INFO-F-302, Cours d'Informatique Fondamentale

Emmanuel Filiot
Département d'Informatique
Faculté des Sciences
Université Libre de Bruxelles

Année académique 2020-2021

38 Comment tester la satisfaisabilité d'une formule ?

38 Comment tester la satisfaisabilité d'une formule ?

- on a vu la méthode des tables de vérité : tester toutes les interprétations de propositions jusqu'à ce qu'on en trouve une qui satisfait la formule
- **problème :** quelle est la complexité d'un tel algorithme ? combien d'interprétations faut-il tester si la formule contient n propositions ?

38 Comment tester la satisfaisabilité d'une formule ?

- on a vu la méthode des tables de vérité : tester toutes les interprétations de propositions jusqu'à ce qu'on en trouve une qui satisfait la formule
- **problème :** quelle est la complexité d'un tel algorithme ? combien d'interprétations faut-il tester si la formule contient n propositions ?
- il faut essayer, dans le pire des cas (c'est à dire le cas où la formule n'est pas satisfaisable), 2^n interprétations : cet algorithme a donc une complexité exponentielle dans le nombre de propositions
- ce n'est pas raisonnable pour les applications que nous allons aborder, car nous allons générer des formules contenant plusieurs centaines de propositions.
- nous allons maintenant étudier un algorithme "plus intelligent" : la méthode des tableaux sémantiques.

2 Tableaux sémantiques

C'est un algorithme pour établir la satisfaisabilité de formules de la logique propositionnelle.
On a besoin d'une nouvelle définition :

Définition

Un littéral est une proposition x ou la négation d'une proposition $\neg x$.

3 Tableaux sémantiques : exemple

Considérons la formule $\phi = x \wedge (\neg y \vee \neg x)$
Essayons de construire systématiquement une fonction d'interprétation V telle que

$$[\![\phi]\!]v = 1$$

Par définition on a

$$[\![\phi]\!]v = 1$$

ssi

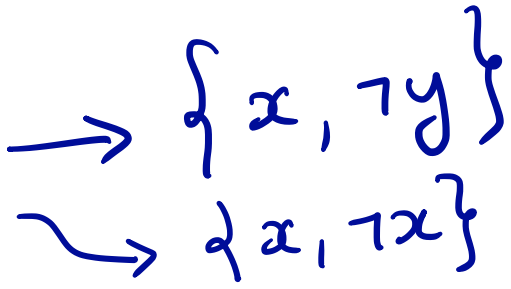
$$[\![x]\!]v = 1 \text{ et } [\![\neg y \vee \neg x]\!]v = 1$$

et

$$[\![\neg y \vee \neg x]\!]v = 1$$
$$\underline{\text{ssi}}$$
$$[\![\neg y]\!]v = 1 \ \ \underline{\text{ou}} \ \ [\![\neg x]\!]v = 1$$

et donc,

$$[\![\phi]\!]v = 1$$
$$\underline{\text{ssi}}$$
$$[\![x]\!]v = 1 \ \underline{\text{et}} \ [\![\neg y]\!]v = 1$$
$$\underline{\text{ou}} \ [\![x]\!]v = 1 \ \underline{\text{et}} \ [\![\neg x]\!]v = 1$$



Pour ϕ, on construit la fonction d'interprétation V de la façon suivante :

- $V(x) = 1$;
- $V(y) = 0$.

et on a bien que $V \models \phi$.

5 Tableaux sémantiques

- Un ensemble S de littéraux est satisfaisable ssi il ne contient pas une paire de *littéraux complémentaires* . Par exemple, $\{x, \neg y\}$ est satisfaisable alors que $\{x, \neg x\}$ ne l'est pas.
- Nous avons réduit le problème de satisfaction de ϕ à un problème de satisfaction d'ensembles de littéraux : ϕ est satisfaisable ssi $\{x, \neg y\}$ est satisfaisable ou $\{x, \neg x\}$ est satisfaisable.

6 Tableaux sémantiques

Un autre exemple : $\phi = (x \vee y) \wedge (\neg x \wedge \neg y)$.
Par définition, on a

$$[\![\phi]\!]v = 1$$
ssi
$$[\![x \vee y]\!]v = 1 \text{ et } [\![\neg x \wedge \neg y]\!]v = 1$$

et donc,

$$[\![\phi]\!]v = 1$$
ssi
$$[\![x \vee y]\!]v = 1 \text{ et } [\![\neg x]\!]v = 1 \text{ et } [\![\neg y]\!]v = 1$$

et donc,

$$[\![\phi]\!]v = 1$$
ssi
soit $[\![x]\!]v = 1$ et $[\![\neg x]\!]v = 1$ et $[\![\neg y]\!]v = 1$ → {x, ¬x, ¬y}
ou $[\![y]\!]v = 1$ et $[\![\neg x]\!]v = 1$ et $[\![\neg y]\!]v = 1$ → {y, ¬x, ¬y}

7 Tableaux sémantiques

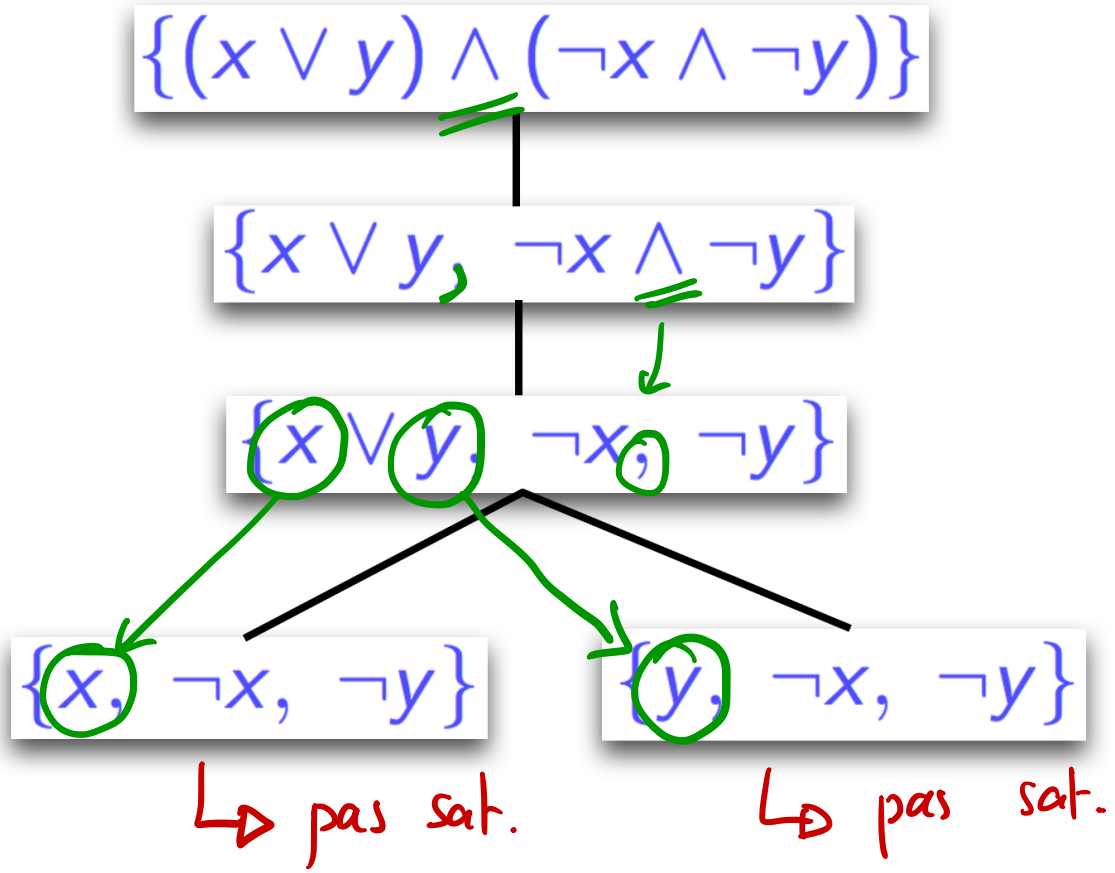
et donc,

$$\phi \text{ est satisfaisable}$$
$$\underline{\text{ssi}}$$
$$\{x, \neg x, \neg y\} \text{ est satisfaisable}$$
$$\underline{\text{ou}}\{y, \neg x, \neg y\} \text{ est satisfaisable.}$$

Vu que les deux ensembles de littéraux contiennent chacun une proposition et sa négation, la formule ϕ n'est pas satisfaisable.

8 Tableaux sémantiques

L'information présente dans les développements précédents est plus facilement représentée sous forme d'un arbre :



9 Tableaux sémantiques – Remarques

- quand on applique le "pas de simplification" à un noeud où on sélectionne une conjonction, alors on obtient un seul fils ; on appelle ces règles, des $\wedge$-règles ;
- quand on applique le "pas de simplification" à un noeud où on sélectionnne une disjonction, alors on obtient deux fils ; on appelle ces règles, des $\vee$-règles.

$\neg(\phi_1 \vee \phi_2) \equiv$
$\neg\phi_1 \wedge \neg\phi_2$

10 Règles de simplification : $\wedge$-règles

Etant donné un ensemble S contenant une formule α, on crée un ensemble fils égal $S \backslash \{\alpha\}$ auquel on ajoute α_1 et α_2.

α	α_1	α_2
$\neg\neg\phi$	ϕ	
$\phi_1 \wedge \phi_2$	ϕ_1	ϕ_2
$\neg(\phi_1 \vee \phi_2)$	$\neg\phi_1$	$\neg\phi_2$
$\neg(\phi_1 \rightarrow \phi_2)$	ϕ_1	$\neg\phi_2$
$\phi_1 \leftrightarrow \phi_2$	$\phi_1 \rightarrow \phi_2$	$\phi_2 \rightarrow \phi_1$

$\wedge$-règles .

Remarque : toutes les formules α peuvent être considérées comme équivalentes à des conjonctions. Par exemple :

$\neg(\phi_1 \vee \phi_2)$ est équivalent à $\neg\phi_1 \wedge \neg\phi_2$

11 Règles de simplification : $\vee$-règles

Etant donné un ensemble S contenant une formule β, on crée deux ensembles fils, l'un étant $(S \cup \{\beta_1\})\backslash\beta$, et l'autre $(S \cup \{\beta_2\})\backslash\beta$.

β	β_1	β_2
$\phi_1 \vee \phi_2$	ϕ_1	ϕ_2
$\neg(\phi_1 \wedge \phi_2)$	$\neg\phi_1$	$\neg\phi_2$
$\phi_1 \rightarrow \phi_2$	$\neg\phi_1$	ϕ_2
$\neg(\phi_1 \leftrightarrow \phi_2)$	$\neg(\phi_1 \rightarrow \phi_2)$	$\neg(\phi_2 \rightarrow \phi_1)$

$\vee$-*règles* .

$\neg(\phi_1 \wedge \phi_2) \equiv \neg\phi_1 \vee \neg\phi_2$

$\phi_1 \rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$

Remarque : toutes les formules β peuvent être considérées comme équivalentes à des disjonctions. Par exemple :

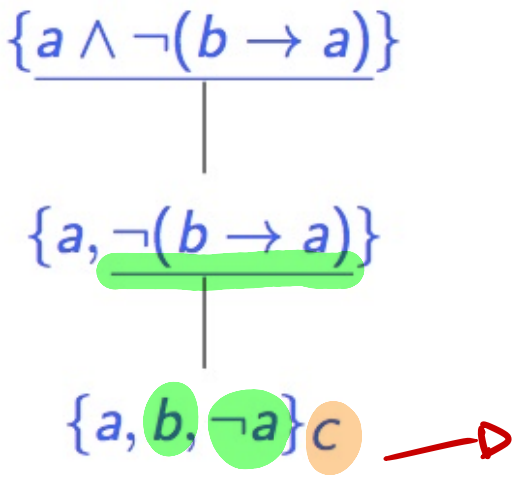
$\neg(\phi_1 \wedge \phi_2)$ est équivalent à $\neg\phi_1 \vee \neg\phi_2$

12 Algorithme

L'algorithme construit à partir d'une formule ϕ un arbre noté T_ϕ dont les noeuds sont des ensembles de formules.

- **Initialisation** : au départ, l'arbre ne contient qu'un seul noeud : l'ensemble $\{\phi\}$;
- **Tant qu**'il existe une feuille de l'arbre F qui contient une formule ψ qui est simplifiable
  - **si** ϕ est simplifiable par une $\wedge$-règle, alors on crée un fils F' à F où F' est obtenu supprimant ϕ de F et en ajoutant la formule (dans le cas d'une double négation) ou les deux formules (pour les autres cas) obtenues par la simplification, à l'ensemble F'
  - **si** ϕ est simplifiable par une $\vee$-règle, alors on crée deux fils F_1 et F_2, chacun étant obtenu en supprimant ϕ de F et en ajoutant respectivement les formules obtenues par la simplification.
- **si** toutes les feuilles de l'arbre contiennent une paire de littéraux complémentaires, **alors retourner** non-satisfaisable, **sinon retourner** satisfaisable.

13 Exemples



Toutes les feuilles (ici une seule) contiennent une paire de littéraux complémentaires (a et $\neg a$), donc la formule n'est pas satisfaisable.

$$\{((a \vee c) \wedge (b \vee c)) \rightarrow (\neg b \rightarrow ((a \wedge b) \vee c))\}$$

$\phi_1 \rightarrow \phi_2$

$\{\neg((a \vee c) \wedge (b \vee c))\}$ $\quad$ $\{\neg b \rightarrow ((a \wedge b) \vee c)\}$

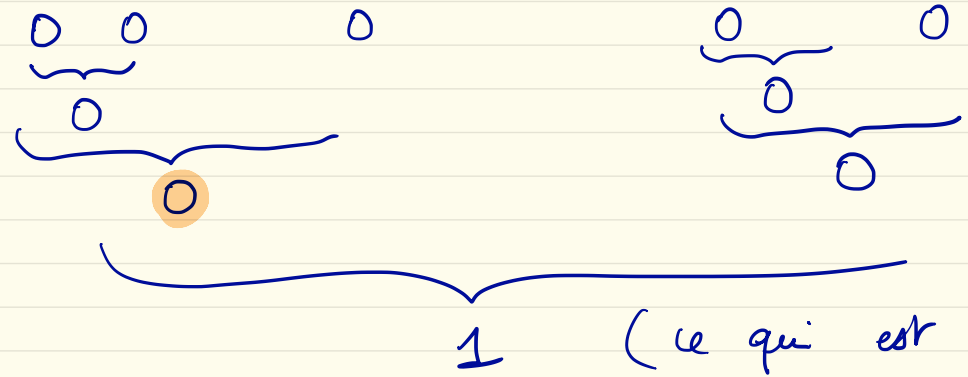
$\{\neg(a \vee c)\}$ $\quad$ $\{\neg(b \vee c)\}$ $\quad$ $\{\neg\neg b\}$ $\quad$ $\{(a \wedge b) \vee c\}$

$\{\neg a, \neg c\}$ $\quad$ $\{\neg b, \neg c\}$ $\quad$ $\{b\}$ $\quad$ $\{a \wedge b\}$ $\quad$ $\{c\}$

$\{a, b\}$

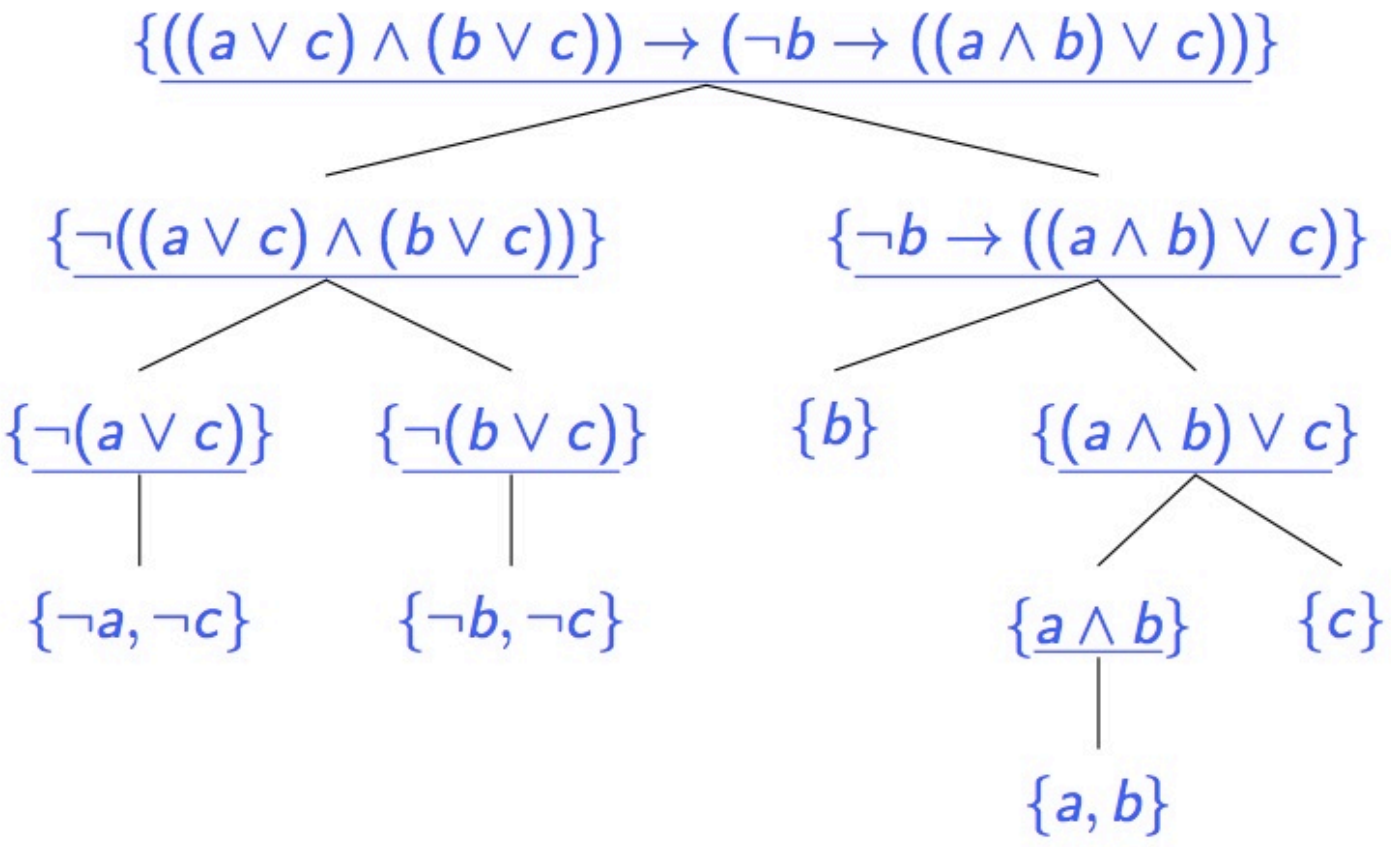
Cette ensemble est satisfaisable par $V(a)=0$ et $V(c)=0$ donc la formule de départ est satisfaisable.

$$\{((a \vee c) \wedge (b \vee c)) \rightarrow (\neg b \rightarrow ((a \wedge b) \vee c))\}$$



1 (ce qui est à gauche de l'implication est faux donc l'implication est vraie)

14 Exemples



Il existe plusieurs feuilles ne contenant pas de paire de littéraux complémentaires, la formule est donc satisfaisable.

15 Questions

Soit ϕ une formule et T_ϕ un arbre construit avec l'algorithme précédent.

1. Si toutes les feuilles de T_ϕ sont satisfaisables, est-ce que cela signifie que ϕ est valide ?

$\phi = x$ $\quad T_\phi = \{x\}$ (arbre qui contient une feuille)

Pourtant x n'est pas valide
mais son arbre ne contient pas
d'ensemble insatisfaisable.

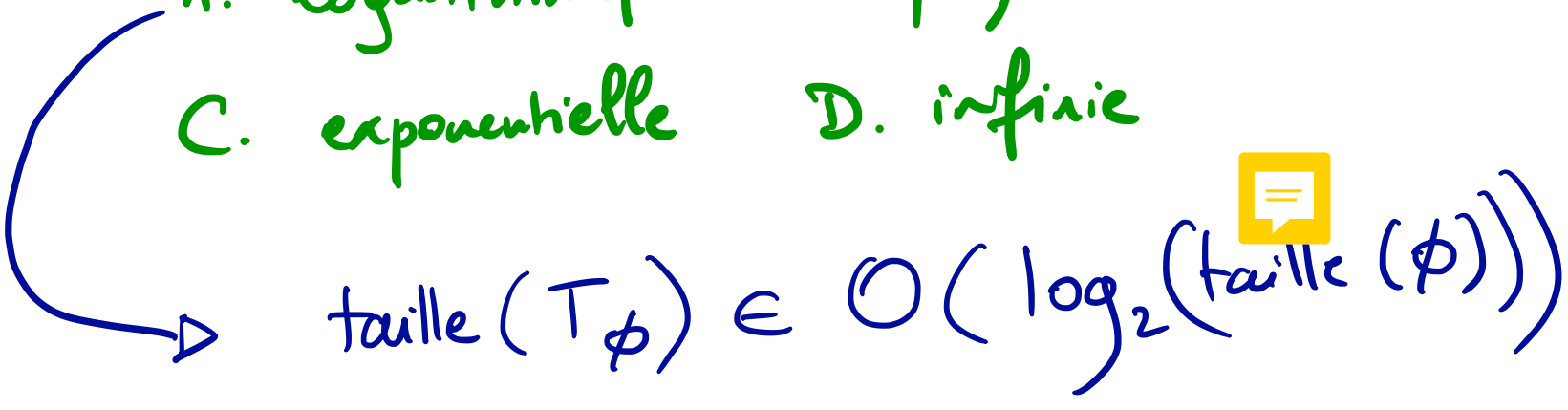
15 Questions

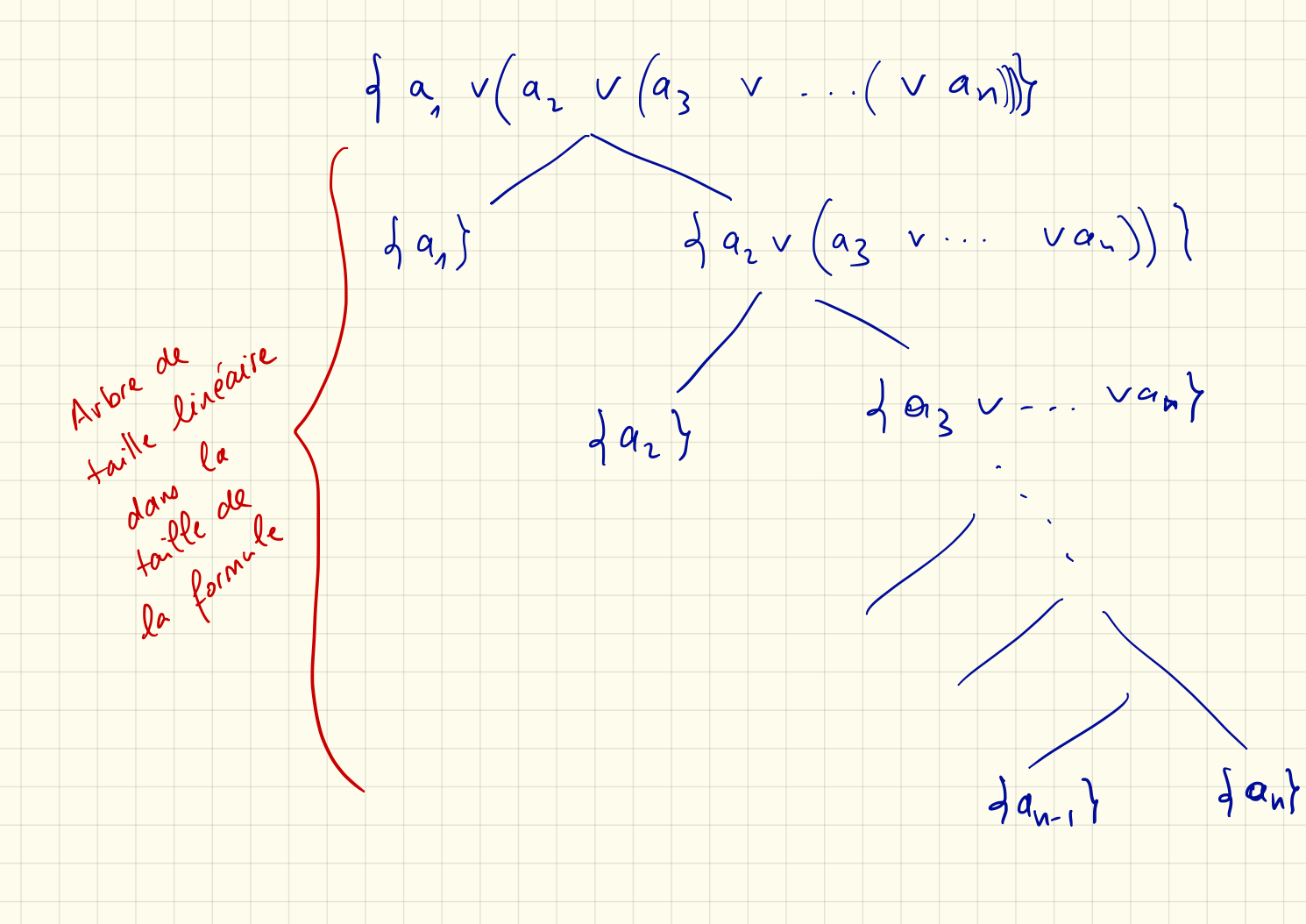
Soit ϕ une formule et T_ϕ un arbre construit avec l'algorithme précédent.

1. Si toutes les feuilles de T_ϕ sont satisfaisables, est-ce que cela signifie que ϕ est valide ? Non : prendre par exemple $\phi = x$.
2. quelle est la taille maximale de T_ϕ (nombre de noeuds) en fonction du nombre de symboles de ϕ ?

A. logarithmique B. polynomiale

C. exponentielle D. infinie

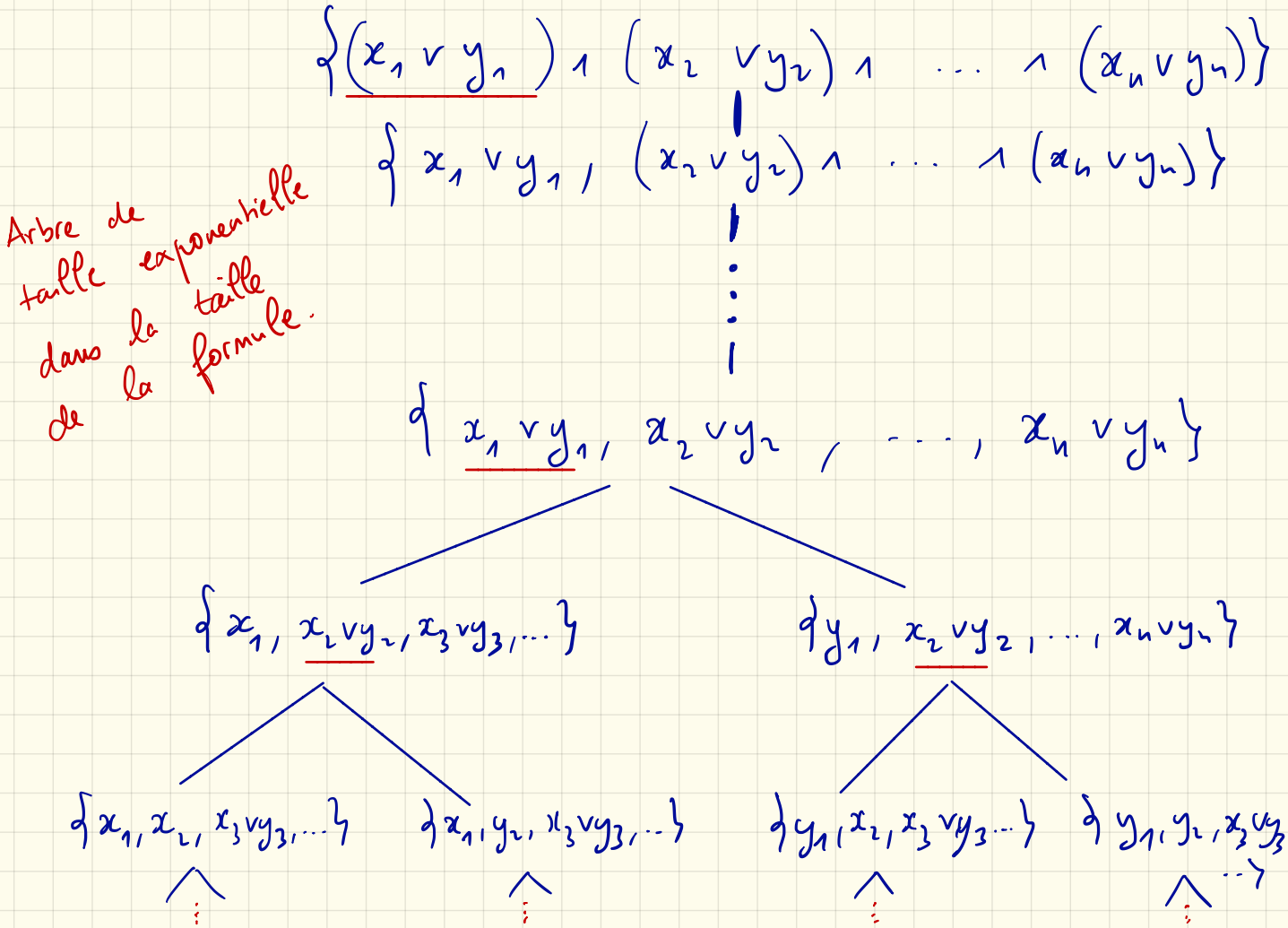



$\{a_1 \vee (a_2 \vee (a_3 \vee \dots (\vee a_n)))\}$
$\{a_1\}$
$\{a_2 \vee (a_3 \vee \dots \vee a_n)\}$
$\{a_2\}$
$\{a_3 \vee \dots \vee a_n\}$
$\{a_{n-1}\}$
$\{a_n\}$
Arbre de taille linéaire dans la taille de la formule

15 Questions

Soit ϕ une formule et T_ϕ un arbre construit avec l'algorithme précédent.

1. Si toutes les feuilles de T_ϕ sont satisfaisables, est-ce que cela signifie que ϕ est valide ? Non : prendre par exemple $\phi = x$.
2. quelle est la taille maximale de T_ϕ (nombre de noeuds) en fonction du nombre de symboles de ϕ ? T_ϕ peut avoir un nombre exponentiel de noeuds. Par exemple, prenons $n \geq 0$ et $\phi = (x_1 \vee y_1) \wedge (x_2 \vee y_2) \cdots \wedge (x_n \vee y_n)$, alors T_ϕ a au moins $2^{n+1} + n - 2$ noeuds, et exactement 2^n feuilles. L'algorithme a donc une complexité exponentielle en temps dans le pire cas.


$\{(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \dots \wedge (x_n \vee y_n)\}$
$\{x_1 \vee y_1, (x_2 \vee y_2) \wedge \dots \wedge (x_n \vee y_n)\}$
$\{x_1 \vee y_1, x_2 \vee y_2, \dots, x_n \vee y_n\}$
$\{x_1, x_2 \vee y_2, x_3 \vee y_3, \dots\}$
$\{y_1, x_2 \vee y_2, \dots, x_n \vee y_n\}$
$\{x_1, x_2, x_3 \vee y_3, \dots\}$
$\{x_1, y_2, x_3 \vee y_3, \dots\}$
$\{y_1, x_2, x_3 \vee y_3 \dots\}$
$\{y_1, y_2, x_3 \vee y_3 \dots\}$
Arbre de taille exponentielle dans la taille de la formule.

16 Tableaux Sémantiques : Résumé

- la méthode des tableaux sémantiques est un algorithme pour tester la satisfaisabilité d'une formule.
- elle est basée sur la construction d'un arbre par applications successives de règles de simplifications de formules (tableaux)
- les formules équivalentes à une disjonction (disjonction, implication, négation d'une conjonction, négation d'une équivalence) sont satisfaites si un des deux membres de la disjonction est satisfait : les deux choix sont représentés par du branchement (deux fils) dans l'arbre.
- les formules équivalentes à une conjonction (conjonction, équivalence, négation d'une implication et d'une disjonction) sont satisfaites si les deux membres de la conjonction sont satisfaits : un seul fils est créer dans l'arbre.
- les feuilles de l'arbres sont des ensembles de littéraux (donc non simplifiables), pour lesquels la satisfaisabilité est facile à tester (absence de littéraux complémentaires).
- dans le pire cas, l'arbre créé est exponentiellement plus grand que la formule
- pour tester la validité d'une formule, on peut tester la non satisfaisabilité de se négation par la méthode des tableaux sémantiques.