# Algorithme de Quine

Lycée Thiers

# Crédits

1. Un cours de mon collègue Q. Fortier

# Crédits

1. Un cours de mon collègue Q. Fortier
2. Ce cours de l'université de Montpellier : ici

# Substitution

- Notation : Si $x$ est une variable, la notation $[x \leftarrow T]$ désigne le contexte partiel (dégénéré) qui remplace $x$ par $T$ (True) et n'affecte aucune autre variable.
  Mais je croyais qu'un contexte affectait 0 ou 1 à une variable !
  WTF ! ! ? ? ?
  Et bien si une variable a pour image $T$ (resp. $F$) par un contexte (dégénéré) alors appliquer au résultat un contexte au sens du cours lui affecte 1 (resp. 0).

# Substitution

- Notation : Si $x$ est une variable, la notation $[x \leftarrow T]$ désigne le contexte partiel (dégénéré) qui remplace $x$ par $T$ (True) et n'affecte aucune autre variable.
- La notation $[x \leftarrow T][y \leftarrow F]$ désigne le contexte partiel (dégénéré) qui remplace $x$ par $T$ (True) ; $y$ par $F$ (False) et n'affecte aucune autre variable. De même :

$$\prod_{i=1}^{n} [x_i \leftarrow C_i]$$

est le contexte partiel (dégénéré) qui remplace chaque $x_i$ par la constante $C_i$ (vrai ou faux).

# Substitution

- Notation : Si $x$ est une variable, la notation $[x \leftarrow T]$ désigne le contexte partiel (dégénéré) qui remplace $x$ par $T$ (True) et n'affecte aucune autre variable.
- La notation $[x \leftarrow T][y \leftarrow F]$ désigne le contexte partiel (dégénéré) qui remplace $x$ par $T$ (True) ; $y$ par $F$ (False) et n'affecte aucune autre variable. De même :

$$\prod_{i=1}^{n}[x_i \leftarrow C_i]$$

  est le contexte partiel (dégénéré) qui remplace chaque $x_i$ par la constante $C_i$ (vrai ou faux).
- Notation : Si $P$ est une proposition et $x$ une variable, $P[x \leftarrow T]$ désigne $P$ dans laquelle toutes les occurrences de $x$ sont remplacées par $T$ (True). Et $P\prod_{i=1}^{n}[x_i \leftarrow C_i]$... à votre avis ?

# Satisfiabilité par backtracking

- On dispose de

`evaluer(`$P$`:prop., `$\mu$`:ctxt complet (dégénéré)) : bool`

Listing 1 – Algorithme de satisfiabilité

```
1  fonction sat(P : proposition;
2               L : ensemble de variables àremplacer par T/F,
3               μ : contexte partiel (dégénéré)
4                        n'agissant pas sur L)
5           sortie : un bouléen
6           si L est vide renvoyer evaluer (P,μ)
7           prendre x ∈ L
8           si sat (P,L \ {x},μ ∘ [x ← T])
9              alors renvoyer Vrai
10          sinon renvoyer sat (P, L \ {x},μ ∘ [x ← F])
```

Cette fonction réalise donc un bactracking et teste potentiellement $2^{\text{nb de variables de } P}$ contextes (dégénérés).

# Satisfiabilité par backtracking

- On dispose de

`evaluer(`$P$`:prop., `$\mu$`:ctxt complet (dégénéré)) : bool`

Listing 2 – Algorithme de satisfiabilité

```
1  fonction sat(P : proposition;
2               L : ensemble de variables àremplacer par T/F,
3               μ : contexte partiel (dégénéré)
4                        n'agissant pas sur L)
5           sortie : un bouléen
6           si L est vide renvoyer evaluer (P,μ)
7           prendre x ∈ L
8           si sat (P,L \ {x},μ ∘ [x ← T])
9              alors renvoyer Vrai
10          sinon renvoyer sat (P, L \ {x},μ ∘ [x ← F])
```

Cette fonction réalise donc un bactracking et teste potentiellement $2^{\text{nb de variables de } P}$ contextes (dégénérés).

- appel initial : **evaluer**($P$, liste des variables de $P$, ctxt vide).

# Algorithme de Quine

- L'algorithme de Quine prend en paramètre une proposition en FNC et réalise un backtracking (comme la recherche de satisfiabilité en force brute).

# Algorithme de Quine

- L'algorithme de Quine prend en paramètre une proposition en FNC et réalise un backtracking (comme la recherche de satisfiabilité en force brute).
- En revanche, chaque substitution est l'occasion de supprimer un littéral d'une clause ou une clause de la formule.

# Algorithme de Quine : suppression de littéral ou de clause

Soit $x$ une variable et $P$ une proposition en forme normale conjonctive : $P = c_1 \wedge c_2 \wedge \cdots \wedge c_n$.
Lors de la substitution $P[x \leftarrow T]$ on effectue des simplifications :

- Si une clause contient le littéral $x$, on enlève cette clause car elle est de la forme $v_1 \vee v_2 \vee \cdots \vee \underbrace{T}_{\uparrow \text{ emplacement de } x} \vee \cdots \vee v_k$ donc vraie.

# Algorithme de Quine : suppression de littéral ou de clause

Soit $x$ une variable et $P$ une proposition en forme normale conjonctive :
$P = c_1 \wedge c_2 \wedge \cdots \wedge c_n$.
Lors de la substitution $P[x \leftarrow T]$ on effectue des simplifications :

- Si une clause contient le littéral $x$, on enlève cette clause car elle est de la forme $v_1 \vee v_2 \vee \cdots \vee \underbrace{T}_{\uparrow \text{ emplacement de } x} \vee \cdots \vee v_k$ donc vraie.
- Si une clause contient $\neg x$, on supprime ce littéral de cette clause car il est faux ($F$ est l'élément neutre de la disjonction).
  $v_1 \vee v_2 \vee \cdots \vee \underbrace{\neg T}_{\uparrow \text{ emplacement de } x} \vee \cdots \vee v_k$

# Algorithme de Quine : suppression de littéral ou de clause

$P$ est en forme normale conjonctive $P = c_1 \wedge c_2 \wedge \cdots \wedge c_n$
Lors de la substitution $P[x \leftarrow F]$ on effectue des simplifications :

- Si une clause contient $\neg x$, on enlève cette clause car elle est de la forme $v_1 \vee v_2 \vee \cdots \vee \underbrace{\neg F}_{\uparrow \text{ emplacement de } x} \vee \cdots \vee v_k$ donc vraie.

# Algorithme de Quine : suppression de littéral ou de clause

$P$ est en forme normale conjonctive $P = c_1 \wedge c_2 \wedge \cdots \wedge c_n$
Lors de la substitution $P[x \leftarrow F]$ on effectue des simplifications :

- Si une clause contient $\neg x$, on enlève cette clause car elle est de la forme $v_1 \vee v_2 \vee \cdots \vee \underbrace{\neg F}_{\uparrow \text{ emplacement de } x} \vee \cdots \vee v_k$ donc vraie.
- Si une clause contient $x$, on supprime ce littéral de cette clause car il est faux.
  $v_1 \vee v_2 \vee \cdots \vee \underbrace{F}_{\uparrow \text{ emplacement de } x} \vee \cdots \vee v_k$

# Algorithme de Quine : proposition ou clause vide

- On supprime à chaque étape soit une/des clause(s) soit un/des littéral/littéraux.

# Algorithme de Quine : proposition ou clause vide

- On supprime à chaque étape soit une/des clause(s) soit un/des littéral/littéraux.
- Si une clause devient vide (disjonction de 0 propositions), la clause est fausse donc la formule (qui est une conjonction de clause) est fausse aussi. On backtrack.

# Algorithme de Quine : proposition ou clause vide

- On supprime à chaque étape soit une/des clause(s) soit un/des littéral/littéraux.
- Si une clause devient vide (disjonction de 0 propositions), la clause est fausse donc la formule (qui est une conjonction de clause) est fausse aussi. On backtrack.
- Si la proposition devient vide (conjonction de 0 propositions), la formule est vraie. C'est gagné !