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 qui remplace x par T (True) et n'affecte aucune autre variable.

Substitution

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

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

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

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

  contexte partiel 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., μ :ctxt complet)**

Listing 1 – Algorithme de satisfiabilité

```
fonction sat(P : proposition;
             L : ensemble de variables à remplacer par T/F,
             μ : contexte partiel n'agissant pas sur L)
    sortie : un bouléen
    si L est vide renvoyer evaluer (P,μ)
    prendre x ∈ L
    si sat (P,L \ {x},μ ∘ [x ← T])
       alors renvoyer Vrai
    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.

Satisfiabilité par backtracking

- On dispose de **evaluer (*P* :prop., μ :ctxt complet)**

Listing 2 – Algorithme de satisfiabilité

```
fonction sat(P : proposition;
             L : ensemble de variables àremplacer par T/F,
             μ : contexte partiel n'agissant pas sur L)
         sortie : un bouléen
         si L est vide renvoyer evaluer (P,μ)
         prendre x ∈ L
         si sat (P,L\ {x},μ ∘ [x ← T])
            alors renvoyer Vrai
         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.

- appel initial : **evaluer(*P*, liste var de *P*)**.

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).

```
prendre une variable x restante dans la formule
tester récursivement si P[x < -T] est satisfiable
tester récursivement si P[x < -F] est satisfiable
```

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).

```
prendre une variable x restante dans la formule
tester récursivement si P[x <- T] est satisfiable
tester récursivement si P[x <- F] est satisfiable
```

- 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 $v_1 \vee v_2 \vee \cdots \vee \underbrace{T}_{\substack{\uparrow \\ \text{emplacement de } x}} \vee \cdots \vee v_k$ 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 $v_1 \vee v_2 \vee \cdots \vee \underbrace{T}_{\substack{\uparrow \\ \text{emplacement de } x}} \vee \cdots \vee v_k$ 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).
  $v_1 \vee v_2 \vee \cdots \vee \underbrace{\neg T}_{\substack{\uparrow \\ \text{emplacement de } x}} \vee \cdots \vee v_k$

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

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 !