Problèmes de satisfaction de contraintes

Cours de Pascal Koiran
Scribes : Nicolas Tremblay, Bruno Grenet, Thomas Garnier,Angeletti Florian

5 janvier 2009

1 Définitions et premier énoncé du théorème de dichotomie

On commence par donner quelques exemples de problèmes de satisfaction de contraintes. Le cadre formel sera donné par la suite.

Exemple 1.1.

1. Sat : satisfaisabilité des formules booléennes sous forme normale conjonctive.

   Instance : une formule F donnée comme conjonction de clauses, une clause étant une disjonction de littéraux (X ou $\overline{X}$).

   Question : Est-ce qu'il existe un assignement ν qui rende F vrai ?

   On cherche une fonction $\nu : \{X_i : 1 \leqslant i \leqslant n\} \to \{O, A\}$ telle que $F(\nu(X_1), \ldots, \nu(X_n)) = 1$. Ce problème est NP-complet.
2. 3-Sat : Restriction de Sat dans laquelle chaque clause contient au plus trois littéraux (NP-complet).
3. 2-Sat : Restriction de Sat dans laquelle chaque clause contient au plus deux littéraux (polynomial).
4. On définit $1 - in - 3(x, y, z) = (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{y} \vee \overline{z})$. Le problème $1 - in - 3 -$ Sat (sur les variables $X_1, X_2, \ldots, X_n$) est la satisfaisabilité d'une formule constituée d'une conjonction de contraintes de la forme $1 - in - 3(X_{i_1}, X_{i_2}, X_{i_3})$.

Ces exemples permettent d'appréhender le cadre formel des problèmes de satisfaction de contrainte.

Définition 1.2.

- Une *contrainte* est une fonction booléenne $f : \{0,1\}^k \to \{0,1\}$.
- Un *ensemble de contraintes* $\mathscr{F} = \{f_1, \ldots, f_l\}$ est un ensemble (fini) de contraintes satisfaisables : $\forall i, \exists \vec{x}, f_i(\vec{x}) = 1$.
- Une *instance* de Sat($\mathscr{F}$) est la donnée de
  - n variables booléennes $X_1, \ldots, X_n$,
  - une liste $g_1, \ldots, g_m$ d'éléments de $\mathscr{F}$.
  - pour chaque g_i d'arité k_i, la liste $X_{j_i(1)}, \ldots, X_{j_i(k_i)}$ des variables auxquelles on applique g_i.
- La *question* associée à Sat($\mathscr{F}$) est : Est-ce qu'il existe un assignement ν tel que $\forall i, g_i(x_{j_i(1)}, \ldots, x_{j_i(k_i)}) = 1$ avec $x_i = \nu(X_i)$?

Les exemples vus auparavant, mis à part Sat, sont exprimables dans ce cadre formel. Sat n'entre pas dans ce cadre à cause de la restriction aux ensembles de contraintes finis.

Exemple 1.3.

1. $1-in-3-\textsc{Sat} = \textsc{Sat}(\mathscr{F})$ avec $\mathscr{F} = \{1-in-3\}$.
2. $2-\textsc{Sat} = \textsc{Sat}(\mathscr{F})$ avec $\mathscr{F} = \{OR_0, OR_1, OR_2\}$ où $OR_0(X,Y) = X \vee Y$, $OR_1(X,Y) = \overline{X} \vee Y$, $OR_2(X,Y) = \overline{X} \vee \overline{Y}$.
3. $3-\textsc{Sat} = \textsc{Sat}(\mathscr{F})$ avec $\mathscr{F} = \{OR_{3,0}, OR_{3,1}, OR_{3,2}, OR_{3,3}\}$ où $OR_{3,0}(X,Y,Z) = X \vee Y \vee Z$, $OR_{3,1}(X,Y,Z) = \overline{X} \vee Y \vee Z$, $OR_{3,2}(X,Y,Z) = \overline{X} \vee \overline{Y} \vee Z$, $OR_{3,0}(X,Y,Z) = \overline{X} \vee \overline{Y} \vee \overline{Z}$.

Le théorème suivant, dont on exprimera dans la suite une version complétée, donne une structure aux problèmes de satisfaction de contraintes.

Théorème 1.4 (Dichotomie, Schaefer (1978))**.** *Pour tout $\mathscr{F}$ fini,* $\textsc{Sat}(\mathscr{F})$ *est soit dans* P*, soit* NP*-complet.*

On fait ici une digression pour montrer que le théorème précédent n'est pas trivial. Pouvoir classifier de façon aussi précise les problèmes est en effet rare.

Théorème 1.5 (Ladner, 1975)**.** *Si* $\mathsf{P} \neq \mathsf{NP}$*, il existe* $A \in \mathsf{NP}$ *qui n'est ni polynomial ni* NP*-complet.*

Exemple 1.6 (Exemple de problème *intermédiaire*)**.** Le problème Facteur est un problème tel que prévu par le théorème de Ladner, à condition qu'il en existe bien.

Instance : trois entiers n, a, b.

Question : est-ce que n a un facteur premier dans l'intervalle $[a, b]$?

- Facteur $\in \mathsf{NP}$: le certificat est la donnée du facteur premier. La vérification se fait en temps polynomial grâce à l'algorithme d'Agrawal, Kayal et Saxena (2002).$f \in \mathscr{F}$
- Facteur $\notin \mathsf{P}$: dans le cas inverse, on pourrait factoriser en temps polynomial. On suppose que ceci est faux, et c'est sur cette supposition que reposent par exemple les protocoles de cryptographie actuels.
- Facteur $\in \mathsf{CoNP}$ [1] : le certificat est la décomposition de n en produit de facteurs premiers.

Donc Facteur n'est pas NP-complet. Cette conclusion suppose que $\mathsf{NP} \neq \mathsf{CoNP}$, ce qui est une hypothèse plus forte que $\mathsf{P} \neq \mathsf{NP}$, mais qui est également supposée vraie.

On peut étendre le problème de Schaefer aux problèmes suivants :
- problèmes de comptage : compter le nombre de solution,
- problèmes d'optimisation : maximiser le nombre de contraintes satisfaites,
- décision pour les formules booléennes quantifiées [2] (QBF),
- domaines non booléens (mais finis).

Il existe des théorèmes de dichotomie pour ces problèmes là.

2 Cas faciles du théorème de Schaefer

On définit certains problèmes étant des cas faciles du théorème de Schaefer, pour lesquels $\textsc{Sat}(\mathscr{F})$ est dans P. Ces problèmes sont 2-Sat, Xor-Sat et Horn-Sat.

Théorème 2.1. 2-Sat *est dans* P.

On ne donne pas la preuve de ce théorème.

1. Pour prouver qu'un problème est dans CoNP, il faut montrer qu'on peut vérifier un certificat des instances *négatives* en temps polynomial.
2. Le problème habituel consiste à résoudre une formule de la forme $\exists \vec{x}, f(\vec{x}) = 1$. Pour les formules booléennes quantifiées, le problème est de la forme $\exists \vec{x} \forall \vec{y} \exists \vec{z} \ldots, f(\vec{x}, \vec{y}, \vec{z}, \ldots) = 1$.

Xor-Sat

Définition 2.2. On définit les fonctions suivantes :
- $XOR(x, y) = x$ « ou exclusif »y
- $XOR_n(x_1, \ldots, x_n) = 1 \Leftrightarrow$ le nombre des x_i égaux à 1 est impair
- $\overline{XOR_n}(x_1, \ldots, x_n) = 1 \Leftrightarrow$ le nombre des x_i égaux à 1 est pair

Alors Xor-Sat = Sat($\mathscr{F}$) avec $\mathscr{F} = \{XOR_n; n \in \mathbb{N}\} \cup \{\overline{XOR_n}; n \in \mathbb{N}\}$

Théorème 2.3. Xor-Sat *est dans* P.

Démonstration. Cela se montre avec la méthode de Gauss dans $\mathbb{Z}/2\mathbb{Z}$, en remarquant que $XOR(x, y) \equiv (x + y)$ [2] et $XOR_n(x_1, \ldots, x_n) \equiv \sum_{i=1}^{n} x_i$ [2]. □

Horn-Sat

Définition 2.4. Une clause est dite de Horn si elle contient au plus un littéral positif. Une formule de Horn est une conjonction de clauses de Horn.

Remarques.
- Une clause qui contient un et un seul littéral positif peut s'écrire $\overline{x_1} \vee \ldots \vee \overline{x_m} \vee y$ ou, de manière équivalente, $(x_1 \wedge \ldots \wedge x_m) \Rightarrow y$. Ces clauses sont dites *implicatives*.
- Une clause qui ne contient aucun littéral positif s'écrit : $\overline{x_1} \vee \ldots \vee \overline{x_m}$ ou $(x_1 \wedge \ldots \wedge x_m) \Rightarrow$ faux.
- Les clauses de Horn sont la base de Prolog.

Théorème 2.5. Horn-Sat *est dans* P.

Démonstration. Soit φ une formule de Horn. On note φ_I la formule formée des clauses implicatives de φ. On va construire un assignement qui satisfait φ_I (il en existe toujours au moins un : $\forall X\ \nu(X) = 1$, mais on va en chercher un qui ait un nombre minimal de 1). L'algorithme de construction est le suivant :

Initialisation : On met toutes les variables à faux.

Itération : Tant qu'il existe une implication insatisfaite, par exemple si $(x_1 \wedge \ldots \wedge x_m) \Rightarrow y$ est non satisfaite, on la satisfait en posant $y :=$ vrai.

Propriétés :
- La boucle est exécutée au plus n fois, n étant le nombre de variables.
- L'algorithme produit toujours un assignement qui satisfait φ_I.
- L'assignement qui est produit est minimal pour l'inclusion. Si on identifie ν à l'ensemble des variables T qui prennent la valeur vraie, on a alors : pour tout assignement T' satisfaisant φ_I, $T \subseteq T'$.

  Démonstration. Soit T' un assignement qui satisfait φ_I. On montre la minimalité de T à chaque étape de l'algorithme :

  Initialisation : $T = \emptyset \subseteq T'$.

  Itération : $(x_1 \wedge \ldots \wedge x_m) \Rightarrow y$ est non satisfaite, ce qui implique que $\forall i\ x_i \in T$. Or $T \subseteq T'$ par hypothèse, donc $y \in T'$ car T' satisfait φ_I. Ainsi, $T \cup \{y\} \subseteq T'$.

  □

Lemme 2.6. *Soit T l'assignement produit par cet algorithme. φ est satisfaisable si et seulement si T satisfait φ.*

Démonstration. ($\Leftarrow$) trivial.

($\Rightarrow$) Montrons la contraposée : si T ne satisfait pas φ alors φ n'est pas satisfaisable.
Si T ne satisfait pas φ, il existe une clause qui n'est pas satisfaite de la forme $\overline{x_1} \vee \ldots \vee \overline{x_m}$ *i.e.* $\{x_1, \ldots, x_m\} \subseteq T$. Par minimalité de T, pour tout T' qui satisfait φ_I, $T \subseteq T'$. Donc, T' ne satisfait pas non plus la clause $\overline{x_1} \vee \ldots \vee \overline{x_m}$. □

Récapitulons. Soit un problème Horn-Sat. On construit l'assignement T en temps polynomial. Il ne nous reste ensuite plus qu'à tester T pour résoudre la satisfaisabilité du problème, ce qui se fait également en temps polynomial. □

Voici maintenant la version complète du théorème de Schaefer.

Théorème de Shaefer

Théorème 2.7. Sat($\mathscr{F}$) *est dans* P *dans les cas suivants :*

1. $f(0, \ldots, 0) = 1$ *pour tout* $f \in \mathscr{F}$. *On dit que* $\mathscr{F}$ *est 0-valide.*
2. $f(1, \ldots, 1) = 1$ *pour tout* $f \in \mathscr{F}$. *On dit que* $\mathscr{F}$ *est 1-valide.*
3. $\mathscr{F}$ *est affine : chaque* $f \in \mathscr{F}$ *est la conjonction de fonctions de* $\{XOR_n; n \in \mathbb{N}\} \cup \{\overline{XOR_n}; n \in \mathbb{N}\}$.
4. *Chaque* $f \in \mathscr{F}$ *est la conjonction de clauses de Horn.*
5. *Chaque* $f \in \mathscr{F}$ *est la conjonction de clauses de anti-Horn (Une clause de anti-Horn est une clause qui contient au plus un littéral négatif).*
6. *Chaque* $f \in \mathscr{F}$ *est la conjonction d'au plus deux littéraux.*

Sinon, Sat($\mathscr{F}$) *est* NP*-complet.*

Démonstration. Dans les cas 1 et 2, l'algorithme est évident. Les cas 3, 4 et 6 ont déjà été traités.

Cas n°5 : Notons $F'(X_1, \ldots, X_n) = F(\overline{X_1}, \ldots, \overline{X_n})$. Résoudre Horn-Sat sur F' est équivalent à résoudre AntiHorn-Sat sur F. En effet, F' est satisfaisable si et seulement si F est satisfaisable.

idée de la preuve : Lorsque l'on sort des 6 cas ainsi défini on va proceder par éviction
Proposition : Sat($\{1 - in - 3\}$) est NP-complet.

Sat($\mathscr{F}$) est dans P dans les cas suivants :

1. il existe $f \in \mathscr{F}$ qui n'est pas 0-valide.
2. il existe $f \in \mathscr{F}$ qui n'est pas 1-valide.
3. il existe $f \in \mathscr{F}$ qui n'est pas la conjonction de fonctions de $\{XOR_n; n \in \mathbb{N}\} \cup \{\overline{XOR_n}; n \in \mathbb{N}\}$.
4. il existe $f \in \mathscr{F}$ qui n'est pas la conjonction de clauses de Horn.
5. il existe $f \in \mathscr{F}$ qui n'est pas la conjonction de clauses de anti-Horn (Une clause de anti-Horn est une clause qui contient au plus un littéral négatif).
6. il existe $f \in \mathscr{F}$ qui n'est pas la conjonction d'au plus deux littéraux.

On va utiliser ces 6 fonctions pour implémenter la fonction $\{1 - in - 3\}$

Définition 2.8. *implémentation* soit $\mathscr{F}$ un ensemble de contraintes,soit $\vec{x} = \{x_1, \ldots, x_p\}$ et $\vec{y} = \{y_1, \ldots, y_p\}$ deux ensembles de variables.

Une application de contrainte de $\mathscr{F}$ aux variables de $\vec{x}$ et de $\vec{y}$ implémente une fonction booléenne $f(\vec{x})$ si :

1. pour tout $\vec{x}$ tel que $f(\vec{x})$ est vrai, il existe $\vec{y}$ tel que $\vec{x}$ et $\vec{y}$ satisfasses toutes les contraintes de l'application de contraintes

2. pour tout $\vec{x}$ tel que $f(\vec{x})$ est faux, il n'existe pas $\vec{y}$ tel que $\vec{x}$ et $\vec{y}$ satisfasses toutes les contraintes de l'application de contraintes

Exemple 2.9. La famille $\{1-in-3\}$ implémente la fonction $XOR(x_1, x_2)$ grâce a une certaine application de contraintes : $\{1-in-3\{x_1, x_2, y_f\}, 1-in-3\{y_t, y_f, y_f\}\}$

si $XOR(x_1, x_2)$ est vrai : il suffit de poser $y_f = faux$, $y_t = vrai$ pour satisfaire l'application des contraintes.

De même si $XOR(x_1, x_2)$ est faux, alors $1-in-3\{x_1, x_2, y_f\}$ ou $1-in-3\{y_t, y_f, y_f\}$ est faux.

En d'autres termes, on peut remplacer XOR quelles que soit les valeurs de y_t et y_f par une certaine instance de $1-in-3$.

Lemme 2.10. *Si* $\text{SAT}(\mathscr{F})$ *est* $\mathsf{NP}-$*dur et si toute contrainte de* $\mathscr{F}$ *peut être implémenté par* $\mathscr{F}'$ *alors* $\text{SAT}(\mathscr{F}')$ *est* NP*-dur également.*

Remarque (Problème NP-dur). Problème qui s'il était résolu en temps polynomial permettrait la résolution en temps polynomial d'un problème NP-complet.

La preuve intégrale n'est pas faite ici car elle implique de traiter beaucoup de sous cas.

□

3 Procédure de résolution

But : démontrer qu'un ensemble de clauses n'est pas satisfaisable.Pour faire cela, on va rechercher une contradiction en exploitant une règle simple :

RÈGLE DE RÉSOLUTION

$$\frac{C \bigvee x \quad D \bigvee \overline{x}}{C \bigvee D} = \frac{hypotheses}{conclusions} \tag{1}$$

se lit "si C ou x est vrai et si D ou $\overline{x}$ est vrai alors $C \bigvee D$ est vrai". soit $S = clauses$ et si S contient les clauses $C \bigvee x$ et $D \bigvee \overline{x}$ on y ajoute la clause $C \bigvee D$.

Pour résoudre l'instance de SAT, on cherche alors a déduire la clause vide en utilisant la règle de résolution. $\frac{x \quad \overline{x}}{\square}$.

Théorème 3.1 (Davis putman, 1960)**.** *on peut dériver* □ *si et seulement si S est non satisfaisable.*

Exemple 3.2.

Les formules de Horn non satisfaisables ont des réfutations par résolution de taille polynomiale.

Basé sur l'algorithme pour les clauses de Horn. On peut mimer les étapes de l'algorithme précédent en utilisant la règle de résolution.

Première étape :Implications : On avait auparavant : Si $x_1 \dots x_m$ et $x_1 \bigwedge x_2 \dots \bigwedge x_m \Rightarrow y$ alors déduire y.

Or $(x_1 \bigwedge x_2 \dots \bigwedge x_m \Rightarrow y) \Leftrightarrow \overline{x_1} \bigvee \overline{x_2} \dots \bigvee \overline{x_m} \bigvee y$ est vrai

On peut donc utiliser la règle de résolution comme suit :

$$
\begin{array}{c}
\overline{x_1} \bigvee \overline{x_2} \ldots \bigvee \overline{x_m} \bigvee y \qquad x_1 \\
\hline
\overline{x_2} \bigvee \overline{x_3} \ldots \bigvee \overline{x_m} \bigvee y \qquad x_2 \\
\hline
\overline{x_3} \bigvee \overline{x_4} \ldots \bigvee \overline{x_m} \bigvee y \\
\vdots \qquad x_{m-1} \\
\hline
x_m \bigvee y \qquad x_m \\
\hline
y
\end{array}
$$

Deuxième étape : Déduction : Une fois toutes les clauses de type implication éliminées, on essaye de résoudre les possibles contradictions de la forme $x_1 \ldots x_m$ vrai et $\overline{x_1} \bigvee \ldots \bigvee \overline{x_m}$ vrai

En effet, par la règle de résolution, on a :

$$
\begin{array}{c}
\overline{x_1} \bigvee \overline{x_2} \ldots \bigvee \overline{x_m} \qquad x_1 \\
\hline
\overline{x_2} \bigvee \overline{x_3} \ldots \bigvee \overline{x_m} \qquad x_2 \\
\hline
\overline{x_3} \bigvee \overline{x_4} \ldots \bigvee \overline{x_m} \\
\vdots \qquad x_{m-1} \\
\hline
x_m \qquad x_m \\
\hline
\square
\end{array}
$$

On arrive donc bien à dériver $\square$ en temps polynomial en présence d'une contradiction.

$\square$

Exemple 3.3. Formule sans résolution courte

$$F_n = \bigwedge_{i=1}^{n+1} \bigvee_{j=1}^{n} p_{i,j} \wedge \bigwedge_{1 \leqslant i < i' \leqslant n+1} \bigwedge_{j=1}^{n} (\overline{p}_{i,j} \vee \overline{p}_{i',j}) \tag{2}$$

F_n est non satisfaisable : Il s'agit de la formule du principe des tiroirs avec $p_{i,j}$ présence de l'objet i en j.

Théorème 3.4. *Toute réfutation par résolution de F_n contient au moins $2^{\Omega(n)}$ clauses.*

4 Graphe et SAT

Afin d'essayer de résoudre le problème SAT, on s'intéresse au graphe des relations SAT afin d'essayer d'exploiter la structure de ce graphe.

Si ce graphe est un arbre ou s'il possède une largeur arborescente bornée, on montrera qu'il est possible de résoudre le problème de satisfaisabilité en temps polynomial.

4.1 Définition des graphes

- Σ : ensemble de clause
- $I(\Sigma)$: graphe d'incidence
- $P(\Sigma)$: graphe primal

Soit V l'ensemble des variables de Σ.

Définition 4.1. On appelle graphe primal, le graphe défini par :
- $P(\Sigma) = (V, E)$.
- $(x, y) \in E \Leftrightarrow$ il existe une clause de Σ contenant x et y.

Le graphe d'incidence est plus riche :

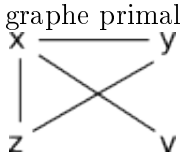
Définition 4.2. On appelle graphe d'incidence, le graphe défini par :
- Les sommets de $I(\Sigma)$ sont $\Sigma \cup V$.
- Il y a une arête entre $v \in V$ et $C \in \Sigma$ si et seulement si v apparaît dans C.

$I(\Sigma)$ est donc un graphe biparti.

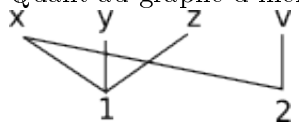
Exemple 4.3. Soit la formule :

$$(x \vee y \vee z) \wedge (\overline{x} \vee v)$$

Le graphe primal se présente sous la forme suivante :



Quant au graphe d'incidence, il peut s'écrire comme :



Théorème 4.4. *SAT se résoud en temps polynomial pour les instances de Σ pour lesquelles I(Σ) est un arbre.*

Démonstration. Idée de l'algorithme.

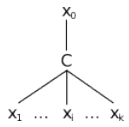
On choisit une racine de l'arbre, on parcourt ensuite l'arbre des feuilles à la racine. En chaque noeud, on calcule une information du type "x=0", "x=1" ou $\emptyset$ et on propage ce message vers le père du noeud.

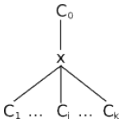
Signification de $x = \varepsilon \in \mathbb{Z}_2$: Soit γ le noeud courant, Γ l'ensemble des clauses apparaissant dans γ ou en dessous de γ.

"$x = \varepsilon$" signifie :

1. Il existe un assignement tel que $x = \varepsilon$ et toutes les clauses de Γ sont satisfaites.
2. Il n'existe pas d' assignements tel que $x \neq \varepsilon$ et toutes les clauses de Γ sont satisfaites.

Description de l'algorithme :
- Pour une feuille $x \in V$, on transmet $\emptyset$.
- Pour une feuille $C \in \Sigma$:

si $C = \overline{x}$ ou $C = x$ on transmet "x=0" ou "x=1" respectivement.
- Pour un noeud interne $C \in \Sigma$: 

avec $C = l_0 \wedge l_1 \wedge ... \wedge l_k$ où $l_i \in \{x_i, \overline{x}_i\}$
  - Si on reçoit le message pour tout $i \in \{1, .., k\}$, $l_i = 0$" des k-fils, on transmet "$l_0 = 1$".
  - Sinon, on transmet $\emptyset$.
- Pour un noeud interne $x \in V$: 

  - Si on reçoit deux messages contradictoires, on s'arrête : l'instance n'est pas satisfaisable.
  - Si on reçoit au moins un message "$x = \varepsilon$", on propage le message.
  - Sinon, on transmet $\emptyset$.
- A la racine :
  - Si la racine est une variable et que l'on ne reçoit aucun message contradictoire, Σ est satisfaisable.
  - Si la racine est une clause $C = l_0 \wedge l_1 \wedge ... \wedge l_k$, Σ est satifaisable si on ne reçoit pas pour tout $i \in \{0, .., k\}$, "$l_i = 0$".

Preuve

Soit A un arbre d'instance SAT. On considère la propriété P suivante :

- A est satisfaisable si et seulement s'il existe un ensemble de messages (défini au sens précédent) E des fils de A tel que
  - Si l'étiquette de A est une variable x, alors $|\{m \in E, m = "x = 0"\}| + |\{m \in E, m = "x = 1"\}| < 2$.
  - Si l'étiquette de A est une clause $C = l_0 \dots l_n$ alors $\sum_{i=0}^{n} |\{m \in E, m = "l_i = 1"\}| > 0$

Initialisation On appelle ici l le seul littéral apparaissant dans un arbre de taille 1 et C l'unique clause.

1. Feuille avec clause : le message "l=1" est valide car $\Gamma = C$ et C a pour unique variable l. Par conséquent, la feuille vérifit la condition P.
2. Feuille avec variable : le message "$\emptyset$" est valide pour la même raison. P est bien vérifiée.

Induction On suppose la propriété P vraie pour les sous-arbres de l' arbre A.

1. Si un sous-arbre n'est pas satisfaisable, il n'existe pas de message valide et A n'est pas satisfaisable.
2. Si tous les sous-arbres sont satisfaisables, la description de l'algorithme donne le seul message possible. On peut facilement vérifier que le message est valide :
   - S'il s'agit d'un noeud interne satisfaisable de type variable d'étiquette x, et si on reçoit "$x = \varepsilon$", les clauses de Γ ne peuvent satisfaites que si $x = \varepsilon$. Donc "$x = \varepsilon$" est un message valide. De même, si l'on reçoit "$\emptyset$" .
   - Pour un noeud de type clause $C = l_0 \dots l_m$, si l'on ne recoit pas "$l_1 = 0$" $\dots$ "l_m" , le message "$\emptyset$" est le seul valide. Dans le cas contraire, seul le message "$l_0 = 1$" est valide.

   Il existe donc un ensemble de messages valides E pour les fils de A.
3. Si A est de type variable avec x comme étiquette :
   - Si les messages reçus sont non-contradictoires, il existe alors une assignation de x telle qu' il existe une assignation des variables propres à chaque sous-arbres qui satisfasse chacune des clauses de l'instance.
   - Réciproquement, s'il existe deux clauses avec un message contradictoire, il est impossible de satisfaire simultanément les deux clauses. L'instance n'est donc pas satisfaisable.
4. Si A est de type clause avec $C = l_0 \dots l_m$ comme étiquette :
   - Si les messages reçus n'imposent pas que C soit non satisfaite, alors l'instance est satisfaisable. En effet, il est possible de satisfaire chaque sous-arbre avec une assignation de variable satisfaisant C.
   - Si les messages sont de la forme "$l_0 = 0$" $\dots l_m$, les clauses du sous-arbres ne sont pas satisfaisables en même temps que C. L'instance n'est donc pas satisfaisable.

$\square$