Mobutu)$
$Cupide(Museveni)$
$Frère(Museveni, Mobutu)$

Le théorème d'Herbrand en pratique

$\forall x\ Président(x) \wedge$ Cupide(x) $\rightarrow Mauvais(x)$
$Père(x)$
$Président(Mobutu)$
$Cupide(Museveni)$
$Frère(Museveni, Mobutu)$

Profondeur 0

Père(Mobutu)
Père(Museveni)
Président(Mobutu)
Cupide(Museveni)
Frère(Museveni, Mobutu)
Président(Mobutu)∧ Cupide(Mobutu) → Mauvais(Mobutu)
Président(Museveni)∧ Cupide(Museveni) → Mauvais(Museveni)
Président(Père(Mobutu)) ∧ Cupide(Père(Mobutu)) → Mauvais(Père(Mobutu))
Président(Père(Museveni)) ∧ Cupide(Père(Museveni)) → Mauvais(Père(Museveni))

Profondeur 1

Profondeur 0 +

Père(Père(Mobutu))
Père(Père(Mobutu))
Président(Père(Père(Museveni))) ∧ Cupide(Père(Père(Museveni)) → Mauvais(Père(Père(Museveni)))

Les problèmes liés à la propositionnalisation

- Problème : fonctionne si α est impliqué, boucle infinie si α n'est pas impliqué
  - La propositionnalisation génère de nombreuses phrases non pertinentes
  - L'inférence peut donc être très inefficace

- Par exemple:

$$\forall x\ Président(x) \wedge \text{Cupide(x)} \rightarrow Mauvais(x)$$
$$Président(Mobutu)$$
$$\exists y\ cupide(y)$$
$$Frère(Museveni, Mobutu)$$

- Il semble évident que Mauvais(Mobutu) est *impliqué*, mais la propositionnalisation produit de nombreux faits tels que Cupide(Museveni) qui ne sont pas pertinents
- Avec *p*, *k-aires* prédicats et *n* constantes, il y a $p.n^k$ instanciations

- Nous pouvons faire de l'inférence directement (c'est-à-dire sans propositionnalisation) avec des phrases logiques de premier ordre.

Inférence sans propositionnalisation

Inférence sans propositionnalisation

- Nous avons la règle d'inférence:

$$\forall x\ Président(x) \wedge \text{Cupide(x)} \rightarrow Mauvais(x)$$

- Nous disposons de faits qui correspondent (partiellement) à la condition préalable ci-dessus

$Président(Mobutu)$ correspond à Président(x)

$\forall y\ Cupide(y)$ correspond à Cupide(x)

- Nous devons les faire correspondre avec des substitutions :
  - On peut faire ceci pour réaliser la substitution: $\theta = \{x/Mobutu, y/Mobutu\}$
  - Ceci marche car x est remplacé par Mobutu et y aussi par Mobutu.
  - Après avoir fait ceci, on constate que les deux expressions juste en haut correspondent à la condition préalable.

Substitution et unification

- La substitution consiste à remplacer la ou les variables d'une phrase par des expressions
  - SUBST({x/Museveni, y/Mobutu}, Frère(x,y)) = Frère(Museveni, Mobutu)
    - On substitue x par Museveni et y par Mobutu

- L'unification consiste à trouver des substitutions (un unificateur) qui font que des phrases différentes soient identiques
  - L'algorithme UNIFY prend deux phrases et renvoie un unificateur pour elles s'il en existe un

Unification

- Exemple :
  - Recherche dans la base de connaissances des phrases qui s'unifient avec Connait(Mobutu, x)

$$UNIFY\big(Connait(Mobutu, x), Connait(Mobutu, Bobila)\big) = \{x/Bobila\}$$
$$UNIFY\big(Connait(Mobutu, x), Connait(y/Bemba)\big) = \{x/Bemba, y/Mobutu\}$$
$$UNIFY\big(Connait(Mobutu, x), Connait(y, Mère(y))\big) = \{y/Mobutu, x/Mère(Mobutu)\}$$
$$UNIFY\big(Connait(Mobutu, x), Connait(x, \quad Janette)\big) = échec$$

- La dernière unification échoue: parce que *x* ne peut pas prendre les valeurs *Janette* et *Mobutu* en même temps.
- Problème : Deux phrases utilisent la même variable
  - Solution simple : Normaliser à part, c'est-à-dire renommer les variables pour éviter les conflits de noms.
  - Exemple: $UNIFY\big(Connait(Mobutu, x), Connait(x_{17}, Janette)\big) = \{x/Janette, x_{17}/Mobutu\}$
- Problème : Plusieurs unificateurs possibles
  - $\theta = \{y/Mobutu, x/z\}$ ou $\theta = \{y/Mobutu, \mathrm{x}/Mobutu,\ z/Mobutu\}$
  - Le deuxième résultat pourrait être obtenu à partir du premier avec une substitution supplémentaire
  - Le premier unificateur est plus général car il impose moins de restrictions sur les valeurs des variables.
  - Pour chaque paire d'expressions unifiables, il existe un seul unificateur le plus général - unique jusqu'au renommage et à la substitution des variables.
- Vérification de l'occurrence : la variable de substitution ne peut pas être présente dans le terme à substituer

Vers une logique d'inférence de premier ordre

- Maintenant que nous disposons d'une stratégie pour faire en sorte que des expressions logiques différentes aient l'air identiques, *nous pouvons effectuer une inférence avec la logique des prédicats*.
- Règles d'inférence généralisées :
  - Utiliser des substitutions qui nous permettent de faire des inférences
  - Substitutions par le processus d'unification
- Exemple : Modus Ponens
  - S'il existe une substitution σ telle que :

$$p \rightarrow q$$
$$p$$
$$\therefore q$$

Logique propositionnelle

$$SUBST\ (\sigma, A_i) = SUBST\ (\sigma, A_i')\ \textbf{pour tout i = 1, 2, n}$$

$$\frac{A_1 \wedge A_2 \wedge \ldots A_n \Rightarrow B, \quad A_1', A_2', \ldots A_n'}{SUBST\ (\sigma, B)}$$

Logique des prédicats

- Avantage des règles généralisées : elles se concentrent sur...
  - Substitutions qui permettent aux inférences de se poursuivre
  - Inférences levées : il n'est pas nécessaire de propositionnaliser les phrases pour faire des inférences.

Modus Ponens généralisé (MPG)

- MPG: Règle d'inférence généralisée simple
  - Combine la généralisation universelle et le modus ponens
  - A partir de *P(c)* et *Q(c)* et ∀x(P(x) ∧ Q(x)) → R(x), dérive *R(c)*
- Pour utiliser le MPG, on doit supposer que toutes les variables sont universellement quantifiées
- En plus, la base de connaissances doit être remplie de clauses définies (nous le verrons bientôt)

$$\frac{p_1', p_2', \ldots, p_n', \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q)}{q\theta} \qquad \text{où } p_i'\theta = p_i\theta \text{ pout tout } i$$

p_1' est $Président(Mobutu)$	p_1 est $Président(x)$
p_2' est $Cupide(y)$	p_2 est $Cupide(x)$
θ est $\{x/Mobutu, y/Mobutu\}$	q est $Mauvais(x)$
$q\theta$ est $Mauvais(Mobutu)$	

Complétude et solidité des MPG

- Le MPG est complet : (pour une base de connaissances constituée de clauses définies)
  - La complétude signifie qu'il peut dériver toutes les phrases qui impliquent
- GMP est solide :
  - Signifie qu'il ne dérive que les phrases qui sont logiquement impliquées
  - Démonstration de la solidité :

$$p_1', \ldots, p_n', \quad (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$$

à condition que $p_i'\theta = p_i\theta$ **pour tout** i

Lemme : Pour toute clause définie p, on a $p \models p\theta$ par instanciation universelle

1. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta)$
2. $p_1', \ldots, p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$
3. **De 1 et 2,** $q\theta$ **découle par Modus Ponens ordinaire**

Autres propriétés de MPG

- Pourquoi le MPG est-il une règle d'inférence efficace ?
  - Elle prend des mesures plus importantes, en combinant plusieurs petites inférences en une seule.
  - Elle prend des mesures raisonnables : elle utilise des instanciations qui sont qui sont garanties pour aider (plutôt que des instanciations universelles aléatoires)
  - Elle utilise une étape de précompilation qui convertit la base de connaissances en la forme canonique des phrases définies
    - Une phrase sous forme définie est une conjonction de clauses définies
    - $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$, c'est-à-dire $(B \rightarrow A) \wedge ((C \wedge D) \rightarrow B)$
  - L'utilisation du MPG consiste essentiellement à prouver des théorèmes.

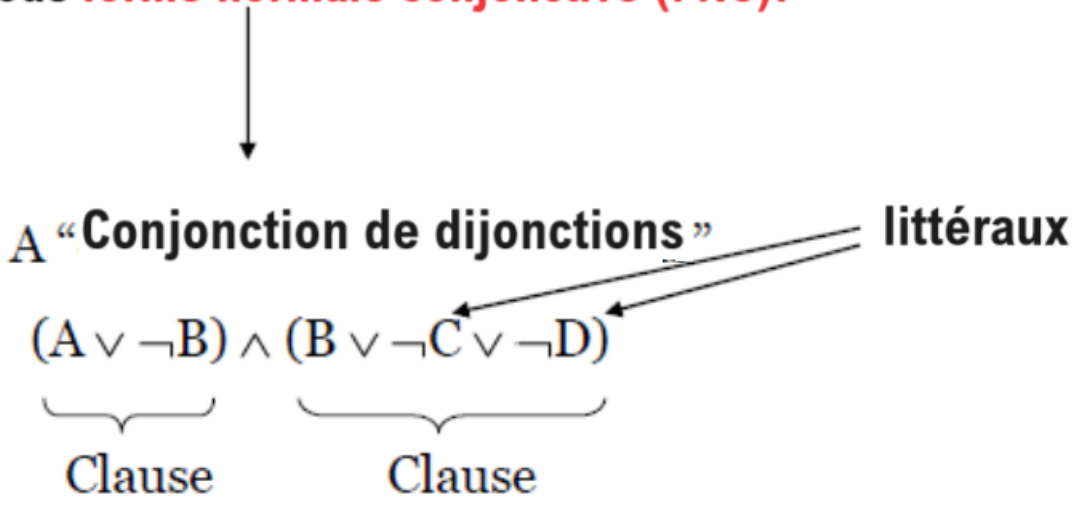
Inférence de la logique de premier ordre avec résolution

Rappel : Inférence basée sur la résolution propositionnelle

On veut prouver : $KB \models \alpha$

équivalent à : $KB \wedge \neg\alpha$ Insatisfaisant

on écrit d'abord $KB \wedge \neg\alpha$ sous forme normale conjonctive (FNC).

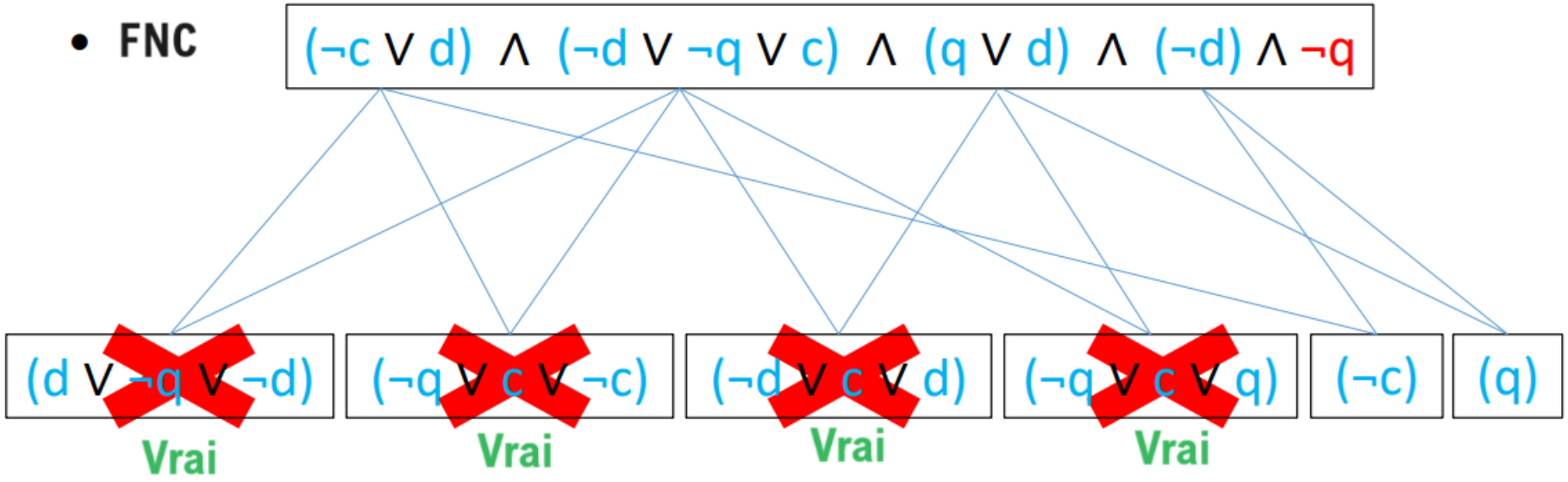


- Toute base de connaissances peut être convertie en FNC
- k-FNC : exactement *k* littéraux par clause

Réexamen de l'algorithme de résolution

- KB: (¬c ∨ d) , (¬d ∨ ¬q ∨ c) , (q ∨ d) , (¬d)

- **FNC**



Dans l'ensemble des clauses résultantes y a-t-il des clauses qui se résolvent en un ensemble vide ?

Règle de résolution

(q) ∧ ¬q → ( )

(KB implique q) ✓

Forme normale conjonctive pour la logique du premier ordre

- La FNC pour la logique du premier ordre demeure une conjonction de clauses où chaque clause est une disjonction de littéraux
  - (a ∨ b ∨ ¬c) ∧ (a ∨ b) ∧ (¬b ∨ ¬c)
- Cependant, les littéraux peuvent désormais contenir des variables
  - Supposé être universellement quantifié
- Chaque phrase en logique du premier ordre peut être convertie en une phrase FNC équivalente par inférence
- La procédure de conversion en FNC est similaire à celle de la logique propositionnelle
  - Cependant, nous devons éliminer le quantificateur existentiel.
  - Quantificateurs universels OK

Conversion en FNC en logique de premier ordre

Toute personne qui aime les animaux est aimée par quelqu'un

$$\forall x[\forall y\ Animal(y) \Rightarrow Aime(x,y)] \Rightarrow [\exists y\ Aime(y,x)]$$

1. Éliminer les biconditionnels et les implications

Implication:

$$P \Rightarrow Q \overset{val}{=\!=} \neg P \vee Q$$

$$\forall x[\neg\forall y\ \neg Animal(y) \vee Aime(x,y)] \vee [\exists y\ Aime(y,x)]$$

2. Déplacer $\neg$ vers l'intérieur : $\neg\forall x\ p \equiv \exists x \neg p, \neg\exists x\ p \equiv \forall x\ \neg p$:

$$\forall x[\exists y\ \neg(\neg Animal(y) \vee Aime(x,y))] \vee [\exists y\ Aime(y,x)]$$

$$\forall x[\exists y\ \neg\neg Animal(y) \wedge \neg Aime(x,y)] \vee [\exists y\ Aime(y,x)]$$

$$\forall x[\exists y\ Animal(y) \wedge \neg Aime(x,y)] \vee [\exists y\ Aime(y,x)]$$

Équivalence de la quantification par négation

De Morgan:

$$\neg(P \wedge Q) \overset{val}{=\!=} \neg P \vee \neg Q$$

$$\neg(P \vee Q) \overset{val}{=\!=} \neg P \wedge \neg Q$$

Conversion en FNC en logique de premier ordre (Cont.)

$$\forall x[\exists y\ Animal(y) \wedge \neg Aime(x,y)] \vee [\exists y\ Aime(y,x)]$$

3. Normaliser les variables : chaque quantificateur devrait utiliser une variable différente.

$$\forall x[\exists y\ Animal(y) \wedge \neg Aime(x,y)] \vee [\exists z\ Aime(z,x)]$$

4. Skolemiser : une forme plus générale d'instanciation existentielle.

- Chaque variable existentielle est remplacée par une fonction Skolem des variables universellement quantifiées qui l'entourent:

$$\forall x\left[Animal\left(F(x)\right) \wedge \neg Aime\left(x, F(x)\right)\right] \vee Aime(G(x), x)$$

Note : On ne peut pas appliquer l'instanciation existentielle ci-dessus parce qu'elle ne correspond pas au modèle.

$$\frac{\exists x P(x)}{\therefore P(c) \text{ Pour certain c}}$$

Conversion en FNC en logique de premier ordre (Cont.)

$$\forall x[Animal(F(x)) \wedge \neg Aime(x, F(x))] \vee Aime(G(x), x)$$

5. Supprimer les quantificateurs universels (la suppression est trivial car on n'a plus besoin de cela):

$$[Animal(F(x)) \wedge \neg Aime(x, F(x))] \vee Aime(G(x), x)$$

5. Distribuer $\wedge$ sur $\vee$:

$$[Animal(F(x)) \vee Aime(G(x), x)] \wedge [\neg Animal(x, F(x)) \vee Aime(G(x), x)]$$

Cette étape peut également nécessiter l'aplatissement des conjonctions et disjonctions imbriquées.

Maintenant en FNC!

Exemple de traduction de la base de connaissances

- Tâche : Traduire le texte naturel suivant en phrases logiques de premier ordre

  - "La loi dit que c'est un crime pour un Congolais de vendre des armes à des nations hostiles. Le Rwanda, pays ennemi de la RDC, possède quelques missiles, et tous ses missiles lui ont été vendus par le colonel Amisi, qui est Congolais".

- Prouver : que le colonel Amisi est un criminel.

Convertir en FNC

- Règles
  - $Congolais(x) \land Arme(y) \land Vend(x, y, z) \land Hostile(z) \rightarrow Criminel(x)$
  - $Missile(M_1)$ et $Possede(Rwanda, M_1)$
  - $\forall x\ Missile(x) \land Possede(Rwanda, x) \rightarrow Vend(Amisi, x, Rwanda)$
  - $Missile(x) \rightarrow Arme(x)$
  - $Ennemi(x, RDC) \rightarrow Hostile(x)$
  - $Congolais(Amisi)$
  - $Ennemi(Rwanda, RDC)$
- Converti en FNC
  - $\neg Congolais(x) \lor \neg Arme(y) \lor \neg Vend(x, y, z) \lor \neg Hostile(z) \lor Criminel(x)$
  - $Missile(M_1)$ et $Possede(Rwanda, M_1)$
  - $\neg Missile(x) \lor \neg Possede(Rwanda, x) \lor Vend(Amisi, x, Rwanda)$
  - $\neg Missile(x) \lor Arme(x)$
  - $\neg Ennemi(x, RDC) \lor Hostile(x)$
  - $Congolais(Amisi)$
  - $Ennemi(Rwanda, RDC)$
- Requête: $\neg Criminnel(Amisi)$
- **N.B**: On a ajouté le requête juste en dessus étant donné que nous voulons faire la preuve par ***résolution***. En effet, la question est de savoir si **Amisi** est criminel ou pas. D'où le rôle de cette requête ajoutée.

Résolution en logique du premier ordre

- Inférence basée sur la résolution (logique du premier ordre) :
  - Réfutation complète pour une base de connaissances générale
    - Peut être utilisée pour confirmer ou réfuter une phrase *p*
    - Ne permet pas de générer toutes les phrases impliquées
  - Nécessite que la base de connaissances de la logique du premier ordre soit réduite à la FNC
  - Utilise une version généralisée de la règle d'inférence propositionnelle

Règle de résolution de la logique propositionnelle

$$(a_1 \vee \dots \vee a_m \vee b)$$
$$(\neg b \vee c_1 \vee \dots \vee c_n)$$
$$\therefore (a_1 \vee \dots \vee a_m \vee c_1 \vee \dots \vee c_n)$$

Version complète du premier ordre :

$$\frac{l_1 \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_n}{(l_1 \vee \dots l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j+1} \vee \dots \vee m_n)\theta}$$

où $UNIFY(l_i, \neg m_j) = \theta$

Par exemple,

$$\frac{\neg Riche(x) \vee Malhereux(x) \quad Riche(Mandala)}{Malheureux(Mandala)}$$

avec $\theta = \{x/Mandala\}$

1. On utilise les deux formules
2. On substitue ceci dans les deux formules
3. On conclut cette formule pour notre KB

N.B:$\neg Riche$ et $Riche$ vont s'annuler. Et, avec la substitution de x par $Mandala$, on reste avec $Malheureux(Mandala)$

Appliquer les étapes de Résolution à FNC $(KB \wedge \neg\alpha)$; complet pour la logique du premier ordre

Logique du premier ordre: Preuve de résolution

- On commence ici la preuve par Résolution pour prouver que Amisi est criminel.
  - On prend la première règle de la KB en FNC avec la requête comme suit:

$$\neg Congolais(x) \vee \neg Arme(y) \vee \neg Vend(x,y,z) \vee \neg Hostile(z) \vee Criminel(x)$$

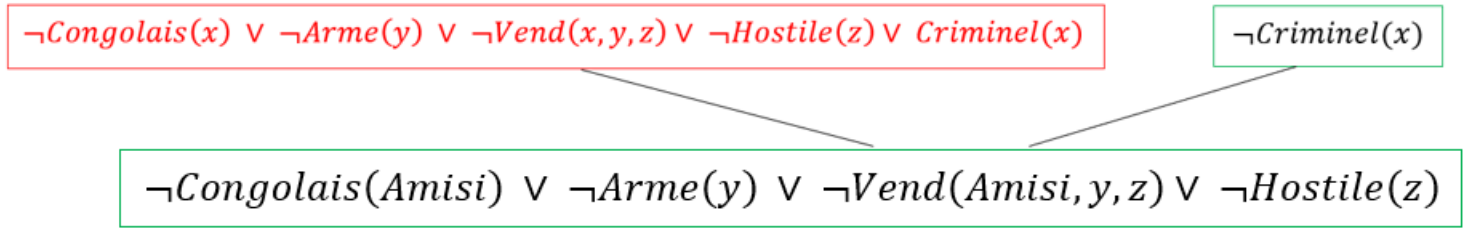
$$\neg Criminel(x)$$

La base de connaissances en FNC

- $\neg Congolais(x) \vee \neg Arme(y) \vee \neg Vend(x,y,z) \vee \neg Hostile(z) \vee Criminel(x)$
- $Missile(M_1)$ et $Possede(Rwanda, M_1)$
- $\neg Missile(x) \vee \neg Possede(Rwanda, x) \vee Vend(Amisi, x, Rwanda)$
- $\neg Missile(x) \vee Arme(x)$
- $\neg Ennemi(x, RDC) \vee Hostile(x)$
- $Congolais(Amisi)$
- $Ennemi(Rwanda, RDC)$

- **Requête :** $\neg Criminnel(Amisi)$

Logique du premier ordre: Preuve de résolution (Cont.)



Où $UNIFY(l_i, \neg m_j) = \theta$

$l_1 = \neg Congolais(x)$

$l_2 = \neg Arme(y)$

$l_3 = \neg Vend(x, y, z)$

$l_4 = \neg Hostile(z)$

$l_5 = Criminel(x)$

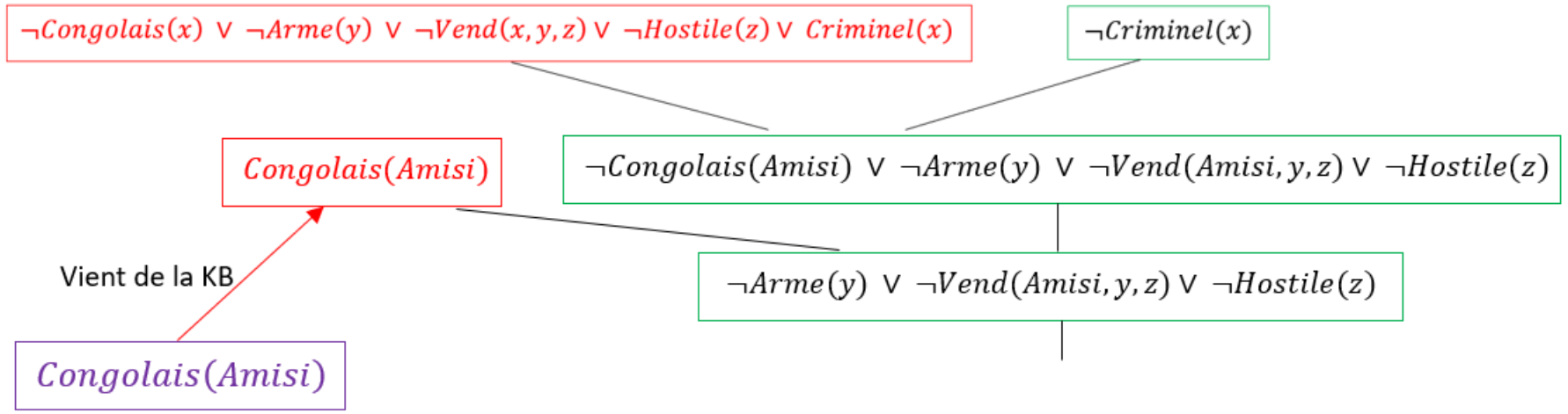
$m_1 = Criminel(Amisi)$

$UNIFY(l_5, \neg m_1) = \theta = \{x/Amisi\}$

$SUBST(\theta, l_1 \vee l_2 \vee l_3 \vee l_4) = \cdots$

- Afin d'utiliser la Résolution, on combine la 1ere règle et la requête ci-dessus en les unifiant.
- Ainsi, on se concentre sur $Criminel(x)$ et $\neg Criminel(Amisi)$. On peut le faire en substituant x par Amisi.
- En faisant ainsi, $Criminel(x)$ et $\neg Criminel(Amisi)$ s'annulent et en même temps Amisi et remplacé partout où il y a x.
- On obtient ainsi la loi juste au-dessus

Logique du premier ordre: Preuve de résolution (Cont.)



Où $UNIFY(l_i, \neg m_j) = \theta$

$l_1 = \neg Congolais(x)$

$l_2 = \neg Arme(y)$

$l_3 = \neg Vend(x, y, z)$

$l_4 = \neg Hostile(z)$

$l_5 = Criminel(x)$

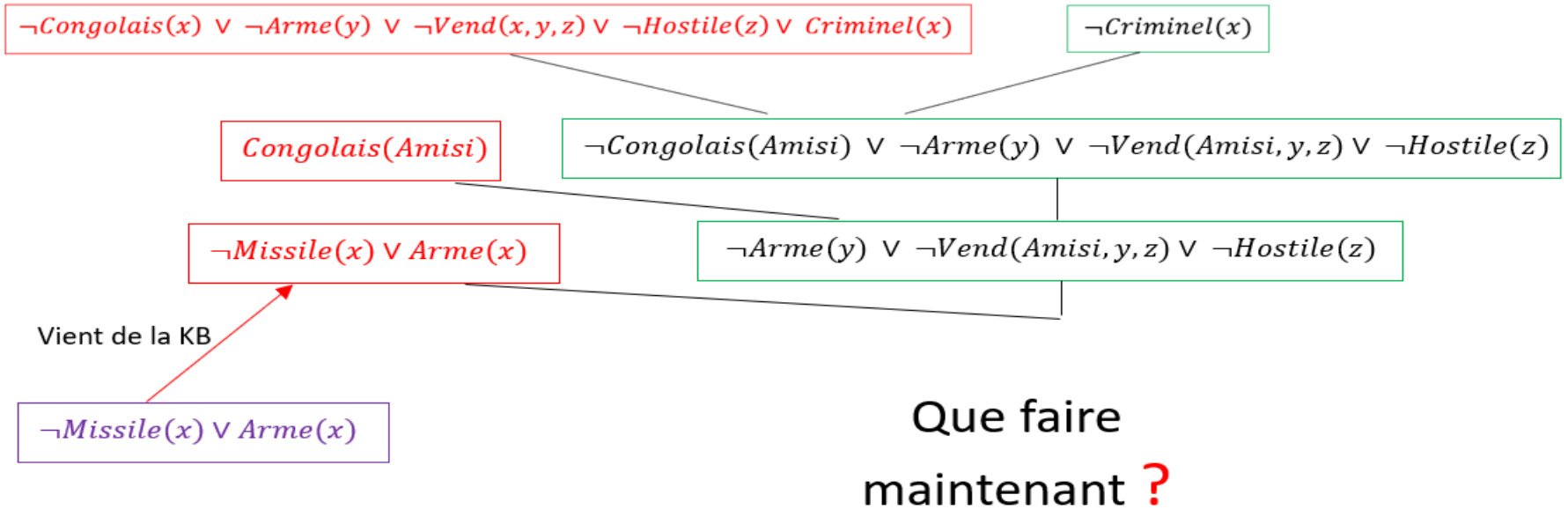
$m_2 = Congolais(Amisi)$

$UNIFY(l_1, \neg m_2) = \theta = \{x/Amisi\}$

$SUBST(\theta,\ l_2 \vee l_3 \vee l_4) = \cdots$

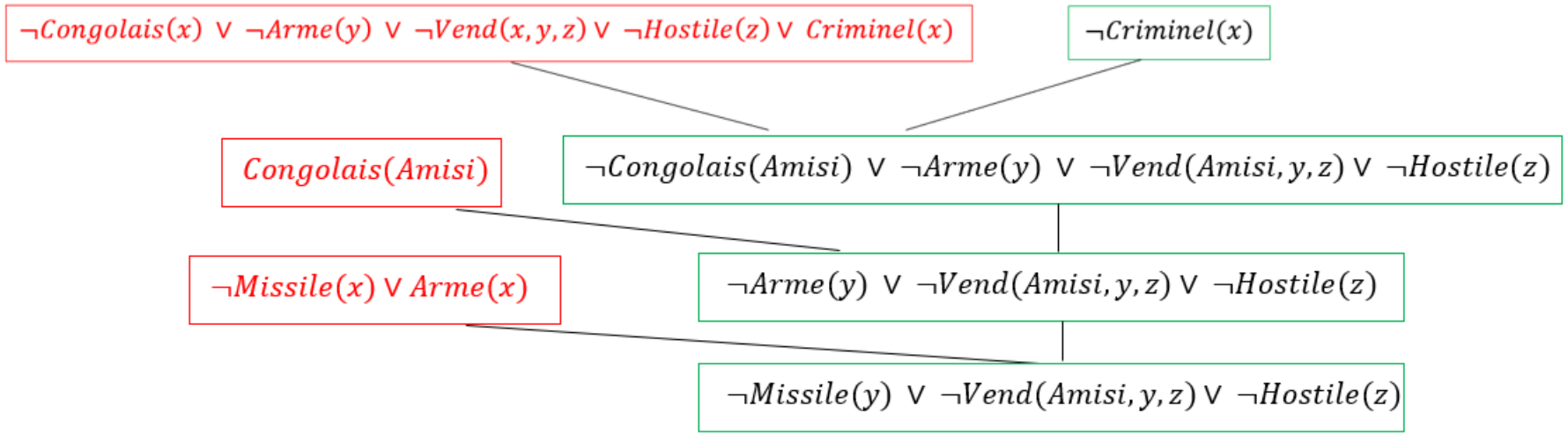
- On récupère cette fois ci $\neg Congolais(Amisi)$ en utilisant la règle $Congolais(Amisi)$ se trouvant dans la KB.
- Pour y arriver, on fait de nouvelle substitution.
- Ainsi, $\neg Congolais(Amisi)$ tombe et on obtient la formule juste en haut:

Logique du premier ordre: Preuve de résolution (Cont.)



- Maintenant on cherche à récupérer la formule $\neg Missile(x) \vee Arme(x)$ de la KB
- On l'unifie avec notre formule retenu dans l'étape précédente et on applique la résolution
- Tout en sachant que *Arme(x) venant de la KB contient x alors que* $\neg Arme(y)$ contient y.
- Du coup, qu'allons nous faire après?
- On va faire une simple substitution dans l'étape du slide qui suit:

Logique du premier ordre: Preuve de résolution (Cont.)



Où $UNIFY(l_i, \neg m_j) = \theta$

$l_2 = \neg Arme(y)$

$l_3 = \neg Vend(x, y, z)$

$l_4 = \neg Hostile(z)$

$m_3 = Arme(x)$

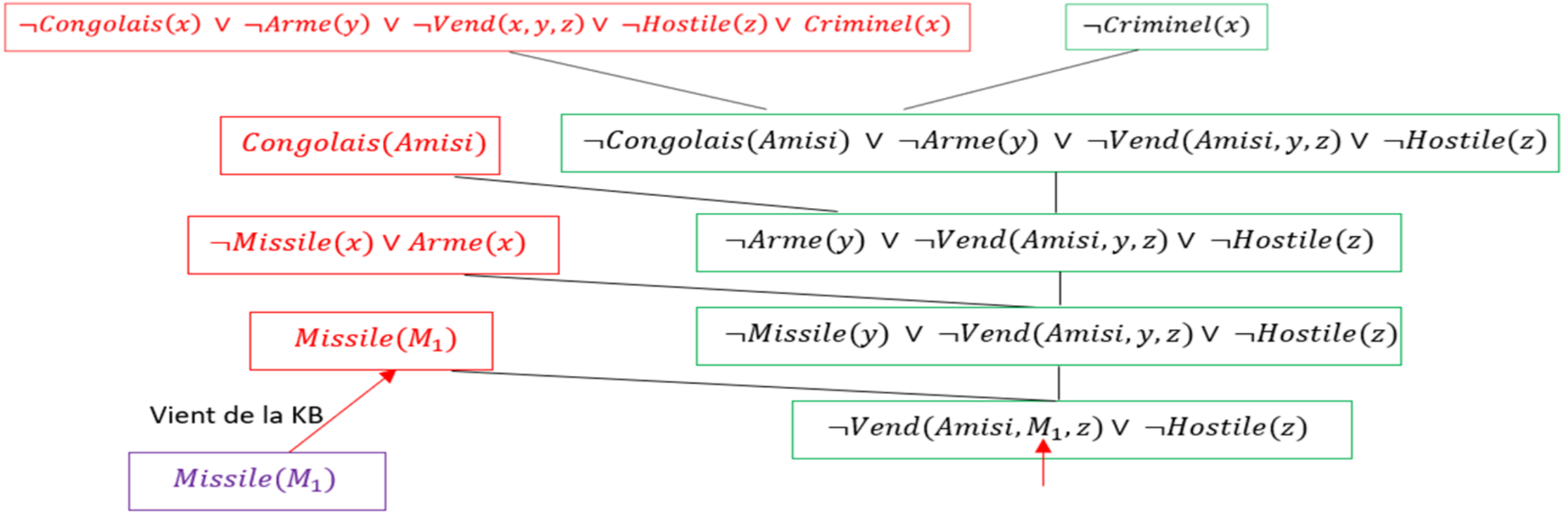
$m_4 = Missile(x)$

$UNIFY(l_2, \neg m_3) = \theta = \{y/x\}$

$SUBST(\theta,\ l_2 \vee l_3 \vee l_4 \vee m_4) = \cdots$

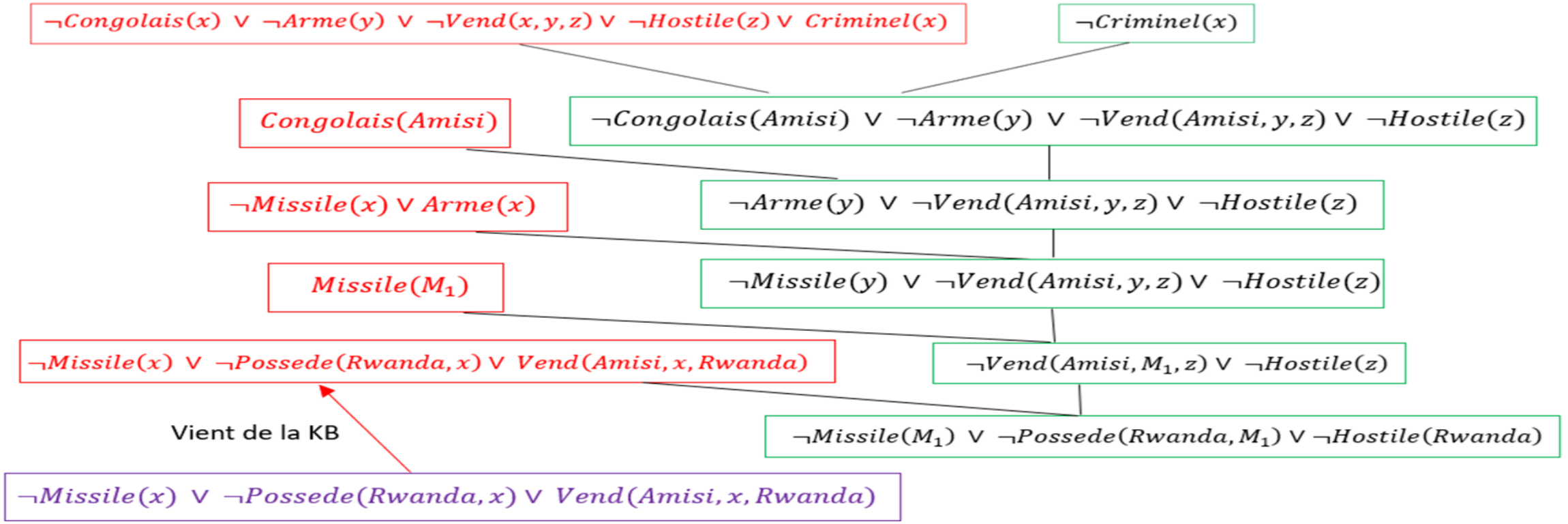
- On utilise une simple substitution en remplaçant y par x dans la formule
- Ainsi, Arme(y) va annuler $\neg Arme(y)$ et $\neg Missile(x)$ va devenir $\neg Missile(y)$

Logique du premier ordre: Preuve de résolution (Cont.)



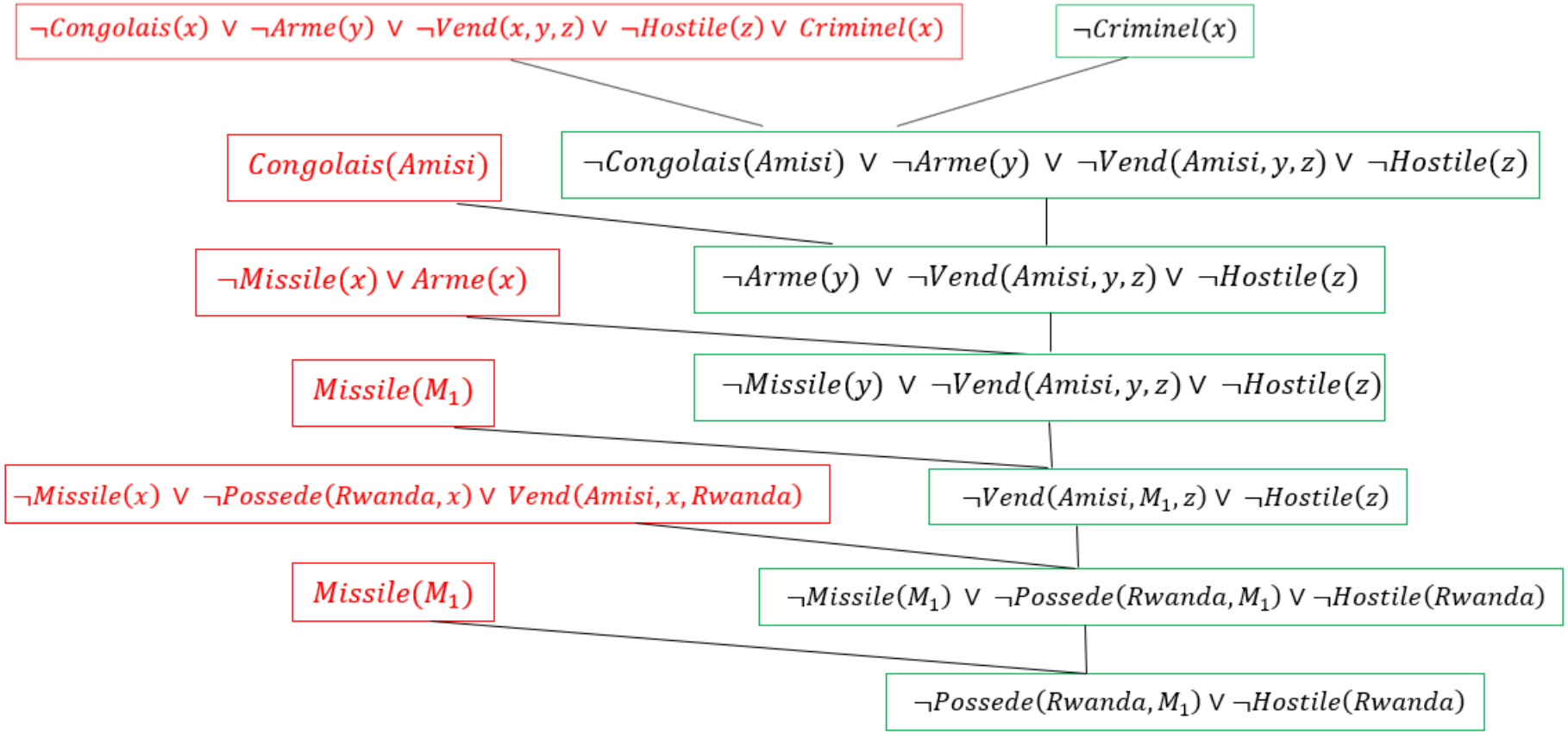
- On récupère $Missile(M_1)$ de la KB
- Pour cela, on doit remplacer y par M_1. Ce qui fait que même le y de Ven(Amisi, y, z) va devenir M_1

Logique du premier ordre: Preuve de résolution (Cont.)



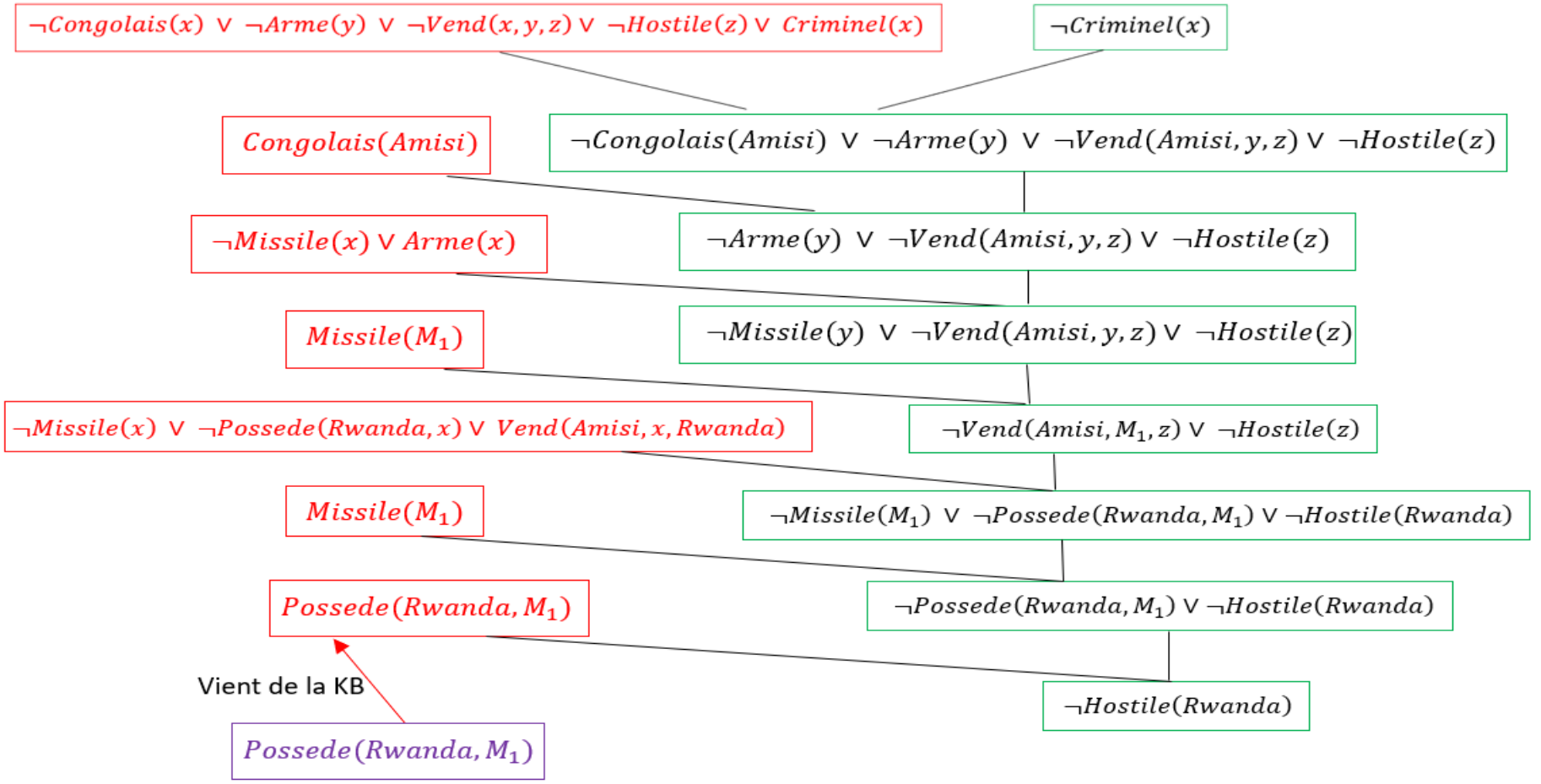
- On récupère maintenant l'expression $\neg Missile(x) \vee \neg Possede(Rwanda, x) \vee Vend(Amisi, x, Rwanda)$ de la KB
- On supprime Vend() et on unifie le reste
- Spécifiquement, x va devenir M_1 et z va devenir Rwanda.

Logique du premier ordre: Preuve de résolution (Cont.)



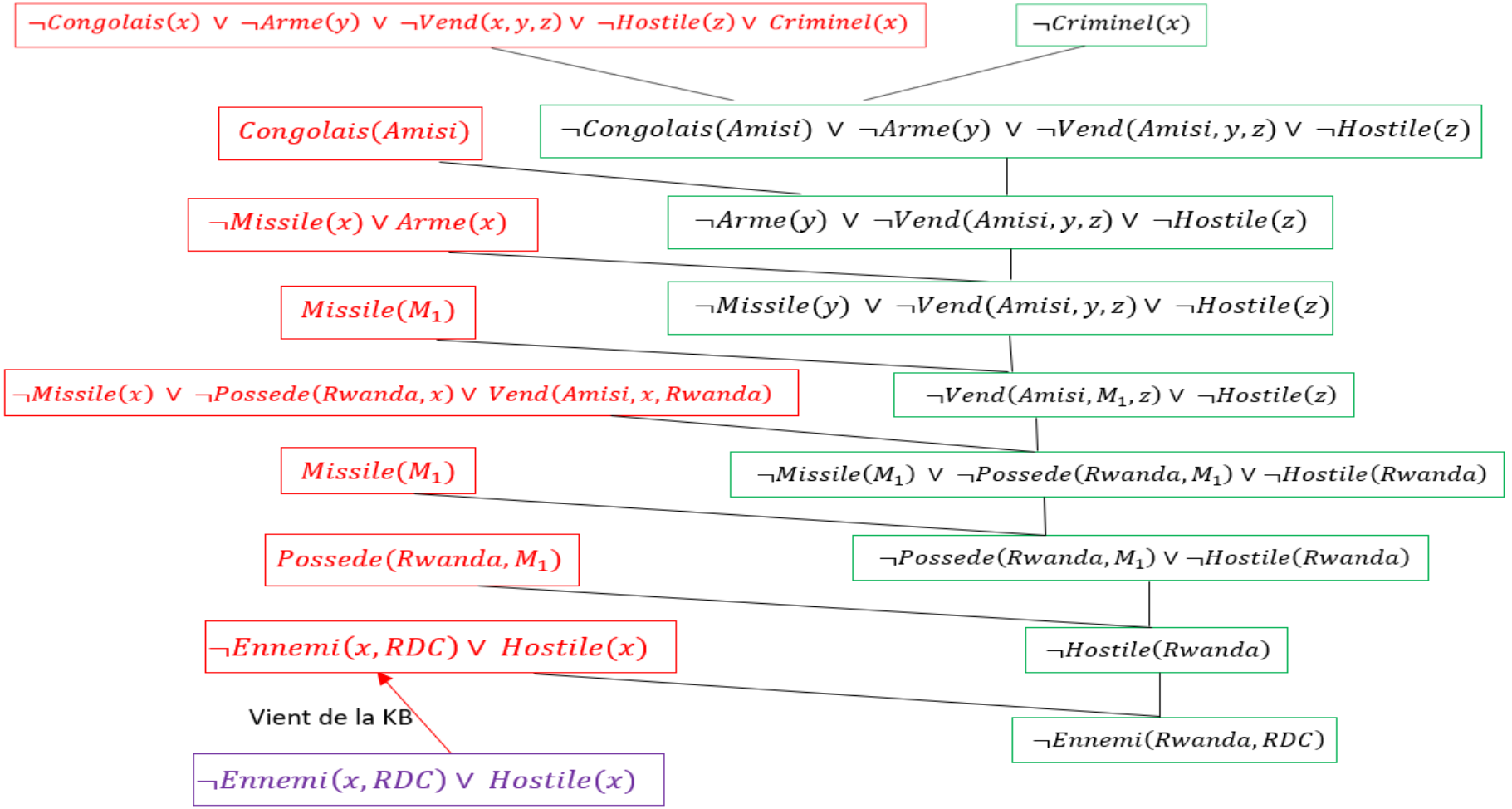
- Au lieu de récupérer une expression dans la KB originale, comme on est en train de le faire, on va récupérer l'expression $Missile(M_1)$ qui a déjà été récupérée ci-haut afin de supprimer $\neg Missile(M_1)$ sans utiliser une quelconque substitution

Logique du premier ordre: Preuve de résolution (Cont.)



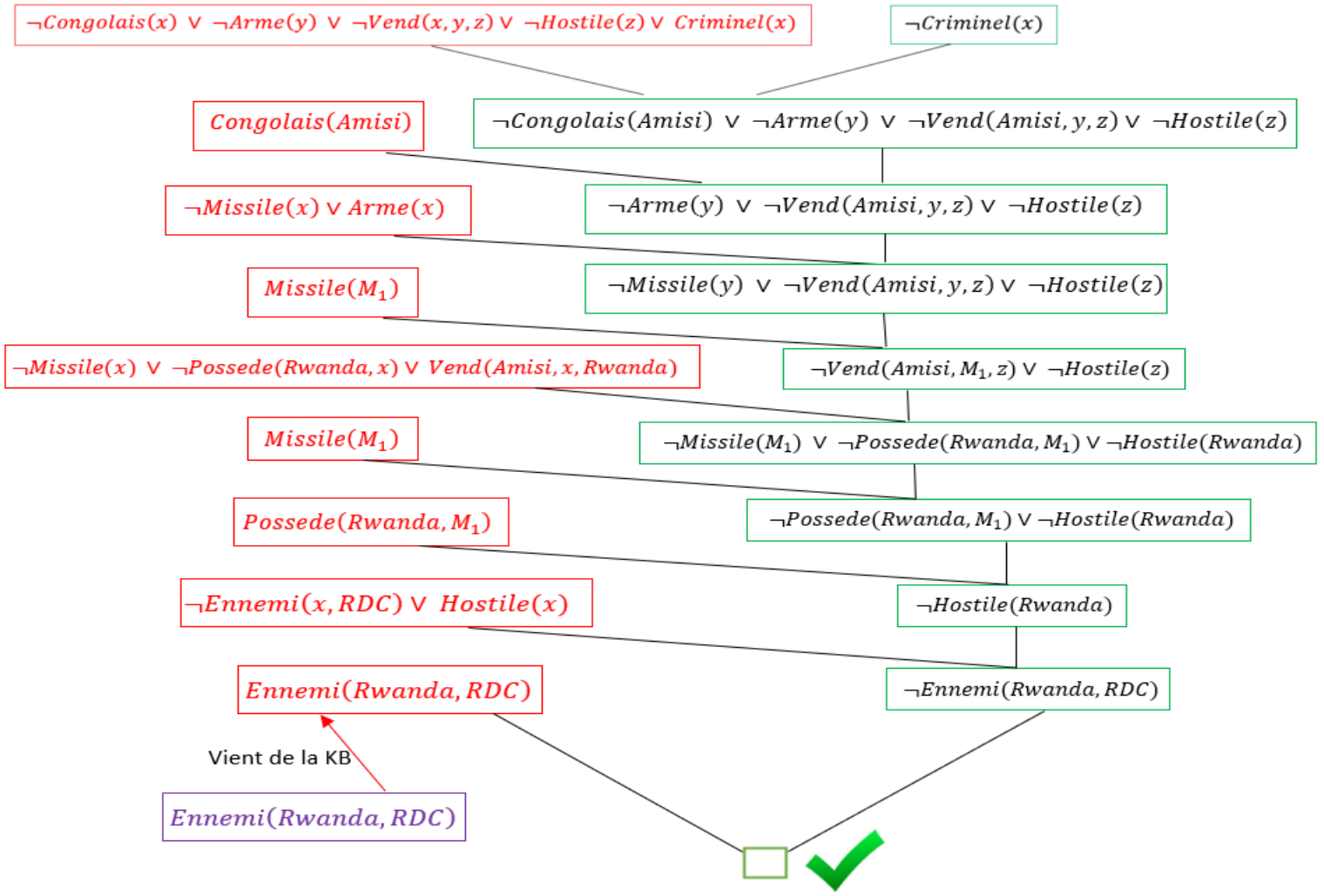
- On récupère $Possede(Rwanda, M_1)$ dans la KB originale afin de supprimer $\neg Possede(Rwanda, M_1)$

Logique du premier ordre: Preuve de résolution (Cont.)



- On va maintenant récupérer $\neg Ennemi(x,RDC) \vee Hostile(x)$ dans la KB originale et on va faire la substitution en remplaçant *x* par Rwanda.

Logique du premier ordre: Preuve de résolution (Cont.)



- Maintenant, en vérifiant notre KB, on se réalise qu'on a l'expression $Ennemi(Rwanda, RDC)$ qui résulte à un ensemble vide une fois combinée avec $\neg Ennemi(Rwanda, RDC)$.
- Parce que nous avons obtenu un ensemble vide, ceci prouve que le Colonel Amisi est un criminel■.

Logique propositionnelle : Clauses de Horn et chaînage

Rappel : Résolution et exhaustivité

- La résolution est complète sur le plan de la réfutation :
  - Si la base de connaissances implique la conclusion, la procédure de réfutation de la résolution, c'est-à-dire l'application de la résolution à la base de connaissances, produira la clause vide.
- La complétude de la résolution en fait une méthode d'inférence importante.
- Dans de nombreuses situations, cependant, la puissance totale de la résolution n'est pas nécessaire
  - La résolution en général peut être exponentielle dans l'espace et le temps
- Certaines bases de connaissances du monde réel satisfont à certaines restrictions sur la forme de la phrase
  - Elles permettent d'utiliser un algorithme d'inférence plus restreint et plus efficace
  - On peut ainsi réduire toutes les clauses à des "clauses de Horn" → la résolution est linéaire en espace et en temps

Clauses définies et clauses de Horn

- Clause définie : Une disjonction de littéraux avec exactement un positif
  - (¬a ∨ b ∨ ¬d ∨ ¬e)
- Clause de Horn : Une disjonction de littéraux avec au plus un élément positif.
  - Légèrement plus générale qu'une clause définie
  - Clauses de but : pas de littéraux positifs
  - (¬a ∨ ¬b ∨ ¬d ∨ ¬e)

- Fermeture dans la résolution : si vous résolvez deux clauses de Horn, vous obtiendrez une clause de Horn.

Bases de connaissances ne comportant que des clauses de Horn

- Toute clause définie peut être écrite comme une implication :
  - La prémisse est une conjonction de littéraux positifs
  - La conclusion est un seul littéral positif
  - (¬a ∨ b ∨ ¬d ∨ ¬e) ≡ a ∧ d ∧ e → b

- Sous forme de Horn :
  - La prémisse est appelée le corps
  - La conclusion est appelée la tête

- Une phrase comportant un littéral positif est appelée un fait.
  - Peut également être écrit sous forme d'implication comme V → **a** (cad si Vrai, alors ***a***; où ***a*** est une proposition qui est le Fait)

Bases de connaissances ne comportant que des clauses de Horn (Cont.)

- Les clauses de Horn peuvent utiliser l'inférence au moyen d'algorithmes de chaînage avant et de chaînage arrière.
  - Les deux algorithmes sont naturels, c'est-à-dire que les étapes de l'inférence sont faciles à suivre.
  - Ce type d'inférence est à la base de la programmation logique.

- Décider de l'implication peut se faire en temps linéaire en fonction de la taille de la base de connaissances !

Chaînage avant

- Comment déterminer qu'une proposition unique (requête) est impliquée par la base de connaissances:
  - Commence par des faits connus (littéraux positifs) dans la base de connaissances
  - Si toutes les prémisses d'une implication sont connues, la tête est ajoutée à la base de connaissances en tant que fait.
    - $a \wedge d \rightarrow b$ (si a et d sont des faits, alors b est ajouté en tant que fait)
  - Le processus se poursuit jusqu'à ce que
    - la requête est ajoutée à la base de connaissances
    - ou qu'aucune autre déduction ne puisse être faite

Chaînage avant : Illustration

- Considérons Q comme la requête
- A et B sont des faits dans la base de connaissances
- Plusieurs liens reliés par un *arc* ( ) indiquent une conjonction (tel que chaque lien doit être prouvé)
- Des liens multiples sans *arc* indiquent une disjonction (tel que tout lien peut être prouvé)

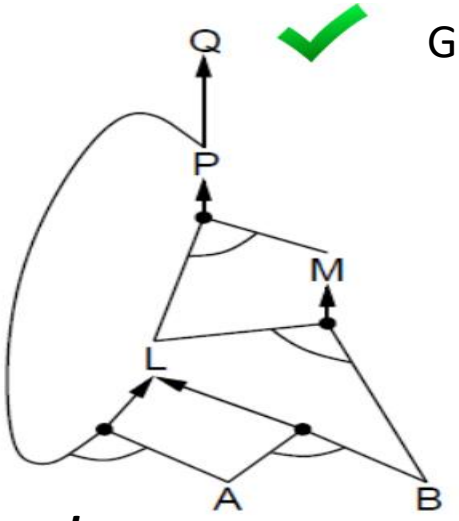
Ensemble de propositions
On commence avec les faits A et B

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Même ensemble de propositions en forme des clauses de Horn

¬P V Q
¬L V ¬M V P
¬B V ¬L V M
¬A V ¬P V L
¬A V ¬B V L

Chaînage avant
Graphe de ET-OU



- Avec le chaînage avant qui a commencé par le bas (A et B), on peut conclure ***L***.
- Mais, on peut aussi conclure L par une autre façon (A ∧ P).
- Avec cela, on a maintenant dans notre schéma B ∧ L; on peut les utiliser pour obtenir ***M***.
- Et, en ayant maintenant L ∧ M, on peut les utiliser comme proposition pour obtenir ***P***.
- Finalement, nous avons ***P***. On peut donc utiliser cette proposition pour obtenir **Q**.

Propriétés de chaînage avant

- Solidité : toute inférence est essentiellement une application de la règle du Modus Ponens

- Complétude : Chaque phrase atomique impliquée sera dérivée

- Exemple du concept de raisonnement basé sur les données
  - Le raisonnement commence par des données connues

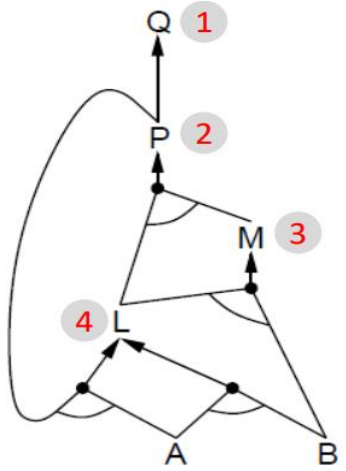
Chaînage arrière

- Travaille à rebours à partir de la requête (Q)
- Si Q est vrai, aucun travail n'est nécessaire (Q est déjà dans la KB)
- Dans le cas contraire, l'algorithme trouve les implications dans la base de connaissances dont la conclusion est Q
- Si toutes les prémisses de l'une de ces implications peuvent être prouvées vraies (par chaînage arrière), alors Q est vraie.
- Une forme de raisonnement orienté vers un but
  - Utile pour répondre à des questions spécifiques telles que :
    - Que dois-je faire maintenant ?
    - Où sont mes clés ?
- Le coût de calcul est souvent beaucoup moins que linéaire dans la base de connaissances, *parce que le processus ne touche que les faits pertinents*.

Illustration du chaînage arrière

- Tester les conclusions en tant qu'hypothèses
- Travailler à partir de *Q* vers l'arrière

1 $P \Rightarrow Q$
2 $L \wedge M \Rightarrow P$
3 $B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
4 $A \wedge B \Rightarrow L$ ✔
A
B



- Premièrement (1), on va chercher si ***Q*** est dans la KB. Ce qui est **Faux** car nous avons seulement A et B comme ***Faits***
- Après, nous cherchons l'implication qui contient ***Q*** comme conclusion. Et pour notre cas, nous avons seulement $P \rightarrow Q$.
- Après, nous prouvons ***P*** car ***P*** impliquant ***Q*** est vrai. Ainsi, on cherche ***P*** dans les Faits, mais il n'est pas dans les Faits.
- Ainsi, après, nous avons à chercher une implication où ***P*** est conclusion. (2) On trouve $L \wedge M \rightarrow P$. Maintenant, pour prouver que ***P*** est vrai, nous devons prouver ***L*** et ***M.***
- Dans la KB, ***L*** et ***M*** aussi n'y sont pas. Ainsi, nous devons aussi prouver ***L*** et ***M***. On commence par prouver ***M***.
- (3) On trouve une implication ($B \wedge L \rightarrow M$) où ***M*** est la conclusion. ***B*** est dans la KB, mais ***L*** non. Par conséquent, nous continuons à prouver ***L***.
- On trouve alors deux implications ($A \wedge P \rightarrow L$ et $A \wedge B \rightarrow L$) où ***L*** est conclusion. $A \wedge P \rightarrow L$ ne peut pas être prouvé d'abord car ***P*** n'est pas encore prouvé. Ainsi, nous prouvons $A \wedge B \rightarrow L$
- (4) Cependant, *$A \wedge B$ prouve **L** car **A** et **B** se trouvent dans la KB. Par conséquent, nous avons prouvé que **Q** est Vrai.*

Inférence logique de premier ordre avec chaînage

Clauses définies de premier ordre

- Ressemblent beaucoup aux clauses définies propositionnelles mais
  - Elles peuvent inclure des variables
    - Elles sont supposées être universellement quantifiées
    - Elles sont omises par défaut, généralement lorsqu'elles sont écrites.
  - Les clauses définies sont soit :
    - Atomiques
    - Une implication dont l'antécédent est une conjonction de littéraux positifs et le conséquent est un seul littéral positif
      - Exemples :
        - $\text{Président}(x) \land \text{Cupide}(x) \rightarrow \text{Mauvais}(x)$
        - $\text{Président}(\text{Mobutu})$
        - $\text{Cupide}(y)$
  - Toutes les bases de connaissances ne peuvent pas être converties en un ensemble de clauses définies.
    - En raison de la restriction d'un seul littéral positif

Exemple de traduction de la base de connaissances

- Tâche : Traduire le texte naturel suivant en phrases logiques de premier ordre

  - "La loi dit que c'est un crime pour un Congolais de vendre des armes à des nations hostiles. Le Rwanda, pays ennemi de la RDC, possède quelques missiles, et tous ses missiles lui ont été vendus par le colonel Amisi, qui est Congolais".

- Prouver : que le colonel Amisi est un criminel.

Exemple de traduction de la base de connaissances (Cont.)

... c'est un crime pour un Congolais de vendre des armes à des nations hostiles :

$$Congolais(x) \land Arme(y) \land Vend(x, y, z) \land Hostile(z) \rightarrow Criminel(x)$$

Rwanda ... possède quelques missiles, cad, $\exists x\ Possede(Rwanda, x) \land Missile(x)$:

$Possede(Rwanda, M_1)$ et $Missile(M_1)$

... tous ses missiles lui ont été vendus par le colonel Amisi

$$\forall x\ Missile(x) \land Possede(Rwanda, x) \rightarrow Vend(Amisi, x, Rwanda)$$

Missiles sont des armes :

$$Missile(x) \rightarrow Arme(x)$$

Un ennemi de la RDC est considéré comme "hostile" :

$$Ennemi(x, RDC) \rightarrow Hostile(x)$$

Amisi, qui est Congolais...

$$Congolais(Amisi)$$

Le pays Rwanda, un ennemi de la RDC...

$$Ennemi(Rwanda, RDC)$$

Prouver : $Criminel(Amisi)$

Un algorithme simple de chaînage avant

- Partir des faits connus, ***déclencher toutes les règles dont les prémisses sont satisfaites en ajoutant leurs conclusions aux faits connus***
  - Le processus se répète jusqu'à ce qu'une réponse soit apportée à la requête
  - Version simple mais inefficace du chaînage avant décrite ici:

fonction $LPO\_CA\_Cherche(KB, \alpha)$ retourne une substitution ou *faux*

répéter jusqu'à ce que $nouvelle$ soit vide

$nouvelle \leftarrow \{\}$

pour chaque phrase r dans KB fait

$(p_1 \wedge \ldots \wedge p_n \rightarrow q) \leftarrow STANDARDISER\_ELEMENT(r)$

pour chaque θ tel que $(p_1 \wedge \ldots \wedge p_n)\theta = (p_1' \wedge \ldots \wedge p_n')\theta$

pour certains $p_1', \ldots, p_n'$ dans KB

$q' \leftarrow SUBST(\theta, q)$

si q' n'est pas un changement de nom d'une phrase déjà présente dans KB ou $nouvelle$ alors faire

ajouter q' dans $nouvelle$

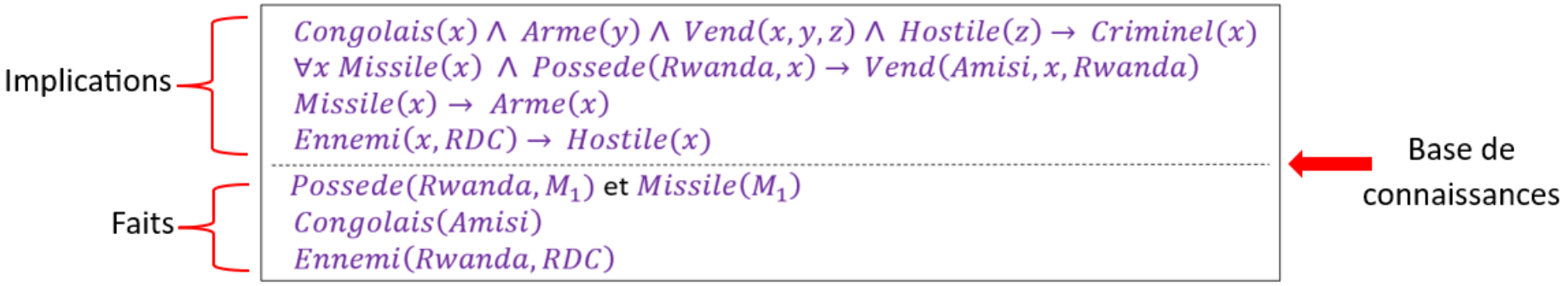
$\emptyset \leftarrow UNIFY(q', \alpha)$

si $\emptyset$ n'est pas un *échec* alors retourner $\emptyset$

ajouter $nouvelle$ dans KB

retourner *faux*

Chaînage avant

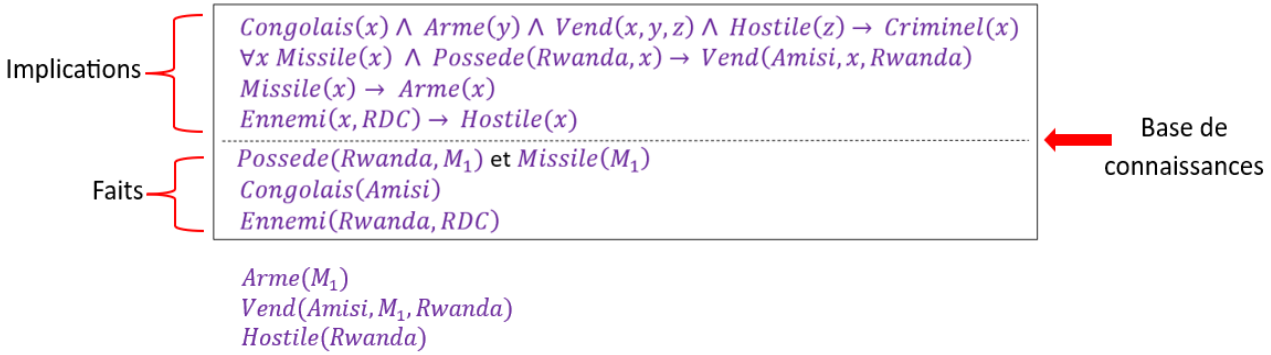


Prouver: $Criminel(Amisi)$

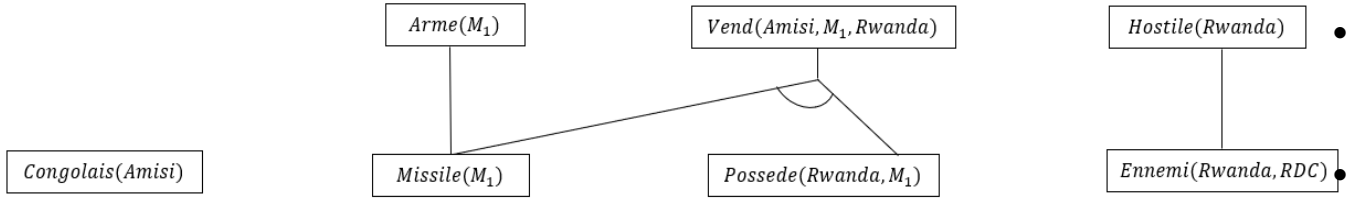
N.B: On met les faits dans le plus bas du graphe

$Congolais(Amisi)$ $Missile(M_1)$ $Possede(Rwanda,M_1)$ $Ennemi(Rwanda,RDC)$

Chaînage avant (Cont.)



Prouver: $Criminel(Amisi)$



- On va d'abord utiliser les faits existants dans la KB pour voir quels nouveaux faits on peut dériver à partir de notre KB
- On peut utiliser la 3e implication dans l'image ci-contre $(Missile(x) \rightarrow Arme(x))$ bien sûr en substituant x par M_1, ce qui nous donne le nouveau fait $Arme(M_1)$.
- En regardant la 2e implication de notre KB, on voit que $Missile(x) \wedge Possede(Rwanda,x)$ implique $Vend(Amisi,x,Rwanda)$, c'est ainsi que dans le graphe ci-contre, on a lié $Missile(M_1)$ et $Possede(Rwanda,M_1)$ avec $Vend(Amisi,M_1,Rwanda)$ qui devient également un nouveau fait dans notre KB.
- Enfin, nous pouvons utiliser le fait $Ennemi(Rwanda,RDC)$ en se basant sur la 4e implication pour générer un nouveau fait ($Hostile(Rwanda)$).
- Du coup, on vient d'ajouter 3 nouveaux faits dans notre KB: $Arme(M_1)$, $Vend(Amisi,M_1,Rwanda)$ et $Hostile(Rwanda)$ comme indiqué dans l'image ci-contre.
- On a maintenant 6 faits dans notre KB.

Chaînage avant (Cont.)

Implications

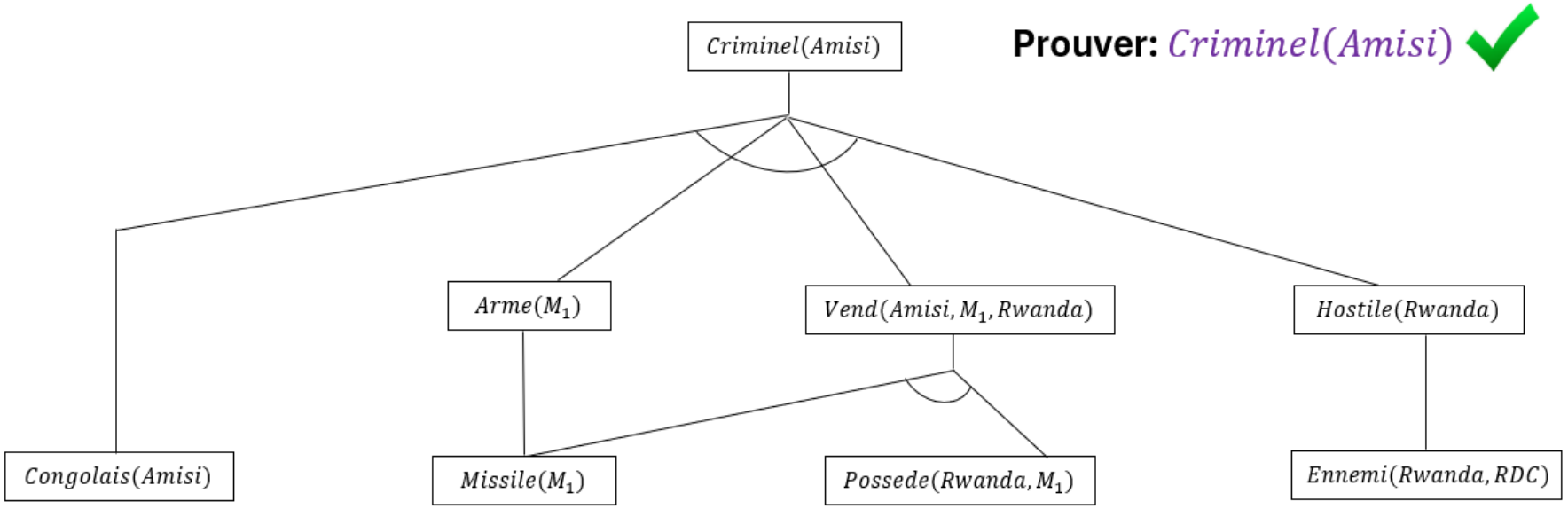
$Congolais(x) \wedge Arme(y) \wedge Vend(x, y, z) \wedge Hostile(z) \rightarrow Criminel(x)$
$\forall x\ Missile(x) \wedge Possede(Rwanda, x) \rightarrow Vend(Amisi, x, Rwanda)$
$Missile(x) \rightarrow Arme(x)$
$Ennemi(x, RDC) \rightarrow Hostile(x)$

Faits

$Possede(Rwanda, M_1)$ et $Missile(M_1)$
$Congolais(Amisi)$
$Ennemi(Rwanda, RDC)$

Base de connaissances

$Arme(M_1)$
$Vend(Amisi, M_1, Rwanda)$
$Hostile(Rwanda)$
$Criminel(Amisi)$



Prouver: $Criminel(Amisi)$ ✓

On peut ainsi utiliser la 1e implication en liant les conjonctions du graphe pour conclure que Amisi est un criminel;

- Ce qui est aussi ajouter dans notre KB.

Ainsi notre preuve est complétée

Propriétés du chaînage avant

- Solide et complète pour les clauses définies de la logique du premier ordre
- Datalog : Clauses définies de la logique du premier ordre + pas de fonction (par exemple, crime KB)
  - Le chaînage avant termine le Datalog en un nombre d'itérations polynomiales :
    - p, prédicats k-aires et n constantes, il y a au plus $p.n^k$ littéraux
      - Similaire à la propositionnalisation
- Peut ne pas se terminer en général si la requête (q) n'est pas impliquée
  - Incontournable car l'implication avec des clauses définies est semi-décidable

Efficacité du chaînage avant

- Observation :
  - Il n'est pas nécessaire de faire correspondre une règle à l'itération *k* si une prémisse n'a pas été ajoutée à l'itération *k - 1.*
  - *Au lieu de cela* : faire correspondre chaque règle dont la prémisse contient un littéral nouvellement ajouté
- La mise en correspondance elle-même peut être coûteuse
- L'indexation de la base de données permet de retrouver des faits connus en O(1)
  - par exemple, la requête *Missile(x)* permet de retrouver *Missile(M1)*
- La mise en correspondance de prémisses conjonctives avec des faits connus est NP-difficile
- Le chaînage avant est largement utilisé dans les bases de données déductives.
- Il existe des stratégies pour accélérer les algorithmes de chaînage avant.
  - Dans une certaine mesure

Chaînage arrière

- Recherche ET/OU : OU (requête prouvée par n'importe quelle règle de la base de connaissances), ET (tous les conjoints des prémisses de l'implication doivent être prouvés
- En travaillant à rebours à partir de la requête (q), l'algorithme trouve des implications dans la KB dont la conclusion est q
- Si toutes les prémisses de l'une de ces implications peuvent être prouvées vraies (par chaînage arrière), alors q est vrai
- Sinon, la recherche se poursuit.

fonction $LPO\_ChArriere\_Cherche(KB, buts, \theta)$ retourne un ensemble de substitutions

inputs : KB, une base de connaissances

$buts$, une liste de conjoints formant une requête (θ déjà appliqué)

θ, la substitution actuelle, initialement la substitution vide { }

variables locales : *réponses*, un ensemble de substitutions, initialement vide

si *buts* est vide alors retourner $\{\theta\}$

$q' \leftarrow SUBST(\theta, PREMIER(buts))$

pour chaque phrase r dans *KB*

où $STANDARDISER\_ELEMENT(r) = (p_1 \wedge \ldots \wedge p_n \rightarrow q)$ et $\theta' \leftarrow UNIFY(q, q')$ est un succès

$nouveau\_buts \leftarrow [p_1, \ldots, p_n | RESTE(buts)]$

$réponses \leftarrow LPO\_ChArriere\_Cherche\big(KB, nouveau\_buts, COMPOSER(\theta', \theta)\big) \cup réponses$

retourner *réponses*

Chaînage arrière (Cont.)

- Première chose, c'est de faire la substitution de ***x*** par ***Amisi***.

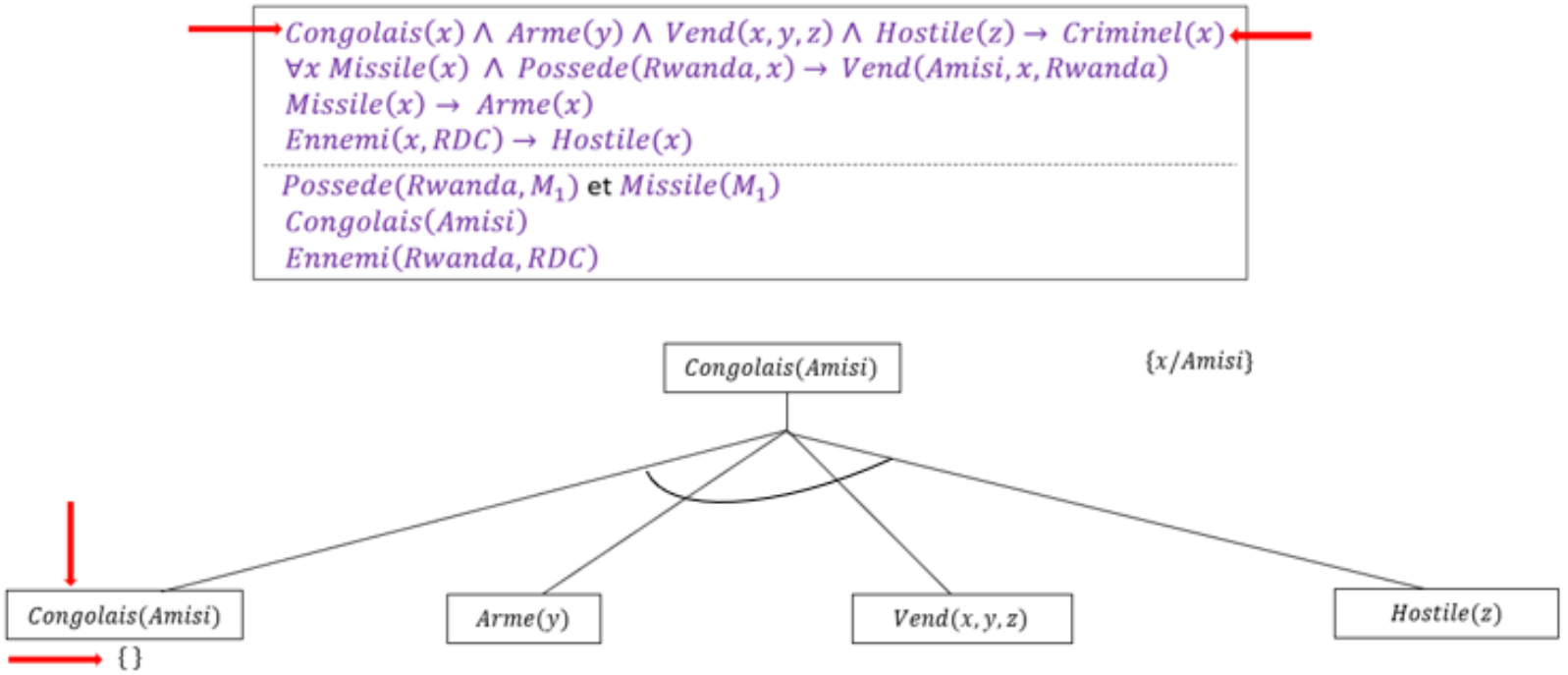
$Congolais(x) \wedge Arme(y) \wedge Vend(x, y, z) \wedge Hostile(z) \rightarrow Criminel(x)$
$\forall x\ Missile(x) \wedge Possede(Rwanda, x) \rightarrow Vend(Amisi, x, Rwanda)$
$Missile(x) \rightarrow Arme(x)$
$Ennemi(x, RDC) \rightarrow Hostile(x)$

$Possede(Rwanda, M_1)$ et $Missile(M_1)$
$Congolais(Amisi)$
$Ennemi(Rwanda, RDC)$

$Congolais(Amisi)$

Chaînage arrière (Cont.)

- Après, nous allons prouver chaque unique prémisse de cette implication
- On commence par Congolais(x)
- On constate qu'on a seulement une substitution de ***x*** pour ***Amisi*** dans Congolais(x)
- Etant donné que Congolais(Amisi) (après substitution) correspond avec ***Congolais(Amisi)*** qui se trouve dans la KB originale, nous concluons que Congolais(Amisi) est vrai.

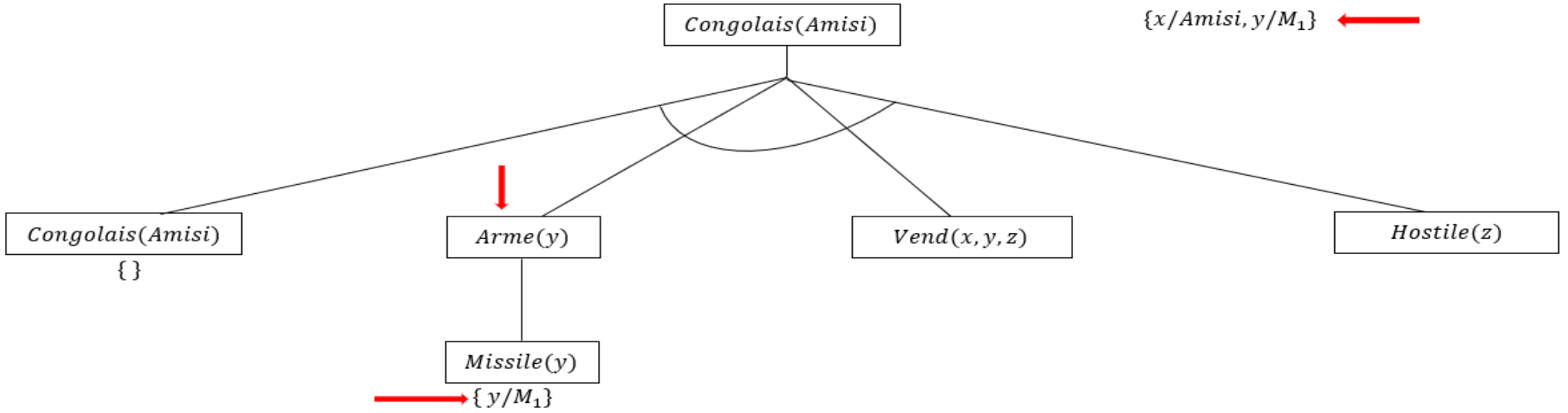


Chaînage arrière (Cont.)

- On trouve, pour la prémisse Arme(y) une implication dans la KB qui lui correspond
- Mais, x ne correspond pas à y. On va alors substituer x par y.
- Ainsi, on peut ajouter Missile(y) dans la KB.
- Malheureusement, Missile(y) qui implique Arme(y) n'est pas prouvé.
- Mais on constate que dans les faits de notre KB, il y a Missile(M_1). On va substituer ainsi y de Missile par M_1
- En substituant ainsi, nous venons maintenant de prouver que Missile(y) qui est ajouté est vrai.

$Congolais(x) \wedge Arme(y) \wedge Vend(x,y,z) \wedge Hostile(z) \rightarrow Criminel(x)$
$\forall x\ Missile(x) \wedge Possede(Rwanda,x) \rightarrow Vend(Amisi,x,Rwanda)$
$Missile(x) \rightarrow Arme(x)$
$Ennemi(x,RDC) \rightarrow Hostile(x)$

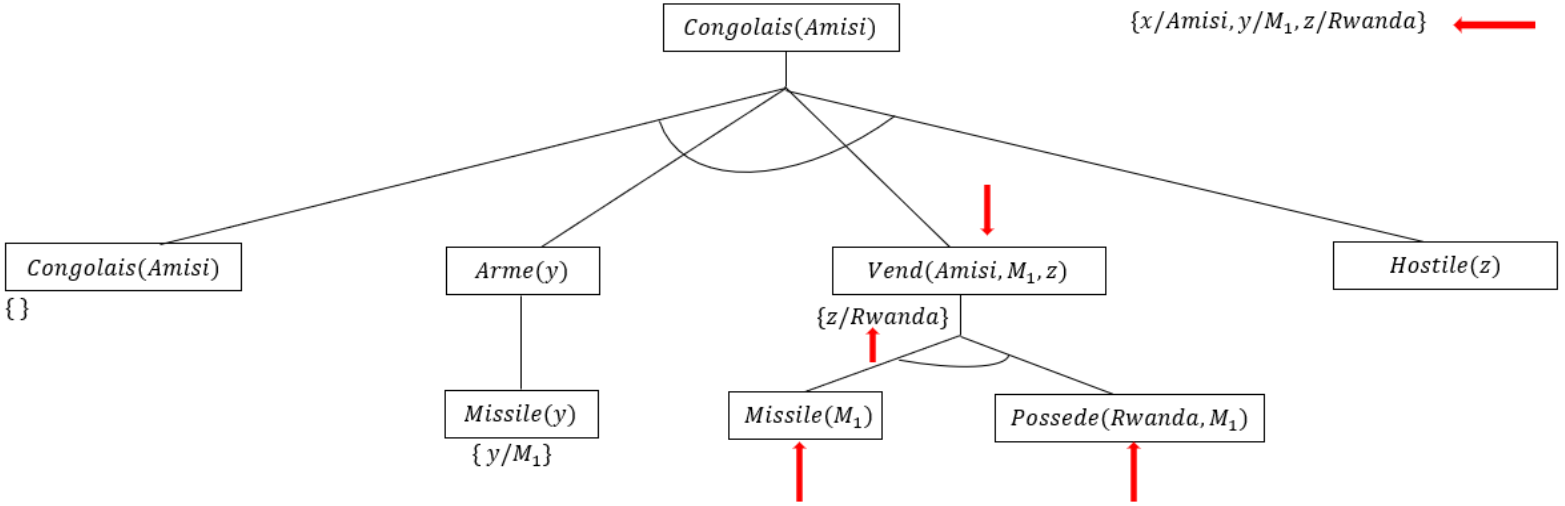
$Possede(Rwanda,M_1)$ et $Missile(M_1)$
$Congolais(Amisi)$
$Ennemi(Rwanda,RDC)$



Chaînage arrière (Cont.)

$Congolais(x) \wedge Arme(y) \wedge Vend(x,y,z) \wedge Hostile(z) \rightarrow Criminel(x)$
$\forall x\ Missile(x) \wedge Possede(Rwanda,x) \rightarrow Vend(Amisi,x,Rwanda)$
$Missile(x) \rightarrow Arme(x)$
$Ennemi(x,RDC) \rightarrow Hostile(x)$

$Possede(Rwanda,M_1)$ et $Missile(M_1)$
$Congolais(Amisi)$
$Ennemi(Rwanda,RDC)$

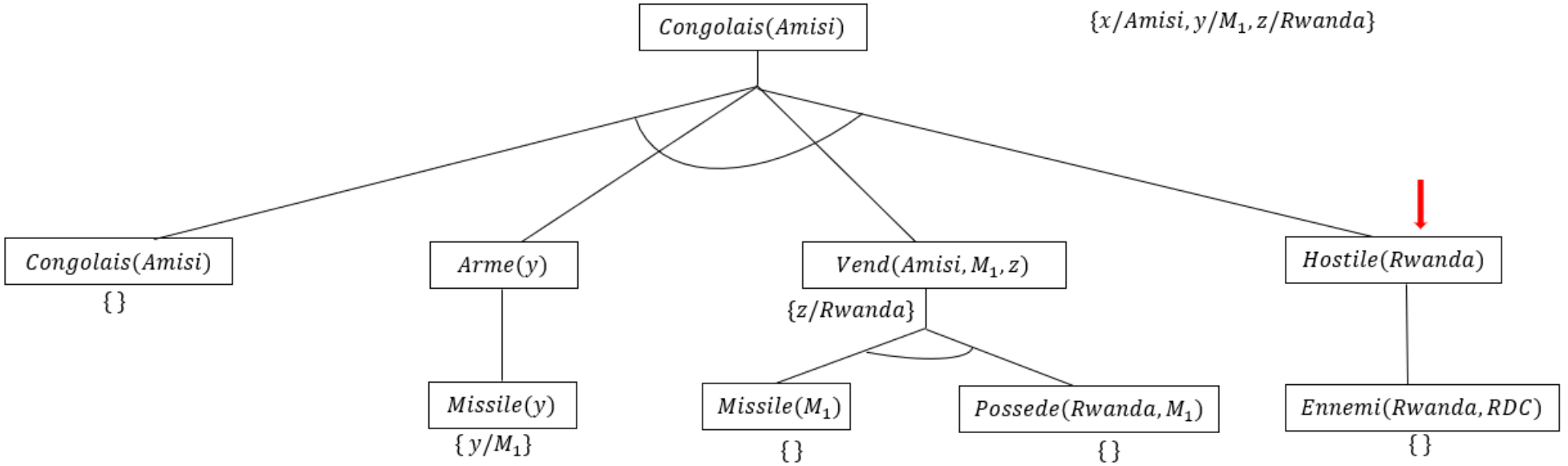


- On constate pour la prémisse Vend(x,y,z) qu'il existe une implication dans la KB
- On va ainsi remplacer x par Amisi, y par M_1 et z par Rwanda car pour correspondre les deux expressions (celle du graphe et l'implication de la deuxième règle de la KB), il faut que z soit remplacer par Rwanda
- Avec ce remplacement, nous obtenons deux nouveaux faits à ajouter dans notre KB ($Missile(M_1)$ et $Possede(Rwanda,M_1)$)
- On sait déjà que $Missile(M_1)$ est vrai. Ainsi, on doit aussi confirmer que $Possede(Rwanda,M_1)$ est vrai.

Chaînage arrière (Cont.)

$Congolais(x) \wedge Arme(y) \wedge Vend(x, y, z) \wedge Hostile(z) \rightarrow Criminel(x)$
$\forall x\ Missile(x) \wedge Possede(Rwanda, x) \rightarrow Vend(Amisi, x, Rwanda)$
$Missile(x) \rightarrow Arme(x)$
$Ennemi(x, RDC) \rightarrow Hostile(x)$

$Possede(Rwanda, M_1)$ et $Missile(M_1)$
$Congolais(Amisi)$
$Ennemi(Rwanda, RDC)$



- On va maintenant prouver Hostile(z). On doit remplacer z pour avoir une correspondance avec la règle 4 de la KB.
- Cependant, z est déjà substitué par Rwanda. Ainsi, en remplaçant par Rwanda, on obtient l'implication Ennemi(Rwanda, RDC)
- En prouvant cette dernière branche de notre graphe, nous venons de prouver la première règle de notre KB.

Propriétés du chaînage arrière

- Recherche récursive de preuves en profondeur :
  - l'espace est linéaire en fonction de la taille de la preuve
- Incomplet en raison de boucles infinies
  - On peut corriger cela par vérification de l'objectif actuel par rapport à tous les objectifs de la pile
- Inefficace en raison des sous-buts répétés (succès et échec)
  - On peut corriger cela par la mise en cache des résultats précédents (nécessite de l'espace supplémentaire !)
- Le chaînage arrière est largement utilisé (sans améliorations) pour la programmation logique

Sommaire: Principales approches d'inférence logique du premier ordre

• Inférence basée sur la résolution (logique du premier ordre) :

- Réfutation complète pour les bases de connaissances générales
  - Peut être utilisé pour confirmer ou réfuter une phrase p
  - Ne permet pas de générer toutes les phrases impliquées
- Nécessite la réduction de la KB de la logique du premier ordre FOL en FNC
- Utilise une version généralisée de la règle d'inférence propositionnelle

• Chaînage avant:

- Utilise le Modus Ponens généralisé pour ajouter de nouvelles phrases atomiques
- Nécessite que la base de connaissances se présente sous la forme de clauses définies du premier ordre
- Utile pour les systèmes qui font des déductions au fur et à mesure que les informations affluent

• Chaînage arrière:

- Hérite des déclarations en (bleu) ci-dessus
- Travaille à rebours à partir d'une requête pour tenter de construire une preuve
- Peut souffrir d'états répétés et d'incomplétude
- Utile pour l'inférence basée sur les requêtes

- Notez que toutes ces méthodes sont des généralisations de leurs équivalents propositionnels (c'est-à-dire qu'elles sont levées).

Brève mention : Programmation logique

- Prolog est le langage de programmation logique le plus utilisé.
  - Base : chaînage arrière avec des clauses de Horn et des fonctions supplémentaires.
- Prolog est utilisé pour :
  - Systèmes experts, systèmes de langage naturel, compilateurs, etc.

Program = set of clauses = head : - $littéral_1, \ldots, littéral_n$.

criminel(x) : - congolais(X), arme(Y), vend(X,Y,Z), hostile(Z).

missile(M_1).

possede(Rwanda, M_1).

vend(Amisi, X, Rwanda) : - missile(X), possede(Rwanda, X).

arme(X) : - missile(X).

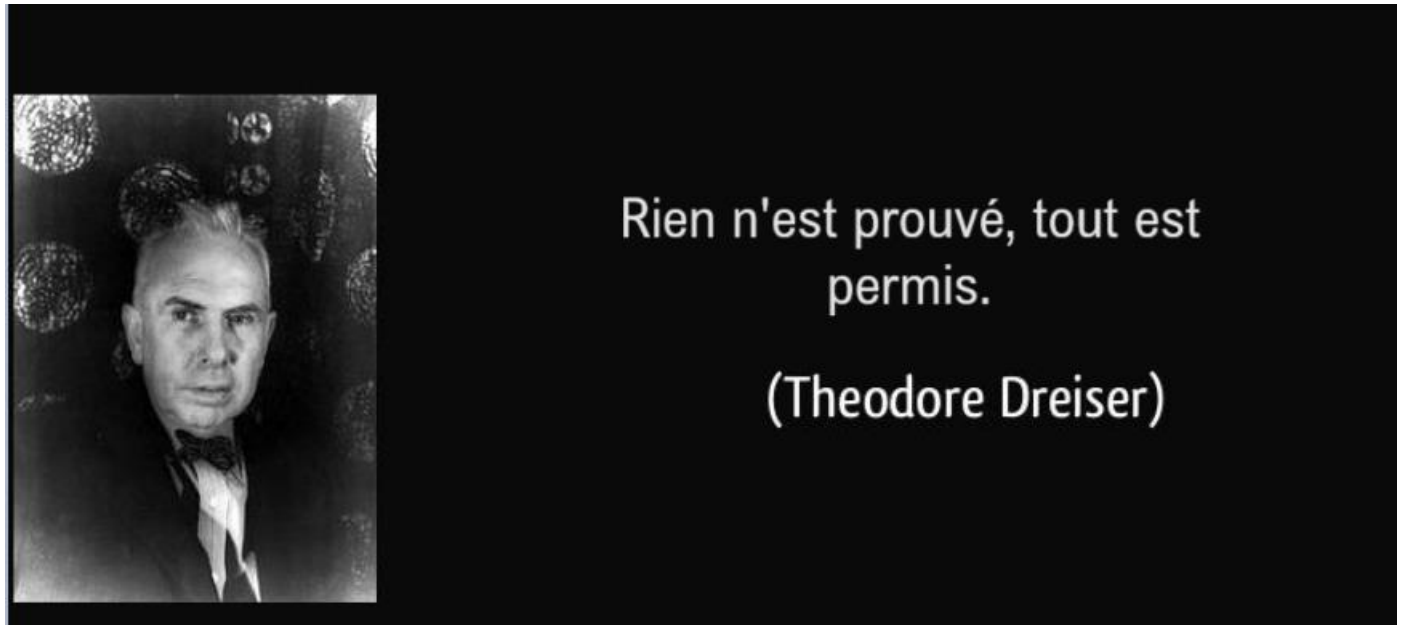
hostile(X) : - ennemi(X, RDC).

Congolais(Amisi).

Ennemi(Rwanda, RDC).

Résumé du raisonnement logique de premier ordre

- Règles d'inférence de la logique du premier ordre :
  - Opérateurs permettant de manipuler les phrases de la logique du premier ordre
- Preuves de théorèmes en logique du premier ordre :
  - Elégantes mais souvent peu pratiques
- Déduction via la propositionnalisation :
  - Simple (c'est-à-dire ramener la logique du premier ordre à la logique propositionnelle)
  - Mais inefficace
- Inférence sans propositionnalisation :
  - Avertissement pour l'inférence avec des expressions logiques du premier ordre
  - Unification
  - Modus Ponens Généralisé (MPG)
- Inférence en logique du premier ordre avec résolution :
  - Un système de preuve complet et efficace pour la logique du premier ordre
- Logique propositionnelle : ***Clauses de Horn et chaînage***:
  - Un travail inachevé en logique propositionnelle
  - Clauses de Horn vs Clauses définies
  - Chaînage avant et arrière (en logique propositionnelle)
- Logique du premier ordre: Inférence avec chaînage :
  - Clauses définies en logique du premier ordre
  - Logique du premier ordre avec chaînage avant : utilise le MPG pour dériver de nouveaux faits à partir de faits connus
  - Logique du premier ordre avec chaînage arrière : utilise les MPG pour prouver des objectifs à partir de faits connus


Rien n'est prouvé, tout est permis.
(Theodore Dreiser)

Travail pratique 2: Rappel

- **Objectifs :**

  - Logique propositionnelle et logique du premier ordre
    - S'entraîner à travailler avec la syntaxe et la sémantique de la logique
    - S'entraîner à manipuler des expressions logiques de manière "algébrique".
    - Appliquer la logique pour faire des déductions et déterminer la "vérité".
    - Comprendre deux des approches d'inférences les plus simples sur lesquelles nous nous concentrerons principalement dans ce cours :
      - Le chaînage avant
      - Le chaînage arrière