

---

# Systèmes à base de règles en logique des propositions

## Mécanismes fondamentaux

ML Mugnier

# Contexte : systèmes à base de connaissances

Base de  
connaissances



Services de  
raisonnement

- **Connaissances générales** du domaine d'application

ex : diagnostic médical

ex : fonctionnement système complexe

- **Connaissances factuelles (« faits »)**  
(observations, situations particulières, descriptions d'objets précis, ...)

ex : symptômes d'un patient précis

ex : données fournies par des capteurs

- **Déductif :**

Quelles conclusions peut-on tirer de la base de connaissances ?

⇒ services de haut niveau :

ex: faire le diagnostic médical d'un patient

ex: déterminer s'il y a des anomalies dans le système

Les connaissances sont exprimées dans un langage *basé sur* la logique et les raisonnements correspondent à des inférences logiques

# Systèmes à base de règles

- Base de connaissances = (Base de faits, Base de règles)
- Forme générale d'une règle : SI <condition> ALORS <conclusion>  
« condition » appelée aussi « prémisses » ou « hypothèse »

Exemples (règles « certaines » : pas de probabilités)

$\text{VoitureSport} \wedge \text{JeuneConducteur} \rightarrow \text{TarifAssuranceElevé}$   
(logique des propositions, ou logique d'ordre 0)

$\text{VoitureSport} = \text{vrai} \wedge \text{Age} < 20 \rightarrow \text{TarifAssurance} = 4$   
(logique d'ordre 0+)

$\forall x. \text{VoitureSport}(x) \rightarrow \text{VehiculeARisque}(x)$

$\forall x. \forall y. \text{AgeDe}(x,y) \wedge y < 20 \rightarrow \text{ProfilARisque}(x)$

$\forall x. \forall y. \text{VehiculeARisque}(x) \wedge \text{ConducteurDe}(x,y) \wedge \text{ProfilARisque}(y) \rightarrow$   
 $\text{TarifAssurance}(y,x,4)$

(logique du premier ordre)

- En programmation logique (Prolog, Datalog, ASP...) on dit <tête> SI <corps>  
 $\text{TarifAssurance}(y,x,4) \text{ :- } \text{VehiculeARisque}(x), \text{ConducteurDe}(x,y), \text{ProfilARisque}(y)$

# Règles en logique des propositions

- Rappel :
  - littéral positif : atome (symbole propositionnel)
  - littéral négatif : négation d'un atome
- Règle conjonctive :  $\langle \text{conjonction de littéraux} \rangle \rightarrow \langle \text{littéral} \rangle$   
Ces règles correspondent aux clauses : disjonction de littéraux

$$H \rightarrow C$$
$$\equiv \neg H \vee C$$
- Règle (conjonctive) positive :  $\langle \text{conjonction d'atomes} \rangle \rightarrow \langle \text{atome} \rangle$   
Ces règles correspondent aux « clauses définies » :  
clauses avec exactement un littéral positif
- Fait = littéral  
Autrement dit : règle avec une condition vide (« toujours vraie »)
- Base de connaissances :  $K = (BF, BR)$ 
  - BF : ensemble de faits vu comme la conjonction des faits
  - BR : ensemble de règles vu comme la conjonction des règles

→ la formule logique associée à K est la conjonction des faits et des règles

# Exemple de base de connaissances (réunion d'amis)

---

$BF = \{ \text{Benoît}, \text{Cloé} \}$

$BR = \{ R_1 \dots R_8 \}$

$R_1 : \text{Benoît} \wedge \text{Djamel} \wedge \text{Emma} \rightarrow \text{Félix}$

$R_2 : \text{Gaëlle} \wedge \text{Djamel} \rightarrow \text{Amandine}$

$R_3 : \text{Cloé} \wedge \text{Félix} \rightarrow \text{Amandine}$

$R_4 : \text{Benoît} \rightarrow \text{Xéna}$

$R_5 : \text{Xéna} \wedge \text{Amandine} \rightarrow \text{Habiba}$

$R_6 : \text{Cloé} \rightarrow \text{Djamel}$

$R_7 : \text{Xéna} \wedge \text{Cloé} \rightarrow \text{Amandine}$

$R_8 : \text{Xéna} \wedge \text{Benoît} \rightarrow \text{Djamel}$

D'un point de vue logique, la base de connaissances correspond à la formule :  
 $\text{Benoît} \wedge \text{Cloé} \wedge R_1 \wedge \dots \wedge R_8$

# Application des règles (positives)

---

- Une règle  $R : H \rightarrow C$  est **applicable** à BF si  $H \subseteq BF$   
*Ici on voit H comme un ensemble d'atomes*
- Cette application est **utile** si  $C \notin BF$
- Appliquer R à BF consiste à ajouter C dans BF
- BF est **saturée** (par rapport à BR)  
si aucune application d'une règle de BR à BF n'est utile

$$BF = \{A, B, C\}$$

$$BR = \left\{ \begin{array}{l} R_1 : A \wedge B \rightarrow E \\ R_2 : C \wedge E \rightarrow D \end{array} \right\}$$

$$BF \text{ saturée} \quad \mathbf{BF^*} = BF \cup \{E, D\}$$

# Mécanismes principaux sur les règles

## ■ Chaînage avant :

- principe : appliquer les règles sur les faits pour produire de nouveaux faits
- la base de faits est saturée si on ne peut plus produire de nouveaux faits

$BF = \{A, B, C\}$

$BR = \left\{ \begin{array}{l} R_1 : A \wedge B \rightarrow E \\ R_2 : C \wedge E \rightarrow D \end{array} \right\}$

$BF^* = BF \cup \{E, D\}$

## ■ Chaînage arrière :

- principe : prouver un but (atome/littéral) en «remontant» le long des règles
- le but initial est prouvé lorsqu'on arrive à une liste de buts vide

But initial :  $D$  ?

Avec  $R_2$  :

Avec le fait  $C$  (vu comme  $\rightarrow C$ ) :

Avec  $R_1$  :

Avec le fait  $A$

Avec le fait  $B$

liste de buts

$\{D\}$
$\{C, E\}$
$\{E\}$
$\{A, B\}$
$\{B\}$
$\{\}$

# Exemple chaînage avant (réunion d'amis)

---

$BF = \{ \text{Benoît}, \text{Cloé} \}$

$BR = \{R_1 \dots R_8\}$

$R_1 : \text{Benoît} \wedge \text{Djamel} \wedge \text{Emma} \rightarrow \text{Félix}$

$R_2 : \text{Gaëlle} \wedge \text{Djamel} \rightarrow \text{Amandine}$

$R_3 : \text{Cloé} \wedge \text{Félix} \rightarrow \text{Amandine}$

$R_4 : \text{Benoît} \rightarrow \text{Xéna}$

$R_5 : \text{Xéna} \wedge \text{Amandine} \rightarrow \text{Habiba}$

$R_6 : \text{Cloé} \rightarrow \text{Djamel}$

$R_7 : \text{Xéna} \wedge \text{Cloé} \rightarrow \text{Amandine}$

$R_8 : \text{Xéna} \wedge \text{Benoît} \rightarrow \text{Djamel}$

Remarque : une règle s'applique (de façon utile) au plus une fois

$BF^* = ?$

$\{ B, C, X, D, A, H \}$



# Algorithme de chaînage avant (version naïve)

**Algorithme ForwardChaining (K)**

// Données :  $K = (BF, BR)$

**Début**

// Résultat :  $BF^* = BF$  saturée par  $BR$

Fin  $\leftarrow$  faux

$BF^* \leftarrow BF$

|BF|

**Pour toute** règle  $R \in BR$

|BR|

appliquée(R)  $\leftarrow$  faux // Remarque : une règle s'applique au plus une fois

**Tant que** non fin

Au plus |BR| (+1) tours de TQ

nouvFaits  $\leftarrow \emptyset$  // ensemble des nouveaux faits obtenus à cette étape

**Pour toute** règle  $(R : H \rightarrow C) \in BR$  telle que appliquée(R) = faux

**Si**  $H \subseteq BF^*$  // R est applicable

appliquée(R)  $\leftarrow$  vrai

// que l'application soit utile ou pas

**Si**  $C \notin (BF^* \cup \text{nouvFaits})$

// l'application de R est utile

Ajouter C à nouvFaits

Pour une règle R :

$(|H| + 1) \times \text{\#faits\_max}$

$\leq \text{taille}(R) \times (|BF| + |BR|)$

**Si** nouvFaits =  $\emptyset$

Fin  $\leftarrow$  vrai

**Sinon** Ajouter les éléments de nouvFaits à  $BF^*$

Total : |BF| + |BR| +

|BR| x taille(BR) x (|BF| + |BR|)

**Retourner**  $BF^*$

**Fin**

FC a-t-il une complexité polynomiale ?

Borner **grossièrement** sa complexité

$O(\text{taille}(K)^3)$

# Algorithme de chaînage avant (avec compteurs)

**Algorithme ForwardChaining2 (K)**

*// Données :  $K = (BF, BR)$*

**Début**

*// Résultat : BF saturée par BR*

àTraiter  $\leftarrow BF$

$BF^* \leftarrow BF$

**Pour toute** règle  $R \in BR$

compteur( $R$ )  $\leftarrow |\text{hypothèse}(R)|$  *// Nombre de littéraux de l'hypothèse de R*

**Tant que** àTraiter  $\neq \emptyset$

Retirer F de àTraiter

**Pour toute** règle  $(R : H \rightarrow C) \in BR$  telle que  $F \in H$

(1)

Décrémenter compteur( $R$ )

**Si** compteur( $R$ ) = 0

*// R est applicable*

**Si**  $C \notin BF^*$

*// l'application de R est utile*

(2)

Ajouter C à àTraiter

Ajouter C à  $BF^*$

Retourner  $BF^*$

**Fin**

## FC2 sur l'exemple « réunion d'amis »

àTraiter  $\leftarrow$  BF

BF\*  $\leftarrow$  BF

**Pour toute** règle  $R \in BR$

compteur(R)  $\leftarrow$  |hypothèse(R)|

**Tant que** àTraiter  $\neq \emptyset$

Retirer F de àTraiter

**Pour toute** règle  $(R : H \rightarrow C) \in BR$  telle que  $F \in H$

Décrémenter compteur(R)

**Si** compteur(R) = 0

**Si**  $C \notin BF^*$

Ajouter C à àTraiter

Ajouter C à BF\*

**Retourner** BF\*

BF = { Benoît, Cloé }

BR = {R<sub>1</sub> ... R<sub>8</sub>}

R<sub>1</sub> : Benoît  $\wedge$  Djamel  $\wedge$  Emma  $\rightarrow$  Félix

R<sub>2</sub> : Gaëlle  $\wedge$  Djamel  $\rightarrow$  Amandine

R<sub>3</sub> : Cloé  $\wedge$  Félix  $\rightarrow$  Amandine

R<sub>4</sub> : Benoît  $\rightarrow$  Xéna

R<sub>5</sub> : Xéna  $\wedge$  Amandine  $\rightarrow$  Habiba

R<sub>6</sub> : Cloé  $\rightarrow$  Djamel

R<sub>7</sub> : Xéna  $\wedge$  Cloé  $\rightarrow$  Amandine

R<sub>8</sub> : Xéna  $\wedge$  Benoît  $\rightarrow$  Djamel

# Algorithme de chaînage avant (avec compteurs)

**Algorithme ForwardChaining2 (K)**

// Données :  $K = (BF, BR)$

**Début**

// Résultat : BF saturée par BR

àTraiter  $\leftarrow$  BF

BF\*  $\leftarrow$  BF

**Pour toute** règle  $R \in BR$

compteur(R)  $\leftarrow$  |hypothèse(R)| // Nombre de littéraux de l'hypothèse de R

**Tant que** àTraiter  $\neq \emptyset$

Retirer F de àTraiter

**Pour toute** règle  $R : H \rightarrow C \in BR$  telle que  $F \in H$

(1)

Décrémenter compteur(R)

**Si** compteur(R) = 0

// R est applicable

**Si**  $C \notin BF^*$

// l'application de R est utile

(2)

Ajouter C à àTraiter

Ajouter C à BF\*

Retourner BF\*

**Fin**

FC2 peut s'implémenter en  $O(K)$  si :

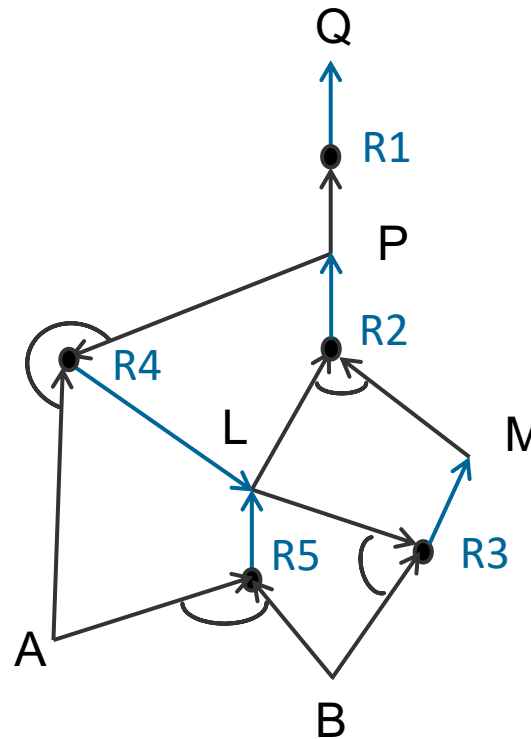
- en (1) accès **direct** à l'ensemble des règles telles que  $F \in H$
- en (2) test  $C \notin BF^*$  **en temps constant**

**Voir TD**

# Graphe « ET-OU » associé à la base de règles (BR)

BF = {A, B}

1.  $P \rightarrow Q$
2.  $L \wedge M \rightarrow P$
3.  $B \wedge L \rightarrow M$
4.  $A \wedge P \rightarrow L$
5.  $A \wedge B \rightarrow L$

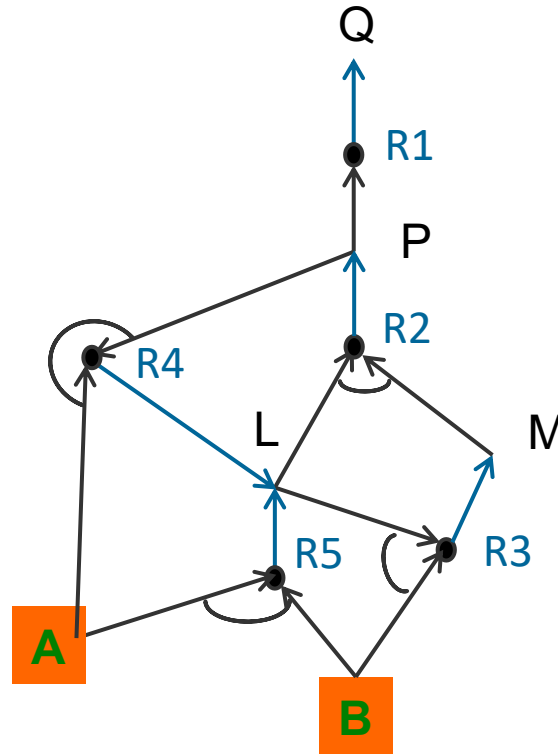


- 2 sortes de **sommets** :
  - symboles/littéraux de BR (sommets « OU »)
  - règles de BR (sommets « ET »)
- **arcs** pour chaque règle de BR :
  - sommets de l'hypothèse  $\rightarrow$  sommet règle
  - sommet règle  $\rightarrow$  sommet de la conclusion

# Chaînage avant sur le graphe ET-OU

BF = {A, B}

1.  $P \rightarrow Q$
2.  $L \wedge M \rightarrow P$
3.  $B \wedge L \rightarrow M$
4.  $A \wedge P \rightarrow L$
5.  $A \wedge B \rightarrow L$



Sommets ET (règles)  
« Pour utiliser R2,  
il faut L **et** M »

## Descente dans le graphe ET-OU :

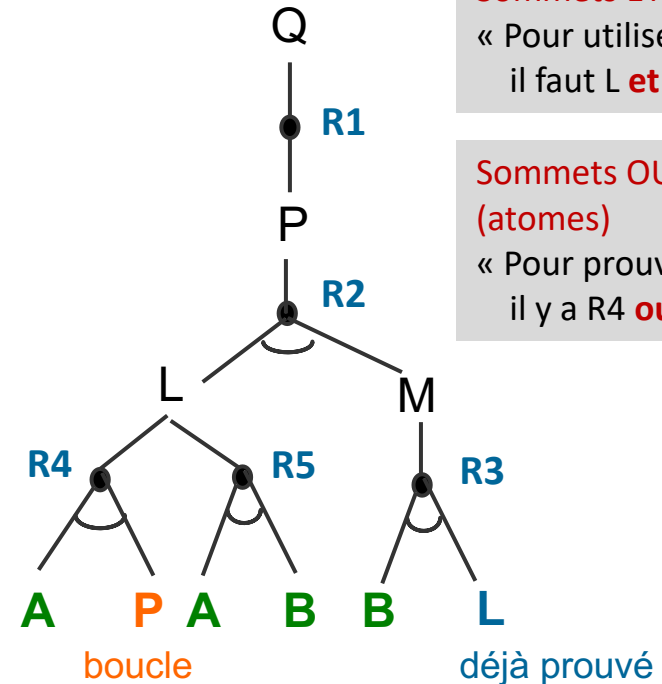
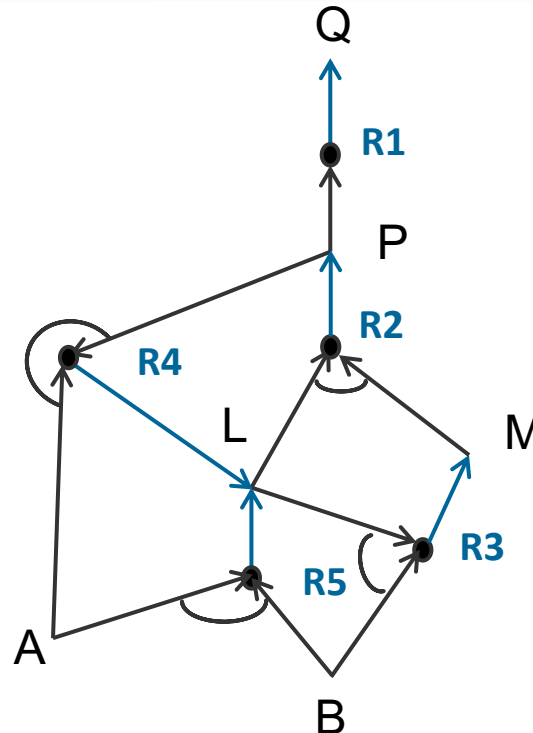
- Marquer les faits
  - Un sommet règle peut être franchi quand tous ses prédécesseurs sont marqués
- On marque alors son successeur

→ Base de faits **saturée** = ensemble des **sommets marqués**

# Chaînage arrière sur le graphe ET-OU

BF = {A, B}

1.  $P \rightarrow Q$
2.  $L \wedge M \rightarrow P$
3.  $B \wedge L \rightarrow M$
4.  $A \wedge P \rightarrow L$
5.  $A \wedge B \rightarrow L$



Sommets ET (règles)

« Pour utiliser R2,  
il faut L **et** M »

Sommets OU  
(atomes)

« Pour prouver L,  
il y a R4 **ou** R5 »

Partant d'un but à prouver, on construit un **arbre** en **remontant** le long des arcs

- Un **fait** est considéré comme **prouvé**
- Si **tous les fils** d'un noeud ET sont prouvés, alors le père de ce noeud est **prouvé**

**Eviter les boucles** : vérifier si le nouveau sous-but est déjà sur le chemin depuis la racine

**Eviter de refaire le même travail** : vérifier si le nouveau sous-but a **déjà été prouvé**

# Algorithme de chaînage arrière (version naïve)

Ici, les faits sont vus comme des règles à hypothèse vide

## Algorithme **BackwardChaining**(K,Q)

// Données :  $K = (BF, BR)$  et  $Q$  une liste de littéraux (ou symboles)

// Résultat (quand l'algorithme s'arrête) : vrai si  $Q$  prouvable, faux si  $Q$  non prouvable

**Début**

Si  $Q = \emptyset$ , retourner vrai

Soit  $C = \text{premier}(Q)$  //  $\text{premier}(Q)$  : premier littéral de  $Q$

**Pour** toute règle  $R = H_1 \wedge \dots \wedge H_n \rightarrow C$  de  $BR$  **et** **BF**

$Q' \leftarrow \text{Concaténer } [H_1, \dots, H_n] \text{ et } \text{reste}(Q)$  //  $\text{reste}(Q)$  :  $Q$  privé de son 1er littéral

**Si** **BC**(K,Q')

retourner vrai

Retourner faux

**Fin**

Problème : cet algorithme peut boucler



# BC sur l'exemple

## Algorithme **BC**(K,Q)

**Début**

**Si**  $Q = \emptyset$ , retourner vrai

Soit  $C = \text{premier}(Q)$

**Pour** toute règle  $R = H1 \wedge \dots \wedge Hn \rightarrow C$  de BR **et BF**

$Q' \leftarrow \text{Concaténer } [H1, \dots, Hn] \text{ et reste}(Q)$   
**Si** **BC**(K,Q')  
retourner vrai

Retourner faux

**Fin**

BF = {A, B}

1.  $P \rightarrow Q$
2.  $L \wedge M \rightarrow P$
3.  $B \wedge L \rightarrow M$
4.  $A \wedge P \rightarrow L$
5.  $A \wedge B \rightarrow L$

Q  
P  
L M  
A P M (par  $R_4$ )  
**P** M  
L M  
.....

# Algorithme de chaînage arrière (version alternative)

**Algorithme BC2(K,Q)** // ici Q est un littéral (et pas une liste de littéraux)

**Début**

**Si**  $Q \in \mathbf{BF}$ , retourner vrai

**Pour** toute règle  $R = H_1 \wedge \dots \wedge H_n \rightarrow Q$  de **BR**

$i \leftarrow 1$

**Tant que**  $i \leq n$  et **BC2**(K, $H_i$ )

incrémenter  $i$

**Si**  $i > n$ , retourner vrai // toute l'hypothèse de R prouvée, donc Q aussi

Retourner faux

**Fin**

Cet algorithme légèrement différent  
peut lui aussi boucler

# BC2 sur l'exemple

**Algorithme BC2(K,Q)** // ici Q est un littéral

**Début**

**Si**  $Q \in \mathbf{BF}$ , retourner vrai

**Pour** toute règle  $R = H1 \wedge \dots \wedge Hn \rightarrow Q$  de **BR**

$i \leftarrow 1$

**Tant que**  $i \leq n$  et **BC2**(K, $H_i$ )  
incrémenter  $i$

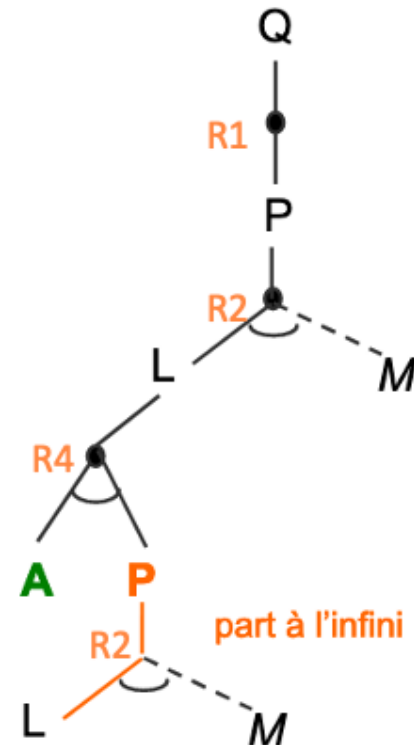
**Si**  $i > n$ , retourner vrai

Retourner faux

**Fin**

$\mathbf{BF} = \{A, B\}$

1.  $P \rightarrow Q$
2.  $L \wedge M \rightarrow P$
3.  $B \wedge L \rightarrow M$
4.  $A \wedge P \rightarrow L$
5.  $A \wedge B \rightarrow L$



# Algorithme de chaînage arrière (qui s'arrête)

**Algorithme BC3(K,Q,Lc)** // Données :  $K = (BF, BR)$ ,  $Q$  un atome  
//  $Lc$  un ensemble d'atomes (chemin de la racine à  $Q$ )  
// Résultat : vrai ssi  $Q$  est prouvable

**Début**

Appel BC3(K,Q,∅)

**Si**  $Q \in BF$ , retourner vrai

**Pour** toute règle  $R = H1 \wedge \dots \wedge Hn \rightarrow Q$  de  $BR$

**Si** aucun des  $H1 \dots Hn$  n'appartient à  $Lc$  // sinon on va boucler

$i \leftarrow 1$

**Tant que**  $i \leq n$  et **BC3**(K,  $H_i$ ,  $Lc \cup \{Q\}$ )

incrémenter  $i$

**Si**  $i > n$ , retourner vrai // hypothèse de  $R$  prouvée, donc  $Q$  aussi

Retourner faux // aucun des faits et aucune des règles ne permet de prouver  $Q$

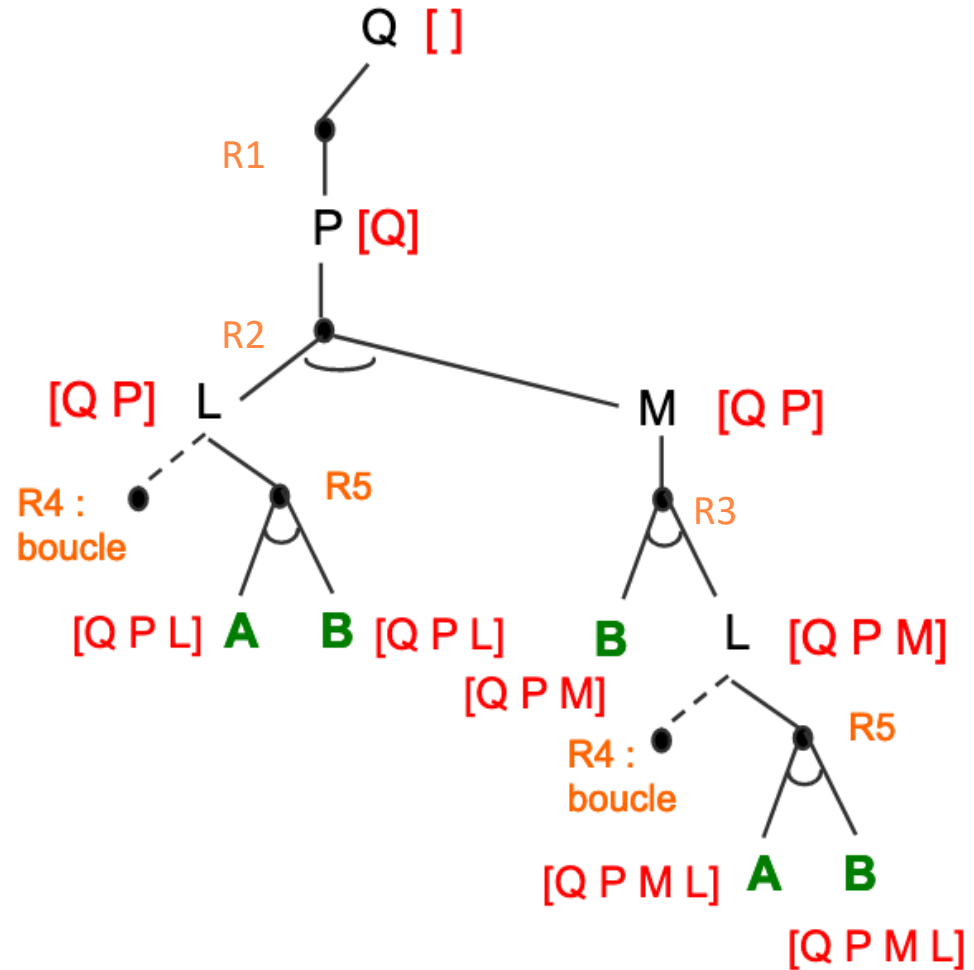
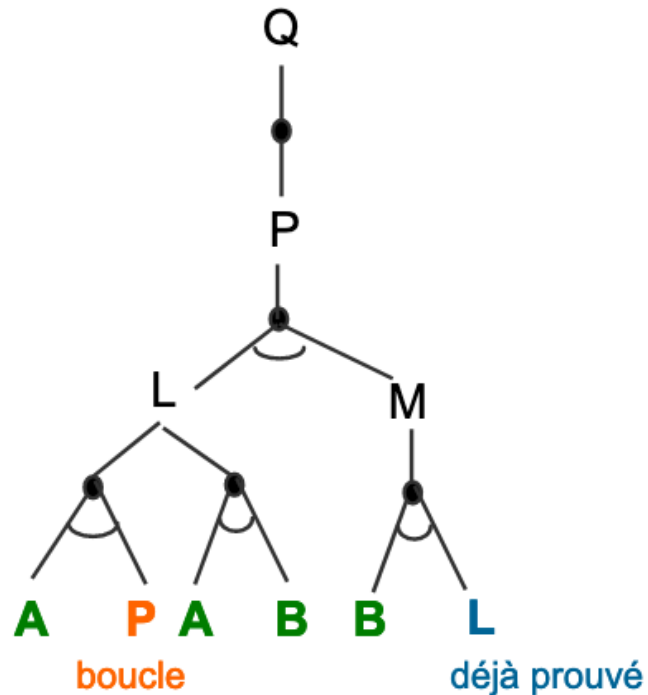
**Fin**

(Reste à « éviter de refaire le même travail »)

# BC3 sur l'exemple

BF = {A, B}

1.  $P \rightarrow Q$
2.  $L \wedge M \rightarrow P$
3.  $B \wedge L \rightarrow M$
4.  $A \wedge P \rightarrow L$
5.  $A \wedge B \rightarrow L$



Déjà prouvé ? Voir TD

# Synthèse

---

## ■ Règles conjonctives en logique des propositions

- *conjonction d'atomes*  $\rightarrow$  *atome* (cas positif)
- *conjonction de littéraux*  $\rightarrow$  *littéral*

## ■ Base de connaissances $K = (BF, BR)$

## ■ **Chaînage avant** : dirigé par les faits

applique les règles sur les faits jusqu'à obtention d'un point fixe  
 $\Rightarrow$  base de faits saturée :  $BF^*$

## ■ **Chaînage arrière** : dirigé par un but

cherche à prouver ce but en « remontant » les règles jusqu'aux faits

Pour tout littéral  $L$ ,

**$L$  s'obtient en chaînage avant** ( $L \in BF^*$ )

**si et seulement si**  $L$  est prouvable en chaînage arrière