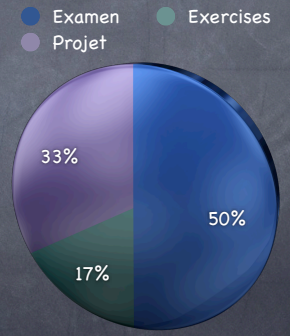
Systèmes Experts

Richard Moot
Richard.Moot@labri.fr

Systèmes Experts

- Exercices à faire et à rendre pendant le TD
- Projet: développement et programmation d'un mini système expert en Prolog
- Examen



Définition

- "un logiciel intelligent qui utilise des connaissances et des inférences logiques pour résoudre des problèmes qui sont suffisamment difficiles pour nécessiter une expertise humaine important pour trouver une solution" (Feigenbaum 1982)
- un système expert est un logiciel qui sait donner des recommandations — pour une domaine et une application bien défini — au même niveau d'un expert humain de ce domaine.

Systèmes Experts

Dans quelles situations peut-on utiliser des systèmes experts?

Systèmes Experts

- Quand on peut résoudre le problème par un appel téléphonique à un expert en moins que 15 minutes

Systèmes Experts

- Quand on peut résoudre le problème par un appel téléphonique à un expert en moins que 15 minutes

En plus de détail

- Les besoins justifient le coût de développement; le problème ne peut pas être résolu avec l'informatique traditionnel
- Le problème est bien structuré et ne dépend pas (trop) de bon sens "common sense"
- Il y a des experts qui sont disponibles et qui savent bien s'exprimer sur le sujet
- La taille du problème se prête bien à un système expert.

Systèmes Experts

Exemples

Exemples - Diagnose

dans le domaine médical, mais aussi en informatique

Exemples - Surveillance

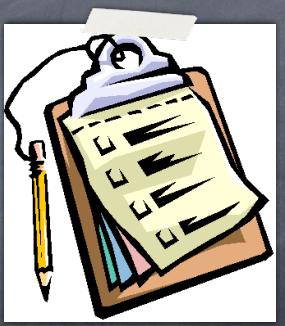
détection de fraudes avec cartes de crédit et téléphones, aide au surveillance continu des patients

Exemples - Surveillance

détection de fraudes avec cartes de crédit et téléphones, aide au surveillance continu des patients

Exemples - Planning/ Scheduling

gestion de personnel, machines, ressources



Exemples - Finance

aide a l'affectation du crédit immobilier ou du crédit à une entreprise, calcul des risques pour assurances, prédiction des marchés

Systèmes Experts

Architecture

Architecture de systèmes experts

A partir d'entretiens avec des experts, on extrait un base de données avec des connaissances spécifiques à ce domaine.

Architecture de systèmes experts

Généralement, ce sont des règles du type:
si on affecte du personnel pour la garde de nuit et si c'est la soirée d'une fête nationale alors assure la présence d'un médecin en plus que pour une nuit normale

Architecture de systèmes experts

Généralement, ce sont des règles du type:
si plusieurs retraits de plus de 100 euros chacun sont fait et si ces retrait sont fait dans des différents pays alors une utilisation frauduleuse est très probable (probabilité 0.9)

Architecture de systèmes experts

Généralement, ce sont des règles du type:
si p et q alors r
(avec probabilité X)

C'est à dire des formules logiques éventuellement avec des probabilités

Architecture de systèmes experts

Le système d'inférences logiques établit une conclusion grâce à l'information fourni par l'utilisateur et au base des connaissances.

Architecture de systèmes experts

L'avantage ce façon d'organiser un système expert est qu'on a besoin de changer que la base de connaissances pour obtenir un système expert dans un autre domaine

Architecture de systèmes experts

L'interaction avec l'utilisateur prend typiquement le forme d'une dialogue, où le système expert pose des questions pour aider à trouver la bonne solution

Architecture de systèmes experts

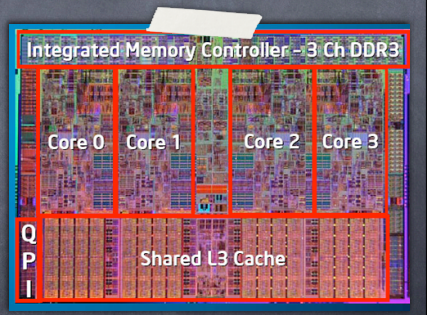
Idéalement, l'utilisateur peut demander le système expert d'expliquer son raisonnement: pourquoi pose-t-il cette question? comment est-il arrivé à cette conclusion?

Architecture des systèmes experts

- La base de données des connaissances ainsi que le système d'inférence sont fondés sur la logique et l'inférence logique,
- Alors, on a besoin de commencer par une petite introduction en logique et déduction automatique,
- On finira par voir comment ces principes donnent un langage de programmation: PROLOG, PROgrammation en LOGique.

Logique des Propositions

- Littéralement au coeur des ordinateurs



Logique des Propositions Classique

- On coupe le monde en petit morceaux, des phrases simples comme "il fait beau" ou "le soleil gravite autour de la terre" qui sont soit vrai (indiqué par la valeur 1) soit faux (indiqué par la valeur 0).
- Comme abstraction, on remplace ces phrases par des lettres p,q,r,... qu'on appelle des propositions atomiques ou des formulas atomiques.

Logique des Propositions Classique — Syntaxe

1. Une proposition atomique est une proposition,
2. Si X est une proposition (pas nécessairement atomique) alors ¬X — "non X" ou "la négation de X" — est une proposition,

Logique des Propositions Classique — Syntaxe

3. Si X et Y sont des propositions alors,

   3.a. $(X \wedge Y)$ "X et Y"

   3.b. $(X \vee Y)$ "X ou Y"

   3.c. $(X \rightarrow Y)$ "si X alors Y"

   3.d. $(X \leftrightarrow Y)$ "X si et seulement si Y"

sont aussi des propositions

Logique des Propositions Classique — Syntaxe

4. Rien d'autre n'est une proposition.

Logique des Propositions Classique — Syntaxe

Grâce a cette définition on peut démontrer que

$(p \wedge \neg q)$

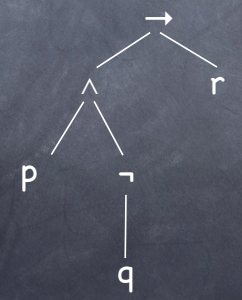
est une proposition mais aussi que

$(p \rightarrow qr)$

n'est pas une proposition.

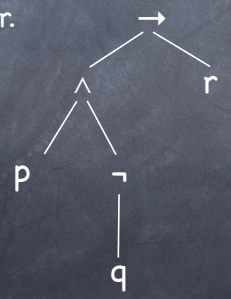
Logique des Propositions Classique — Syntaxe

Une formule correspond naturellement à un arbre, par exemple le formule $(p \wedge \neg q) \rightarrow r$ correspond à l'arbre:



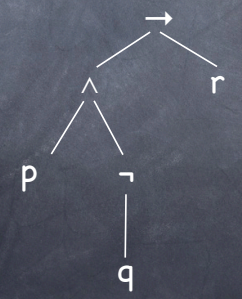
Logique des Propositions Classique — Syntaxe

On appelle les noeuds dans cet arbre les sous-formules de ce formule. Alors p, q, r, $\neg q$, $p \wedge \neg q$ et $(p \wedge \neg q) \rightarrow r$ même sont tous de sous-formules de $(p \wedge \neg q) \rightarrow r$.



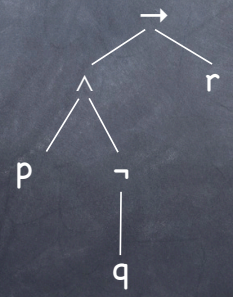
Logique des Propositions Classique — Modélisation

On assume d'avoir une méthode pour vérifier si une proposition atomique est vrai ou faux.



Logique des Propositions Classique — Modélisation

p "le feu est vert"
q "il y un piéton en train de traverser"
r "j'avance au vitesse maximale"



Interprétation des Propositions Complexes

- L'interprétation d'un formule complexe dépend que de l'interprétation de ses sous-formules directs.
- Alors, étant donné une interprétation pour les prépositions atomiques, on peut trouver

¬X	X
0	1
1	0

¬: négation logique

X∧Y	X	Y
0	0	0
0	0	1
0	1	0
1	1	1

∧: "et" logique

X∨Y	X	Y
0	0	0
1	0	1
1	1	0
1	1	1

∨: "ou" logique

X∨Y	X	Y
0	0	0
1	0	1
1	1	0
1	1	1

Remarque
le "ou" logique est inclusif, aux menus "dessert ou fromage" et "vin ou café" ne veulent pas dire "aux moins un des deux", mais plutôt "exactement un"

X→Y	X	Y
1	0	0
1	0	1
0	1	0
1	1	1

→: implication logique

X→Y	X	Y
1	0	0
1	0	1
0	1	0
1	1	1

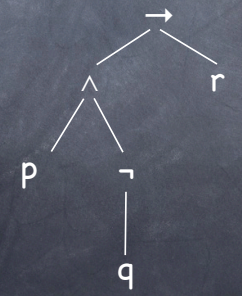
L'implication logique est peut-être le connecteur le plus difficile à comprendre. Aucun causalité entre X et Y ne peut être déduit! X→Y est faux que quand X est vrai et Y est faux et vrai dans tous les autres cas.

X↔Y	X	Y
1	0	0
0	0	1
0	1	0
1	1	1

↔: "équivalence" logique

Logique des Propositions Classique — Modélisation

p "le feu est vert" vrai
q "il y un piéton en train de traverser" vrai
r "j'avance à la vitesse maximale" faux

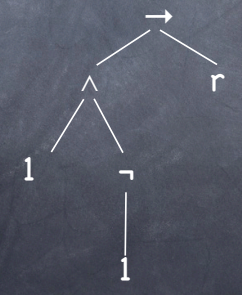


Logique des Propositions Classique — Modélisation

p "le feu est vert" vrai
q "il y un piéton en train de traverser" vrai
r "j'avance à la vitesse maximale" faux

Logique des Propositions Classique — Modélisation

p "le feu est vert" vrai
q "il y un piéton en train de traverser" vrai
r "j'avance à la vitesse maximale" faux

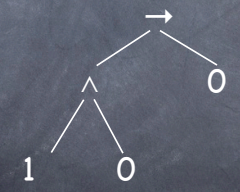


Logique des Propositions Classique — Modélisation

p "le feu est vert" vrai
q "il y un piéton en train de traverser" vrai
r "j'avance à la vitesse maximale" faux

Logique des Propositions Classique — Modélisation

p "le feu est vert" vrai
q "il y un piéton en train de traverser" vrai
r "j'avance à la vitesse maximale" faux



Logique des Propositions Classique — Modélisation

p "le feu est vert" vrai
q "il y un piéton en train de traverser" vrai
r "j'avance à la vitesse maximale" faux

Logique des Propositions Classique — Modélisation

p "le feu est vert" vrai
q "il y un piéton en train de traverser" vrai
r "j'avance à la vitesse maximale" faux
1

Alors, $(p \wedge \neg q) \rightarrow r$ est vrai

Vérifier tout les valeurs de vérité

- Des fois on s'intéresse à savoir pour quels valeurs de vérité des formules atomiques un formule complexe est vrai ou faux,
- S'il se trouve qu'un formule X n'est jamais vrai, on parle d'une contradiction
- Sinon on le formule X est valide,
- De plus, s'il se trouve X est toujours vrai, on parle d'un tautologie

Vérifier tout les valeurs de vérité

- Par contre, décider si un formule est valide, un tautologie ou une contradiction est difficile.
- Car, pour **a** formules atomiques, il y a 2^a combinaisons à vérifier.

Algorithme: comment construire un table de vérité pour un formule

1. On fait un table avec:
   - 1 ligne pour chaque combinaison de valeurs de vérité pour les formules atomique
   - 1 colonne pour chaque sous-formule de la formule.
2. On remplit les lignes et les colonnes correspondant aux formules atomique en énumérant toutes les possibilités.

Algorithme: comment construire un table de vérité pour un formule

3. Maintenant pour chaque formule X dont on a énuméré tout les possibilité de valeurs de ses sous-formules direct, on remplit la colonne qui correspond a X grâce à la définition du table de vérité pour ce connecteur.
4. Continue jusqu'au moment où on a rempli tous les champs du table.

tautologie

- L'exemple le plus simple possible: un proposition atomique, p, et une implication.
- p peut être soit faux (0) soit vrai (1).

p→p	p
	0
	1

tautologie

- Il nous reste à remplir la colonne p→p.
- On vérifie la table de vérité pour X→Y. Pour p→p on a juste deux cas.
  - X = 1 et Y = 1, qui donne 1
  - X = 0 et Y = 0, qui donne 1

p→p	p
	0
1	1

X→Y	X	Y
1	0	0
1	0	1
0	1	0
1	1	1

tautologie

- Il nous reste à remplir la colonne p→p.
- On vérifie la table de vérité pour X→Y. Pour p→p on a juste deux cas.
  - X = 1 et Y = 1, qui donne 1
  - X = 0 et Y = 0, qui donne 1

p→p	p
1	0
1	1

X→Y	X	Y
1	0	0
1	0	1
0	1	0
1	1	1

tautologie

- Il nous reste à remplir la colonne p→p.
- On vérifie la table de vérité pour X→Y. Pour p→p on a juste deux cas.
  - X = 1 et Y = 1, qui donne 1
  - X = 0 et Y = 0, qui donne 1
- Alors on est terminé et p→p est un tautologie

p→p	p
1	0
1	1

X→Y	X	Y
1	0	0
1	0	1
0	1	0
1	1	1

tautologie

- Encore une exemple, cette fois avec négation et disjonction: p∨¬p
- On commence comme avant, énumérant les possibilités pour p et en faisant un colonne pour chaque sous-formule: p, ¬p et p∨¬p

p∨¬p	¬p	p
		0
		1

tautologie

- La formule dont on sait les possibilités de ses sous-formules est ¬p (car on vient de traiter p)
- Grâce au table de vérité pour la négation on remplit la colonne ¬p

¬X	X
0	1
1	0

p∨¬p	¬p	p
	1	0
	0	1

tautologie

- Maintenant on sait les valeur de vérité pour les deux sous-formules de p∨¬p: p et ¬p
- En utilisant la table on trouve 1∨0 = 1 et 0∨1 = 1
- Alors p∨¬p est un tautologie

X∨Y	X	Y
0	0	0
1	0	1
1	1	0
1	1	1

p∨¬p	¬p	p
1	1	0
1	0	1

p∨¬p	¬p	p
1	1	0
1	0	1

tautologie

p∧¬p	¬p	p
		0
		1

contradiction

p∧¬p	¬p	p
	1	0
	0	1

contradiction

p∧¬p	¬p	p
0	1	0
0	0	1

contradiction

(p∧¬q)→r	p∧¬q	¬q	p	q	r
			0	0	0
			0	0	1
			0	1	0
			0	1	1
			1	0	0
			1	0	1
			1	1	0
			1	1	1

Exemple: (p∧¬q)→r

$(p\land\neg q)\to r$	$p\land\neg q$	$\neg q$	p	q	r
			0	0	0
			0	0	1
			0	1	0
			0	1	1
			1	0	0
			1	0	1
			1	1	0
			1	1	1

Exemple: $(p\land\neg q)\to r$

$(p\land\neg q)\to r$	$p\land\neg q$	$\neg q$	p	q	r
		1	0	0	0
		1	0	0	1
		0	0	1	0
		0	0	1	1
		1	1	0	0
		1	1	0	1
		0	1	1	0
		0	1	1	1

Exemple: $(p\land\neg q)\to r$

$(p\land\neg q)\to r$	$p\land\neg q$	$\neg q$	p	q	r
		1	0	0	0
		1	0	0	1
		0	0	1	0
		0	0	1	1
		1	1	0	0
		1	1	0	1
		0	1	1	0
		0	1	1	1

Exemple: $(p\land\neg q)\to r$

$(p\land\neg q)\to r$	$p\land\neg q$	$\neg q$	p	q	r
		1	0	0	0
		1	0	0	1
		0	0	1	0
		0	0	1	1
		1	1	0	0
		1	1	0	1
		0	1	1	0
		0	1	1	1

Exemple: $(p\land\neg q)\to r$

$(p\wedge\neg q)\to r$	$p\wedge\neg q$	$\neg q$	p	q	r
	0	1	0	0	0
	0	1	0	0	1
	0	0	0	1	0
	0	0	0	1	1
	1	1	1	0	0
	1	1	1	0	1
	0	0	1	1	0
	0	0	1	1	1

Exemple: $(p\wedge\neg q)\to r$

$(p\wedge\neg q)\to r$	$p\wedge\neg q$	$\neg q$	p	q	r
	0	1	0	0	0
	0	1	0	0	1
	0	0	0	1	0
	0	0	0	1	1
	1	1	1	0	0
	1	1	1	0	1
	0	0	1	1	0
	0	0	1	1	1

Exemple: $(p\wedge\neg q)\to r$

$(p\wedge\neg q)\to r$	$p\wedge\neg q$	$\neg q$	p	q	r
	0	1	0	0	0
	0	1	0	0	1
	0	0	0	1	0
	0	0	0	1	1
	1	1	1	0	0
	1	1	1	0	1
	0	0	1	1	0
	0	0	1	1	1

Exemple: $(p\wedge\neg q)\to r$

$(p\wedge\neg q)\to r$	$p\wedge\neg q$	$\neg q$	p	q	r
1	0	1	0	0	0
1	0	1	0	0	1
1	0	0	0	1	0
1	0	0	0	1	1
0	1	1	1	0	0
1	1	1	1	0	1
1	0	0	1	1	0
1	0	0	1	1	1

Exemple: $(p\wedge\neg q)\to r$

$(p \wedge \neg q) \rightarrow r$	$p \wedge \neg q$	$\neg q$	p	q	r
1	0	1	0	0	0
1	0	1	0	0	1
1	0	0	0	1	0
1	0	0	0	1	1
0	1	1	1	0	0
1	1	1	1	0	1
1	0	0	1	1	0
1	0	0	1	1	1

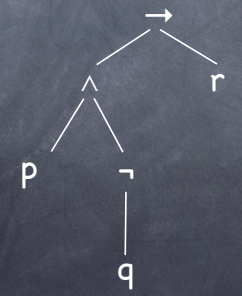
Exemple: $(p \wedge \neg q) \rightarrow r$

Logique des Propositions Classique — Modélisation

p "le feu est vert"

q "il y un piéton en train de traverser"

r "j'avance au vitesse maximale"

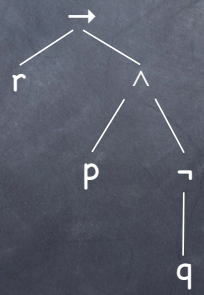


Logique des Propositions Classique — Modélisation

p "le feu est vert"

q "il y un piéton en train de traverser"

r "j'avance au vitesse maximale"



$r \rightarrow (p \wedge \neg q)$	$p \wedge \neg q$	$\neg q$	p	q	r
			0	0	0
			0	0	1
			0	1	0
			0	1	1
			1	0	0
			1	0	1
			1	1	0
			1	1	1

Exemple: $r \rightarrow (p \wedge \neg q)$

r→(p∧¬q)	p∧¬q	¬q	p	q	r
	0	1	0	0	0
	0	1	0	0	1
	0	0	0	1	0
	0	0	0	1	1
	1	1	1	0	0
	1	1	1	0	1
	0	0	1	1	0
	0	0	1	1	1

Exemple: r→(p∧¬q)

r→(p∧¬q)	p∧¬q	¬q	p	q	r
1	0	1	0	0	0
1	0	1	0	0	1
1	0	0	0	1	0
1	0	0	0	1	1
0	1	1	1	0	0
1	1	1	1	0	1
1	0	0	1	1	0
1	0	0	1	1	1

Exemple: r→(p∧¬q)

r→(p∧¬q)	p∧¬q	¬q	p	q	r
1	0	1	0	0	0
0	0	1	0	0	1
1	0	0	0	1	0
0	0	0	0	1	1
1	1	1	1	0	0
1	1	1	1	0	1
1	0	0	1	1	0
0	0	0	1	1	1

Exemple: r→(p∧¬q)

Interprétation

- Alors, on vient de déduire que la proposition r→(p∧¬q) est faux si et seulement si
  - r est vrai et p est faux "le conducteur avance en vitesse mais le feux n'est pas vert" (s'il y a un piéton ou pas)
  - r et q sont vrais: "le conducteur avance en vitesse quand il y a un piéton" (si le feu est rouge ou vert

Conclusion

- Quand on veut modéliser le monde en logique, c'est facile de se tromper!
- C'est un des raisons pour faire la validation: faire des tests (à la main ou par ordinateur) pour vérifier que nos formules correspondent bien aux intuitions et intentions de l'expert.

Algorithme de Quine

- les constants T,⊥
- substitution des formules
- équivalences
- l'algorithme

Algorithme de Quine

Les constants T et ⊥

- T, "vrai", est un constante qui a toujours la valeur de vérité 1
- ⊥, "faux", est un constante qui a toujours la valeur de vérité 0

Algorithme de Quine

Lemme: Substitution

Soit
X un formule,
Y un des sous-formules de X,
Z un formule tel que Y↔Z,

alors quand on remplace toutes les occurrences de Y dans X par Z, dénoté par X[Y := Z], X est équivalent à X[Y := Z], c'est à dire

$$X \leftrightarrow X[Y := Z]$$

Algorithme de Quine
Equivalences

Définitions de $\rightarrow$ et $\leftrightarrow$

- $(X \rightarrow Y) \equiv (\neg X \vee Y)$
- $(X \leftrightarrow Y) \equiv ((X \rightarrow Y) \wedge (Y \rightarrow X))$

Algorithme de Quine
Equivalences

Définitions de $\rightarrow$ et $\leftrightarrow$

- $(X \rightarrow Y) \equiv (\neg X \vee Y)$
- $(X \leftrightarrow Y) \equiv ((X \rightarrow Y) \wedge (Y \rightarrow X)) \equiv ((\neg X \vee Y) \wedge (\neg Y \vee X))$

Algorithme de Quine
Equivalences

Commutativité de $\wedge$ et $\vee$

- $(X \wedge Y) \equiv (Y \wedge X)$
- $(X \vee Y) \equiv (Y \vee X)$

Algorithme de Quine
Equivalences

Associativité de $\wedge$ et $\vee$

- $((X \wedge Y) \wedge Z) \equiv (X \wedge (Y \wedge Z))$
- $((X \vee Y) \vee Z) \equiv (X \vee (Y \vee Z))$

Algorithme de Quine Equivalences

Distributivité de $\wedge$ et $\vee$

- $((X \wedge Y) \vee Z) \equiv ((X \vee Z) \wedge (Y \vee Z))$
- $((X \vee Y) \wedge Z) \equiv ((X \wedge Z) \vee (Y \wedge Z))$

Algorithme de Quine Equivalences

Règles de De Morgan

- $\neg T \equiv \bot$
- $\neg \bot \equiv T$
- $\neg\neg X \equiv X$
- $\neg(X \wedge Y) \equiv (\neg X \vee \neg Y)$
- $\neg(X \vee Y) \equiv (\neg X \wedge \neg Y)$

Algorithme de Quine Equivalences

Simplifications de Quine

- $(T \rightarrow X) \equiv X$
- $(X \rightarrow T) \equiv T$
- $(\bot \rightarrow X) \equiv T$
- $(X \rightarrow \bot) \equiv \neg X$

Algorithme de Quine Equivalences

Simplifications de Quine

- $(T \vee X) \equiv T$
- $(X \vee T) \equiv T$
- $(\bot \vee X) \equiv X$
- $(X \vee \bot) \equiv X$

Algorithme de Quine Equivalences

Simplifications de Quine

- $(T \wedge X) \equiv X$
- $(X \wedge T) \equiv X$
- $(\bot \wedge X) \equiv \bot$
- $(X \wedge \bot) \equiv \bot$

Algorithme de Quine

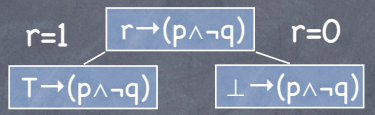
L'algorithme

1. Initialiser par l'arbre simple qui contient que la formule a vérifier comme feuille (et racine).
2. Applique les simplifications, ajoutant une feuille pour chaque simplification. Continue jusqu'au moment qu'aucune simplification ne peut être fait.
3. Trouve un feuille qui contient un formule atomique, disons X avec sous-formule atomique p. Ajout deux feuilles X[p := T] et X[p :=⊥]. Continue avec 2.

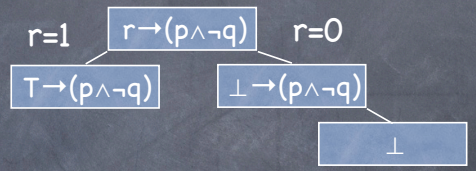
Exemple de l'algorithme de Quine

$r \rightarrow (p \wedge \neg q)$

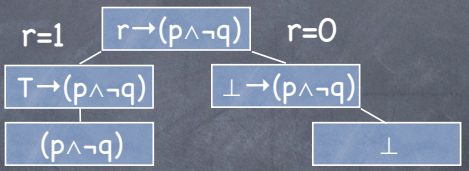
Exemple de l'algorithme de Quine



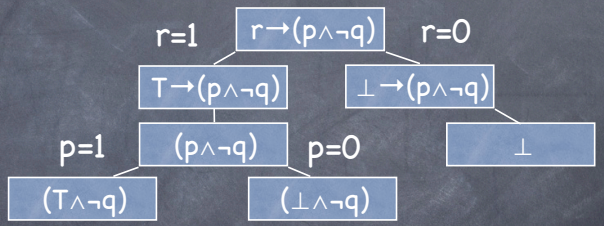
Exemple de l'algorithme de Quine



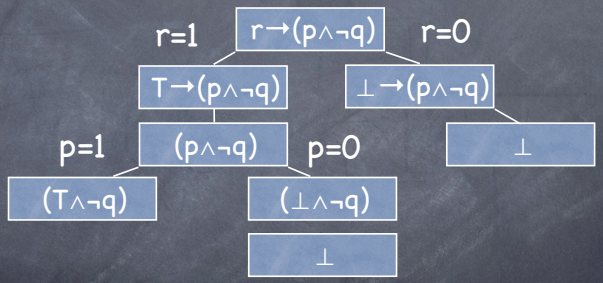
Exemple de l'algorithme de Quine



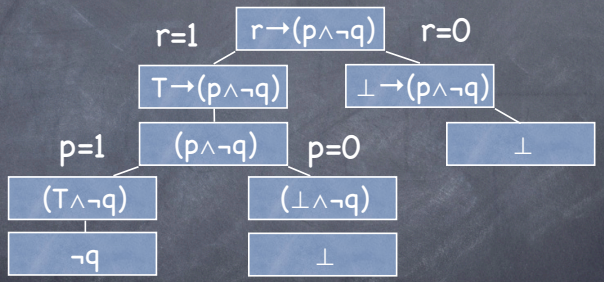
Exemple de l'algorithme de Quine



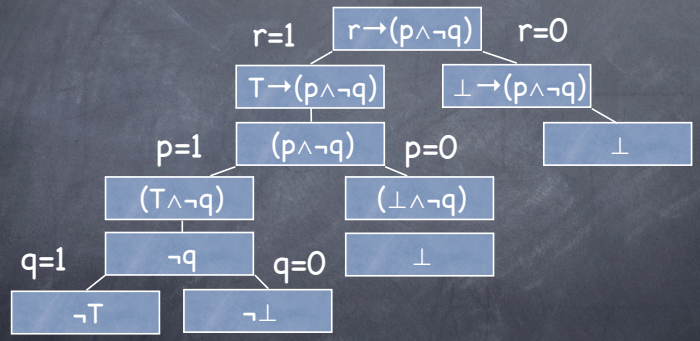
Exemple de l'algorithme de Quine



Exemple de l'algorithme de Quine



Exemple de l'algorithme de Quine



Exemple de l'algorithme de Quine