IA02 : Résolution de problèmes et Programmation Logique

Logique propositionnelle : de la théorie à la résolution de problèmes

Sylvain Lagrue

sylvain.lagrue@hds.utc.fr

À propos...

Information	Valeur
Auteur	Sylvain Lagrue (sylvain.lagrue@utc.fr)
Licence	Creative Common CC BY-SA 3.0
Version document	1.2.3

Sources/bibliographie :

- **Artificial Intelligence: A Modern Approach** : Stuart Russell and Peter Norvig, I.S.B.N 0136042597, 2009
- **Intelligence Artificielle et Informatique Théorique (2^{e} édition)** : Jean-Marc Alliot, Pascal Brisset, Frederick Garcia, Thomas Schiex, I.S.B.N. 2854285786, 2002

Des coquilles ?

sylvain.lagrue@utc.fr ou sur le forum du cours moodle

I. Introduction

But de la logique

Formaliser *mathématiquement* le raisonnement humain pour :

Période classique (Aristote, Platon et les péripatéticiens…)

- Analyser des raisonnements et de l'argumentation (dialectique vs. rhétorique)
- 2 types de raisonnement fallacieux : le paralogisme et le sophisme
- Objectif : la recherche de la Vérité

Période moderne

- Donner un sens aux Mathématiques
  - Établir leurs non-contradictions (2^{e} problème de Hilbert)
  - Axiomatiser leurs diverses branches
- Mécaniser le raisonnement
- Formaliser certains concepts pour l'informatique théorique (décidabilité, finitude, complexité, etc.) et l'IA

Les trois formes de raisonnement[1]

> *Les mêmes causes produisent les mêmes effets.* (Paul Valéry)

$$A \to B$$

- **Déduction** : à partir de la cause et de la règle, trouver les conséquences
- **Abduction** : à partir de la règle et des conséquences, trouver les causes
- **Induction** : à partir des causes et des conséquences, trouver la règle

Seule la déduction est **valide** : si les causes et les règles générales sont justes, les conséquences sont certaines

I. Introduction

Quelques exemples…

Les Ferrari sont des voitures rouges.

$$F \rightarrow R$$

Les voitures qui ne sont pas rouges ne sont pas des Ferrari ?

$$\neg R \rightarrow \neg F$$

Est-ce que toutes les voitures rouges sont des Ferrari ?

$$R \rightarrow F$$

Les voitures qui ne sont pas des Ferrari ne sont pas des voitures rouges ?

$$\neg F \rightarrow \neg R$$

Trouver un argument fallacieux…

$$ISF \rightarrow EF$$

Donc, pour ne plus avoir d'EF, il suffit de supprimer l'ISF…

$$\neg ISF \rightarrow \neg EF$$

L'ISF provoque de l'EF. Les PF provoquent de l'EF. **Donc** l'ISF est la cause des PF. Il faut supprimer l'ISF !

$$ISF \rightarrow EF$$
$$PF \rightarrow EF$$
$$ISF \rightarrow PF$$

I. Introduction

Applications

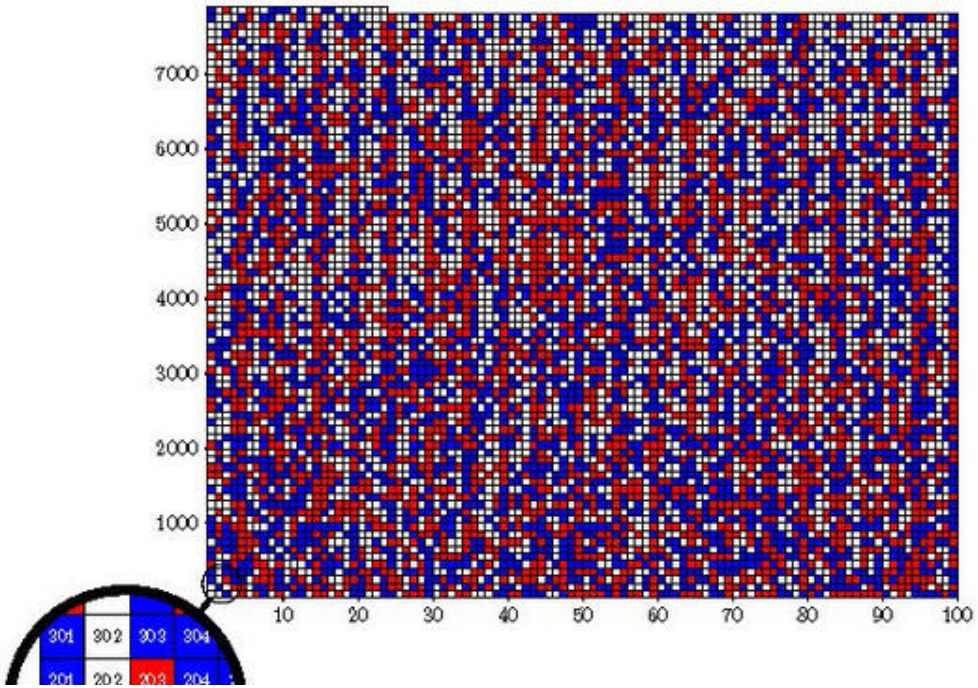
- Conception et vérification de circuits
- Preuve de programmes
- Langage de programmation
- Base de données (déductive)
- Web sémantique
- Diagnostic/panne
- Aide à la décision
- Robotique
- Analyse de documents/traitement du langage naturel
- Démonstration automatique
- etc.

I. Introduction

Exemple de preuve : bicoloration des triplets de Pythagore

> *Est-il possible de colorier chaque entier positif en bleu ou en rouge de telle manière qu'aucun triplet d'entiers a, b et c qui satisfait la fameuse équation de Pythagore a² + b² = c² ne soient pas tous de la même couleur ? Par exemple, pour le triplet 3, 4 et 5, si 3 et 5 sont coloriés en bleu, alors 4 doit être rouge.*

- Problème ouvert depuis les années 1980 (possible jusqu'à 7824)
- Résolu informatiquement en 2016 https://lejournal.cnrs.fr/articles/la-plus-grosse-preuve-de-lhistoire-des-mathematiques
- **20 To** de preuve…



I. Introduction

La logique propositionnelle

- Fragment le plus simple de la logique mathématique
- Issue des travaux de Georges Boole (1815-1864) et d'Auguste de Morgan (1806-1871)
- Liens évidents avec l'électronique, la téléphonie et l'informatique…

II. Le langage

- $V_S = \{a, b, \ldots, p, q, \ldots\}$ est un ensemble fini de variables propositionnelles
- $V_C = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \top, \bot\}$ est un ensemble de connecteurs (resp. d'arité 1, 2, 2, 2, 2, 0, 0)

Remarque : les connecteurs $\neg$ et $\vee$ forment un *système complet* (tous les autres peuvent être définis à partir de ceux-ci).

Définition : formules propositionnelles (bien formées)

1. Tout élément de V_S est une formule ;
2. Si F est une formule, alors $(\neg F)$ est une formule ;
3. Si F et G sont des formules alors $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$ et $(F \leftrightarrow G)$ sont des formules ;
4. $\top$ et $\bot$ sont des formules ;
5. Toute formule s'obtient en appliquant un nombre fini de ces règles.

On notera F_{V_S} l'ensemble des formules bien formées basées sur V_S.

II. Le langage

Priorité des opérateurs

Pour limiter les parenthèses, on peut utiliser les règles de priorité suivantes :

$$\neg > \wedge > \vee > \rightarrow , \leftrightarrow$$

Exemples :

- $\neg a \vee b \rightarrow c$ est équivalent à $(((\neg a) \vee b) \rightarrow c)$
- $\neg a \leftrightarrow b \rightarrow c$ n'est pas une formule bien formée (pas de priorité droite/gauche)

Littéral

- C'est une variable propositionnelle ou sa négation
- p et $\neg p$ sont 2 littéraux
- si $|V_S| = n$, alors il y a $2n$ littéraux

Représentation sous forme de graphes

On peut représenter toute formule sous forme d'arbre (ordonné) :

- chaque feuille de l'arbre correspond à une variable propositionnelle ;
- les autres nœuds correspondent à des connecteurs.

Exemple :

- $\neg a \lor b \to c$

On peut représenter une formule sous forme de DAG (graphe dirigé acyclique) pour représenter une formule de façon plus concise/compacte…

Exemple :

- $a \lor b \to c \land (a \lor b)$

III. Sémantique

Objectif : donner des valeurs de vérité aux formules

Pour celà, on va considérer deux valeurs (principe du **tiers exclu**) :

- {vrai, faux}
- {0, 1}
- {true, false}
- {T, F}
- $\{\top, \bot\}$
- {vert, rouge}
- **{V, F}**
- …

Interprétation

Définition : une interprétation ω est une application de V_S dans $\{V, F\}$ qui associe à chaque proposition la valeur V ou F

On notera Ω l'ensemble des interprétations possibles définies sur le langage.

(si $n = |V_S|$, on a $|\Omega| = 2^n$)

Exemple :

- $V_S = \{a, b, c\}$
- $\omega_0(a) = F$
- $\omega_0(b) = F$
- $\omega_0(c) = F$

Valuation

Définition : Soit φ une formule bien formée et $\omega \in \Omega$, la valuation de φ pour ω (notée $Val(\varphi, \omega)$) est telle que :

- si φ est une variable propositionnelle, alors $Val(\varphi, \omega) = \omega(\varphi)$;
- $Val(\top, \omega) = V$ et $Val(\bot, \omega) = F$;
- si φ est de la forme $\neg A$ (resp. $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$), alors appliquer récursivement la table de vérité suivante.

A	B	¬A	A∧B	A∨B	A→B	A↔B
F	F	V	F	F	V	V
F	V	V	F	V	V	F
V	F	F	F	V	F	F
V	V	F	V	V	V	V

Remarque : on est sûr que la valuation se termine car à chaque étape un connecteur est résolu.

III. Sémantique

Exemple

- $\omega = \{a, b, \neg c\}$
- $\varphi = \neg(a \lor (b \rightarrow \neg c))$
- $Val(\varphi, \omega) = F$

D'autres définitions...

- ω **satisfait** φ, noté $\omega \vDash \varphi$ ssi $V(\varphi, \omega) = V$. On dit alors que ω est un **modèle** de φ.
- L'ensemble des modèles de φ est noté $Mod(\varphi)$, i.e. :

$$Mod(\varphi) = \{\omega \in \Omega : \omega \vDash \varphi\}$$

- ω **falsifie** φ, noté $\omega \nvDash \varphi$ ssi $V(\varphi, \omega) = F$. On dit alors que ω est un **contre-modèle** de φ.

III. Sémantique

Une formule propositionnelle φ est dite :

- **valide** (noté $\models \varphi$) ssi pour toute interprétation $\omega \in \Omega$ on a $\omega \models \varphi$. Dans ce cas φ est également appelé **tautologie** ;
- **contradictoire** ssi pour toute interprétation $\omega \in \Omega$ on a $\omega \not\models \varphi$;
- **satisfiable** ssi elle n'est pas contradictoire ;
- **contingente** ssi il existe $\omega \in \Omega$ tel que $\omega \models \varphi$ et il existe $\omega' \in \Omega$ tel que $\omega' \not\models \varphi$.

III. Sémantique

Calculer la validité d'une formule

3 méthodes :

1. en passant par des tables de vérité
2. par arbre sémantique/algorithme de Quine
3. par l'absurde

Exemples :

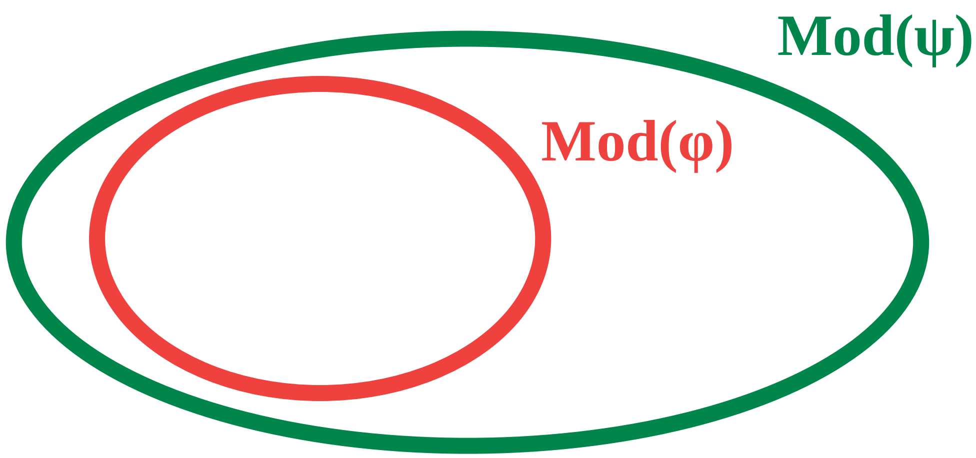
- $\varphi_1 = \neg(a \lor b) \leftrightarrow \neg a \land \neg b$ (règle de Morgan)
- $\varphi_2 = \neg q \land (p \to r) \to (\neg q \lor r)$

III. Sémantique

Conséquence logique

Définition : une formule ψ est dite **conséquence logique** de φ (noté $\varphi \models \psi$) ssi quelque soit $\omega \in \Omega$, $\omega \models \varphi$ implique $\omega \models \psi$

En d'autres termes : $Mod(\varphi) \subseteq Mod(\psi)$



III. Sémantique

Par extension : $\varphi_1, \ldots, \varphi_n \vDash \psi$ ssi pour tout $\omega \in \Omega$ tel que quel que soit φ_i, $\omega \vDash \varphi_i$, on a $\omega \vDash \psi$

Exemples :

- $a \vDash a \vee b$
- $a, a \rightarrow b \vDash b$
- $\bot \vDash a \rightarrow b \vee c$

III. Sémantique

Remarques :

- Équivalence logique $\varphi_1 \equiv \varphi_2$ ssi $\varphi_1 \models \varphi_2$ et $\varphi_2 \models \varphi_1$
- $\models \varphi$ est une écriture raccourcie de $\top \models \varphi$
- On peut tout déduire de la contradiction…

> *Principe d'explosion : ex falso quodlibet.*

⇒ Avant de chercher à déduire quoi que se soit d'un ensemble de formules, il faut toujours vérifier préalablement leur cohérence (c.a.d prouver l'existence d'au moins un modèle) !

III. Sémantique

Théorème et corollaires…

Théorème de la déduction :

$$\varphi_1, \ldots, \varphi_n \vDash \psi \text{ ssi } \varphi_1, \ldots, \varphi_{n-1} \vDash \varphi_n \rightarrow \psi$$

Corollaire 1 :

$$\varphi \vDash \psi \text{ ssi } \vDash \varphi \rightarrow \psi$$

L'implication matérielle et la déduction logique coïncident !

Corollaire 2 :

$$\varphi_1, \ldots, \varphi_n \vDash \psi \ \text{ssi} \ \varphi_1 \wedge \ldots \wedge \varphi_n \vDash \psi$$

En particulier si les φ_i sont des littéraux...

Corollaire 3 :

$$\varphi_1, \ldots, \varphi_n \vDash \psi \ \text{ssi} \ \varphi_1, \ldots, \varphi_n, \neg\psi \vDash \bot$$

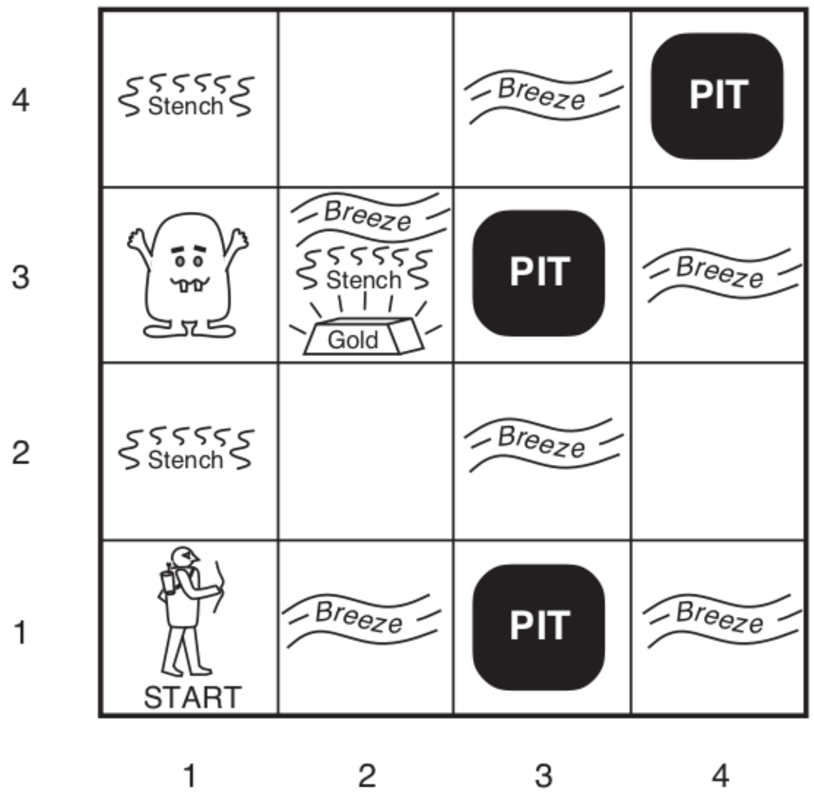
C'est le raisonnement par l'absurde : la conséquence logique peut se ramener à un simple test de satisfiabilité !

Complexité...

- Tester si une formule est satisfiable est NP-complet.
- Tester si une formule est une conséquence logique d'une autre est CoNP-complet.

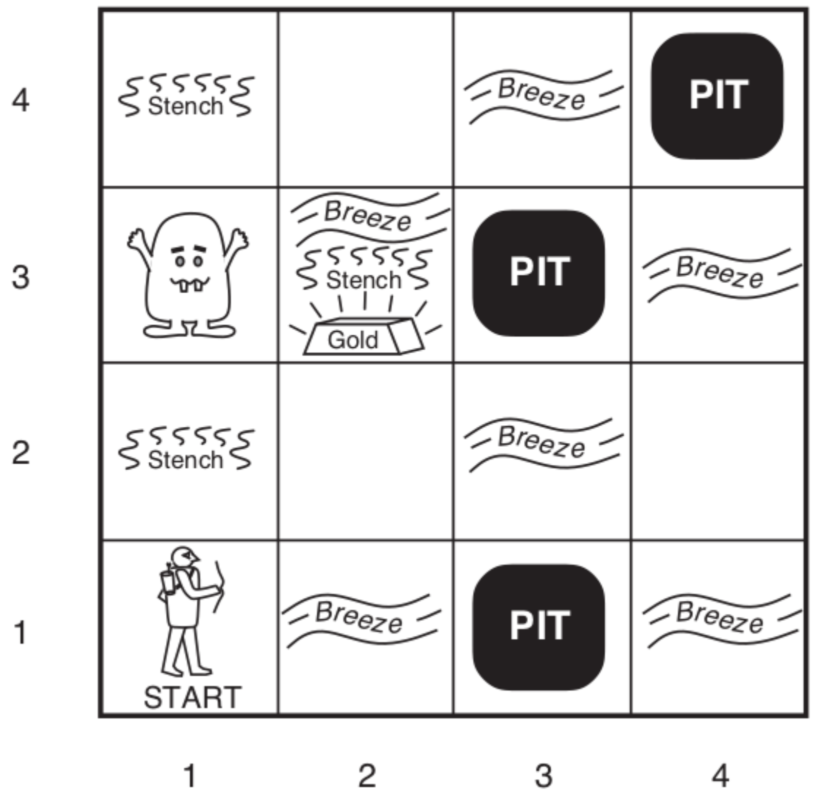
III. Sémantique

Illustration avec le Wumpus...



- Règle 1 : Il n'y a pas de puits en 1,1
- Règle 2 : Autour de chaque puits il y a de la brise
- Règle 3 : Autour du Wumpus il y a une odeur fétide

Application au Wumpus...



Base de connaissance propositionnelle (concernant les puits)

$R_1 : \neg P_{1,1}$

$R_2 : B_{1,1} \leftrightarrow P_{1,2} \lor P_{2,1}$

$R_3 : B_{2,1} \leftrightarrow P_{1,1} \lor P_{2,2} \lor P_{3,1}$

...

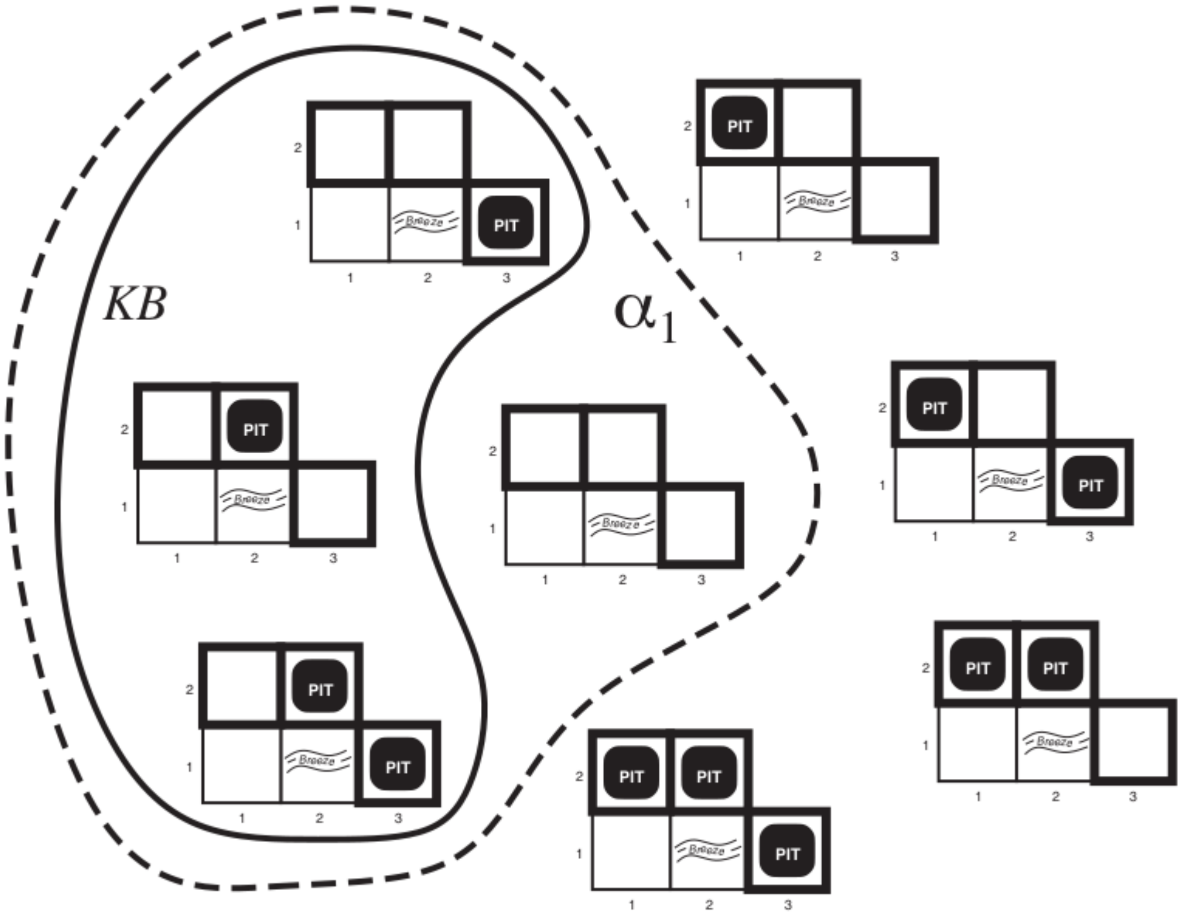
Faits

Le héros est en (1,1) et il ne perçoit rien.

$R_4 : \neg B_{1,1}$

Le héros décide d'aller en (2,1) et il perçoit une brise...

$R_5 : B_{2,1}$



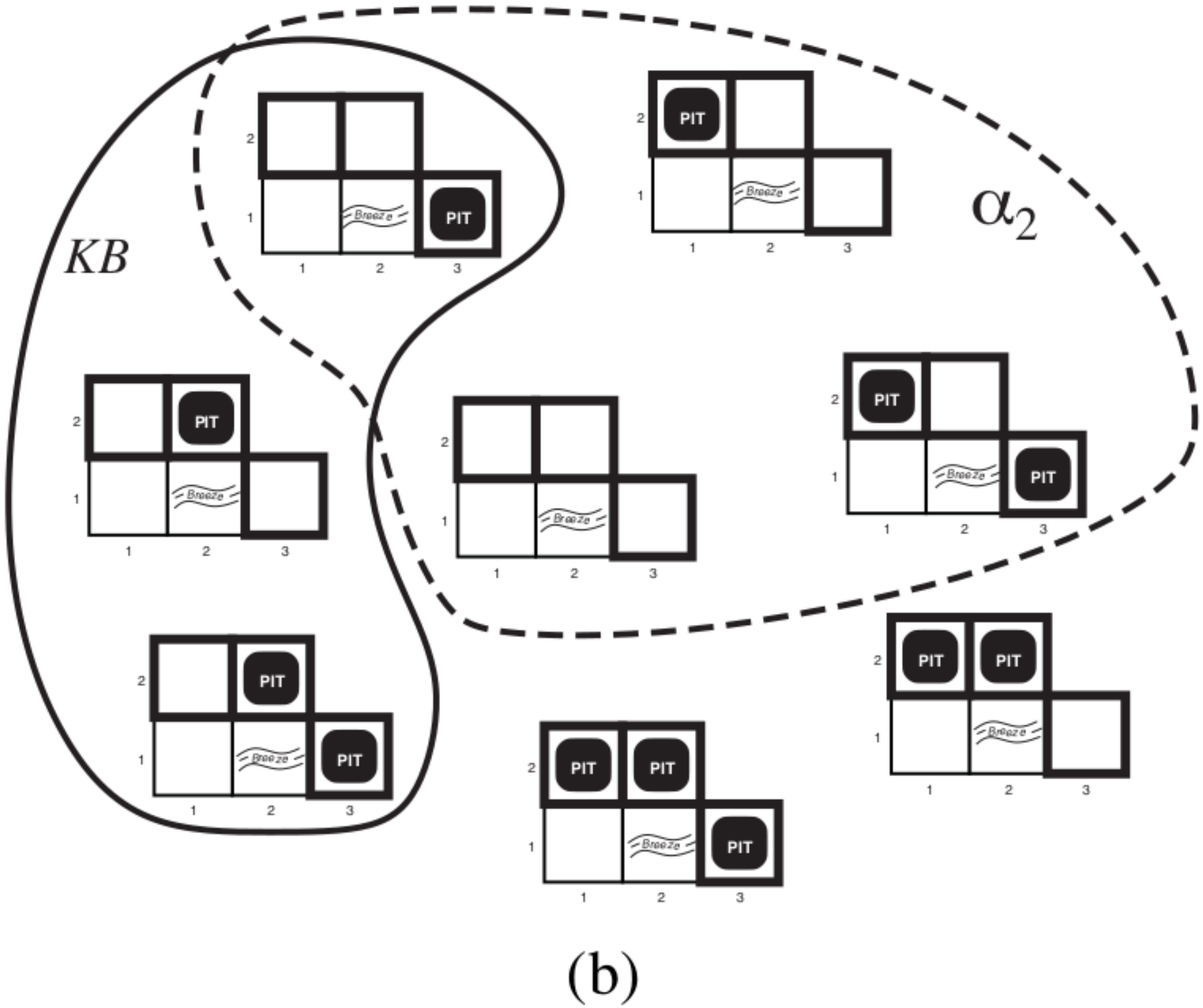
(a)

Peut-on déduire qu'il n'y a pas de puits en (1,2) ?

$KB = \{R_1, R_2, R_3, R_4, R_5\}$

$\alpha_1 : \neg P_{1,2}$

$KB \vDash \alpha_1$ **?**



Peut-on déduire qu'il n'y a pas de puits en (2,2) ?

$\alpha_2 : \neg P_{2,2}$

$KB \vDash \alpha_2$ **?**

IV. Axiomatique et théorie de la démonstration

Objectif : apporter des axiomes et une règle d'inférence permettant de modéliser le raisonnement en se basant uniquement sur la syntaxe.

On introduit pour cela un nouveau symbole de déduction syntaxique : $\vdash$

Définition : la *déduction* (ou preuve, ou démonstration) d'une formule A à partir d'hypothèses $H_1, \ldots, H_m$ (notée $H_1, \ldots, H_m \vdash A$) est une liste finie de formules $(A_1, \ldots, A_n)$ tel que :

- $A_n = A$
- pour $i = 1, \ldots, n$ la formule A_i est :
  - soit un axiome (avec éventuelles substitutions)
  - soit égale à une des hypothèses H_j
  - soit obtenue par application d'une règle d'inférence à des prémisses précédant A_i dans la liste

Un *théorème* est une formule toujours vraie (notée $\vdash A$), c.-à.-d. une formule déductible sans hypothèse.

Un système hilbertien

De David Hilbert (1962-1943), mathématicien allemand et auteur de ses célèbres 23 problèmes.

Schéma d'axiomes

- $A1: \vdash A \rightarrow (B \rightarrow A)$
- $A2: \vdash (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
- $A3: \vdash (\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$

Règle d'inférence

- *Modus Ponens*:

$$\frac{\vdash A, \quad \vdash A \rightarrow B}{\vdash B}$$

Règle de substitution

- Les A, B et C peuvent être remplacés par n'importe quelle formule bien formée

Remarque : il s'agit du plus petit schéma d'axiomes connu à ce jour...

Exemple

Montrons : $\vdash A \to A$

Étape 1 : $\vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C))$ (Axiome 2)

Étape 2 : en substituant $A \to A$ à B et A à C on obtient :

$$\vdash (A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$$

Étape 3 : $\vdash A \to (B \to A)$ (Axiome 1)

Étape 4 : en substituant $A \to A$ à B dans 3 on obtient :

$$\vdash A \to ((A \to A) \to A)$$

Étape 5 : modus ponens entre 4 et 2 permet d'obtenir :

$$\vdash (A \to (A \to A)) \to (A \to A)$$

Étape 6 : en substituant A à B dans l'axiome 1 on obtient :

$$\vdash A \to (A \to A)$$

Étape 7 : modus ponens entre 5 et 6

$$\vdash A \to A$$

Autre schéma d'axiomes hilbertien

- $A \to (B \to A)$
- $(A \to B) \to ((A \to (B \to C)) \to (A \to C))$
- $A \to (B \to A \land B)$
- $(A \land B) \to A$
- $(A \land B) \to B$
- $A \to A \lor B$
- $B \to A \lor B$
- $(A \to C) \to ((B \to C) \to (A \lor B \to C))$
- $(A \to B) \to ((A \to \neg B) \to \neg A)$
- $\neg\neg A \to A$
- $\neg \perp$

Propriétés fondamentales

Propriété 1. (de complétude) Le calcul propositionnel est fortement complet, c'est à dire :

si $E \vDash A$ alors $E \vdash A$

Corollaire. Le calcul propositionnel est faiblement complet :

si $\vDash A$ alors $\vdash A$

Proposition 2. (d'adéquation) Le calcul propositionnel est adéquat :

si $\vdash A$ alors $\vDash A$

Propriétés fondamentales

Proposition 3. (de consistance) Le calcul propositionnel est consistant :

il n'existe pas de formule A telle que $\vdash A$ et $\vdash \neg A$

C'est une absence de paradoxe.

> *Paradoxe de Russell : l'ensemble des ensembles n'appartenant pas à eux-mêmes appartient-il à lui-même ?*

Propriétés fondamentales

Proposition 4. (de décidabilité) Le calcul des propositions est décidable, c'est-à-dire qu'il existe une procédure mécanique permettant d'établir en un temps **fini** si une formule est un théorème ou n'est pas un théorème.

Exemple : tables de vérité...

Proposition 5. Le calcul des propositions n'est pas syntaxiquement complet, c'est-à-dire qu'il peut exister des formules φ tel qu'on ait ni $\vdash \varphi$, ni $\vdash \neg \varphi$.

Règles de déductions à partir de faits...

Un système incomplet mais utilisable...

- Modus Ponens : $\dfrac{A \rightarrow B, A}{B}$
- Élimination de la conjonction : $\dfrac{A \wedge B}{A}$
- Élimination de l'équivalence : $\dfrac{A \leftrightarrow B}{(A \rightarrow B) \wedge (B \rightarrow A)}$
- Apparition de l'équivalence : $\dfrac{(A \rightarrow B) \wedge (B \rightarrow A)}{A \leftrightarrow B}$
- Contraposée : $\dfrac{A \rightarrow B}{\neg B \rightarrow \neg A}$
- Règles de Morgan : $\dfrac{\neg (A \vee B)}{\neg A \wedge \neg B}$
- Règles de Morgan : $\dfrac{\neg (A \wedge B)}{\neg A \vee \neg B}$
- Double négation : $\dfrac{\neg (\neg A)}{A}$

On supposera acquise l'associativité du $\wedge$ et du $\vee$

Wumpus II: le retour...

- $R_1 : \neg P_{1,1}$
- $R_2 : B_{1,1} \leftrightarrow P_{1,2} \vee P_{2,1}$
- $R_3 : B_{2,1} \leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$
- $R_4 : \neg B_{1,1}$
- $R_5 : B_{2,1}$

Soit $KB = \{R_1, R_2, R_3, R_4, R_5\}$, montrons que $KB \vdash \neg P_{1,2}$

- R_2 + élimination de l'équivalence donne $R_6 : (B_{1,1} \rightarrow P_{1,2} \vee P_{2,1}) \wedge (P_{1,2} \vee P_{2,1} \rightarrow B_{1,1})$
- R_6 + élimination de la conjonction donne $R_7 : P_{1,2} \vee P_{2,1} \rightarrow B_{1,1}$
- R_7 + contraposée donne $R_8 : \neg B_{1,1} \rightarrow \neg(P_{1,2} \vee P_{2,1})$
- R_4 + R_8 + Modus Ponens donne $R_9 : \neg(P_{1,2} \vee P_{2,1})$
- R_9 + de Morgan donne $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$
- R_{10} + élimination de la conjonction donne $R_{11} : \neg P_{1,2}$

V. Clauses et calcul propositionnel

Formes normales disjonctives

Définition : un **cube** est une conjonction de littéraux

Exemple : $a \wedge b \wedge \neg c$

Définition : une **forme normale disjonctive** (DNF) est une disjonction de cubes.

Exemples :

$(a \wedge b) \vee (\neg a \wedge \neg a) \vee (a \wedge b \wedge \neg c)$ est une forme disjonctive (mais pas normale, elle contient des cubes non purs)

$(a \wedge b) \vee (\neg a \wedge b) \vee (a \wedge b \wedge \neg c)$ est une DNF

Remarque 1 : pour vérifier si une DNF est valide, il suffit de vérifier un à un ses cubes.

Remarque 2 : une DNF peut être de taille exponentielle, par exemple $\top$.

Remarque 3 : les cubes sont dits purs si une variable n'apparaît qu'une seule fois.

V. Clauses et calcul propositionnel

Formes normales conjonctives

Définition : une **clause** est une disjonction de littéraux. Une clause est pure si chaque variable n'apparaît au plus qu'une seule fois.

Exemple : $a \lor b \lor \neg c$

Définition : une **forme normale conjonctive** (CNF) est une conjonction de clauses pures.

Exemple : $(a \lor b) \land (\neg a \lor b) \land (a \lor b \lor \neg c)$ est une CNF.

Remarque 1 : la taille d'une CNF peut également être exponentielle.

Remarque 2 : cette forme est souvent plus utile pour représenter des connaissances...

Remarque 3 : ... mais la recherche de validité n'est plus immédiate.

V. Clauses et calcul propositionnel

Autres écritures

Écriture implicative

- $\neg a \lor b$ peut s'écrire $a \to b$
- $\neg a \lor \neg b \lor c$ peut s'écrire $a \land b \to c$, voire $a, b \to c$
- $\neg a \lor b \lor c$ peut s'écrire $a \to b \lor c$, voire $a \to b, c$
- $\neg a \lor \neg b \lor c \lor d$ peut s'écrire $a \land b \to c \lor d$, voire $a, b \to c, d$
- a peut s'écrire $\top \to a$ ou $\to a$
- $\neg a$ peut s'écrire $a \to \bot$ ou $a \to$

V. Clauses et calcul propositionnel

Écriture ensembliste

$C_1 \wedge C_2 \wedge \ldots \wedge C_n$ peut s'écrire sous forme ensembliste : $\{C_1, C_2, \ldots, C_n\}$ voire sous forme d'ensembles d'ensembles.

Exemple : $(a \vee b) \wedge (\neg a \vee b) \wedge (a \vee b \vee \neg c)$ peut s'écrire :

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

Le problème d'existance de modèle (problème de satisfiabilité) devient :

> *Il existe $\omega \in \Omega$ tel que quel que soit $C \in N$, il existe $l \in C$ tel que $\omega \models l$*

Théorème de normalisation

Théorème. Toute formule peut se mettre sous forme CNF (resp. DNF).

Pour cela, on utilise les règles suivantes :

1. tous les $A \leftrightarrow B$ se réécrivent en $(A \rightarrow B) \wedge (B \rightarrow A)$
2. tous les $A \rightarrow B$ se réécrivent en $\neg A \vee B$
3. on utilise les règles de Morgan :
   - $\neg(A \vee B)$ se réécrit $\neg A \wedge \neg B$
   - $\neg(A \wedge B)$ se réécrit $\neg A \vee \neg B$
4. $\neg\neg A$ se réécrit A
5. on utilise la distributivité du $\wedge$ et du $\vee$:
   - $A \vee (B \wedge C)$ se réécrit $(A \vee B) \wedge (A \vee C)$
   - $A \wedge (B \vee C)$ se réécrit $(A \wedge B) \vee (A \wedge C)$

Remarque : La conjonction de 2 CNF est une CNF ⇒ application récursive des règles de transformation quand on a une conjonction entre 2 formules quelconques

Exemple

Mettre sous forme CNF la formule $(p \rightarrow (q \rightarrow r)) \rightarrow (p \wedge s \rightarrow r)$

- Suppression des implications : $\neg(\neg p \vee (\neg q \vee r)) \vee (\neg(p \wedge s) \vee r)$
- Règle de Morgan : $\neg(\neg p \vee \neg q \vee r) \vee ((\neg p \vee \neg s) \vee r)$
- Règle de Morgan : $(p \wedge q \wedge \neg r) \vee (\neg p \vee \neg s \vee r)$
- Distribution : $(p \vee \neg p \vee \neg s \vee r) \wedge (q \vee \neg p \vee \neg s \vee r) \wedge (\neg r \vee \neg p \vee \neg s \vee r)$
- Associativité du $\vee$: $(p \vee \neg p \vee \neg s \vee r) \wedge (q \vee \neg p \vee \neg s \vee r) \wedge (\neg r \vee r \vee \neg p \vee \neg s)$
- Tautologies... : $(\top \vee \neg s \vee r) \wedge (q \vee \neg p \vee \neg s \vee r) \wedge (\top \vee \neg p \vee \neg s)$
- Tautologies et $\vee$: $\top \wedge (q \vee \neg p \vee \neg s \vee r) \wedge \top$
- Tautologies et $\wedge$: $q \vee \neg p \vee \neg s \vee r$

V. Clauses et calcul propositionnel

Wumpus III le retour...

Mettre les règles du Wumpus sous forme de clauses...

- $R_1: \neg P_{1,1}$
- $R_2: B_{1,1} \leftrightarrow P_{1,2} \vee P_{2,1}$
- $R_3: B_{2,1} \leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$
- $R_4: \neg B_{1,1}$
- $R_5: B_{2,1}$

OK pour R_1, R_4 et R_5

Pour R_2...

V. Clauses et calcul propositionnel

- $R_2 : B_{1,1} \leftrightarrow P_{1,2} \vee P_{2,1}$

1. Suppression des équivalence : $(B_{1,1} \rightarrow P_{1,2} \vee P_{2,1}) \wedge (P_{1,2} \vee P_{2,1} \rightarrow B_{1,1})$
2. Suppression des implications : $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
3. De Morgan : $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
4. Distributivité du $\vee$: $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

En exercice : R_3...

Principe de résolution

Théorème. Soit N une forme normale conjonctive et C_1 et C_2 deux clauses de N. Soit p un atome tel que $p \in C_1$ et $\neg p \in C_2$. Soit la clause $R = C_1 \setminus \{p\} \cup C_2 \setminus \{\neg p\}$ alors la CNF N et $N \cup R$ sont équivalentes.

Ou encore sous forme de règle de réécriture :

$$\frac{A \vee B, \quad \neg B \vee C}{A \vee C}$$

Preuve : il suffit de montrer que $A \vee B, \neg B \vee C \vDash A \vee C$

V. Clauses et calcul propositionnel

- Pour montrer qu'un ensemble de clauses est incohérent, on montre que l'on peut déduire la clause vide (autrement dit ⊥)

Exemple

$(\neg p \lor r) \land (\neg q \lor r) \land (p \lor q) \land \neg r$

Version ensembliste :

$\{\{\neg p, r\}, \{\neg q, r\}, \{p, q\}, \{\neg r\}\}$

$\{\{\neg p, r\}, \{\mathbf{\neg q, r}\}, \{p, q\}, \{\mathbf{\neg r}\}\}$

$\{\{\mathbf{\neg p, r}\}, \{\neg q, r\}, \{p, q\}, \{\mathbf{\neg r}\}, \{\neg q\}\}$

$\{\{\neg p, r\}, \{\neg q, r\}, \{\mathbf{p, q}\}, \{\neg r\}, \{\neg q\}, \{\mathbf{\neg p}\}\}$

$\{\{\neg p, r\}, \{\neg q, r\}, \{p, q\}, \{\neg r\}, \{\mathbf{\neg q}\}, \{\neg p\}, \{\mathbf{q}\}\}$

$\{\{\neg p, r\}, \{\neg q, r\}, \{p, q\}, \{\neg r\}, \{\neg q\}, \{\neg p\}, \{q\}, \varnothing\}$

V. Clauses et calcul propositionnel

Procédure automatique : Davis et Putnam (1960)

Entrée: une CNF N

1. Si $N = \varnothing$ alors N est cohérente
2. Si $\varnothing \in N$ alors N est incohérente sinon
    1. Choisir un atome p
    2. $N_p = \{\text{clauses contenant } p\}$
    3. $N_{\neg p} = \{\text{clauses contenant } \neg p\}$
    4. $N_c = N \setminus (N_p \cup N_{\neg p})$
    5. Calculer $N_p' = \{N_p, \text{ sans } p\}$ */* cas ou p est faux */*
    6. Calculer $N_{\neg p}' = \{N_{\neg p}, \text{ sans } \neg p\}$ */* cas ou p est vrai */*
    7. N est incohérent si $N_p' \cup N_c$ et $N_{\neg p}' \cup N_c$ le sont également

V. Clauses et calcul propositionnel

Remarques

- L’algorithme termine toujours et est complet (on balaie l'ensemble des littéraux possibles)
- **Amélioration possible** : commencer par les clauses unitaires
- Dans le pire des cas, l'algorithme est exponentiel...

En théorie

Théorème de Cook-Levin (1961)

> *Sous l'hypothèse que P ≠ NP, le problème de satisfiabilité d'une formule booléenne est NP-complet.*

NB : un problème est dans NP si il est décidable par une machine de Turing non déterministe en un temps polynomial. Un problème est dans la classe P s'il est décidable en temps polynomial

En pratique…

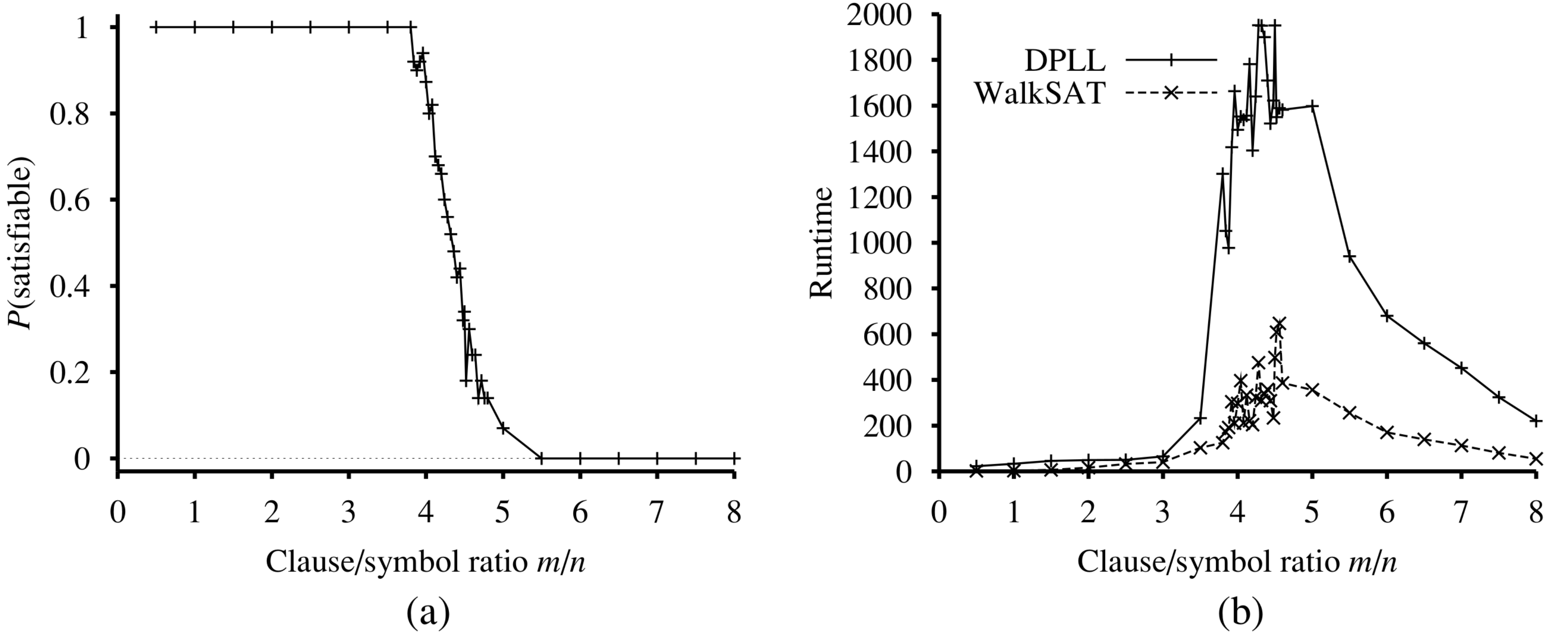
Les problèmes issus de vrais problèmes sont solubles très rapidement. Les *solvers* modernes peuvent gérer des millions de clauses et des dizaines de milliers de variables.

Quelques solver opensources :

- Glucose http://www.labri.fr/perso/lsimon/glucose/, issu de minisat http://minisat.se/ en C++
- SAT4J http://www.sat4j.org/ en Java
- gophersat https://github.com/crillab/gophersat en Go
- pysat https://github.com/pysathq/pysat en Python

Transition de phase

Les problèmes les plus difficiles sont générés aléatoirement et ont un rapport #clauses/#variables d'environ 4.3...



V. Clauses et calcul propositionnel

Les clauses de Horn

Idée : puisqu'il ne semble pas exister d'algorithme toujours efficace, on peut se concentrer sur des fragments de la logique propositionnelle pour résoudre le problème SAT.

Exemples : 2-SAT (mais pas 3-SAT !), Horn, Horn renommable, etc.

Définition. Une clause de Horn est une clause où apparaît au plus un littéral positif. *Exemples :*

- clause de Horn stricte : $a \lor \neg b \lor \neg c \lor \neg d \lor \neg e$
- clause de Horn négative : $\neg a \lor \neg b \lor \neg c$
- clause de Horn positive : a *Question :* et sous forme implicative ?

Les clauses de Horn

Définition. Une clause sous-sommée (subsumée) est une clause pouvant être déduite par une autre clause de la base de clauses.

Exemple : $a \lor \neg b \lor \neg c$ est sous-sommée par $a \lor \neg b$

Remarque : Lors de la recherche de modèles, on peut supprimer toutes les clauses sous-sommées.

V. Clauses et calcul propositionnel

Les clauses de Horn

Application aux clauses de Horn :

Une clauses positive p permet :

- d'enlever toutes les clauses qui contiennent p
- de réduire les clauses qui contiennent $\neg p$ (propagation unitaire)

Exemple

- $\{a \lor \neg b \lor \neg c, \neg a \lor b, a\}$
- $\{\neg a \lor b, a\}$
- $\{b, a\}$

Une clause unitaire négative $\neg p$ permet :

- d'enlever toutes les clauses qui contiennent $\neg p$
- de réduire les clauses qui contiennent p (propagation unitaire)

V. Clauses et calcul propositionnel

Les clauses de Horn

Algorithme

1. On applique toutes les propagations unitaires
2. On supprime toutes les clauses sous-sommées
3. Si on obtient la clause vide, l'ensemble est inconsistant
4. Sinon, on peut exhiber un modèle

Exercice :

1. $\{\neg a \lor \neg b, \neg c \lor d, a, \neg a \lor \neg d\}$
2. $\{\neg p \lor r, \neg r \lor s, p, \neg r\}$

VI. Modéliser et résoudre des problèmes en logique propositionnelle

Objectif

- Utiliser la logique propositionnelle pour modéliser un problème et utiliser un solveur SAT pour le résoudre

Méthode/démarche systématique

- **Étape 1 :** Choix du vocabulaire
- **Étape 2 :** Modélisation du problème/de la base de connaissance KB en logique propositionnelle
- **Étape 3 :** Mise sous forme clausale
- **Étape 4 :** Vérifier la cohérence de KB (via SAT)
- **Étape 5 :** Encoder une requête en se ramenant à un problème SAT

VI. Modéliser et résoudre des problèmes en logique propositionnelle

Étape 1 : Choix du vocabulaire

- Il peut être utile de détecter que deux variables sont synonymes/équivalentes ($petit\_lutin\_bleu \leftrightarrow Schtroumpf$) ou antonymes/contraires ($sorcier\_competent \leftrightarrow \neg gargamel$)
- Problème des variables qui comprennent plus de 2 valeurs (exemple des couleurs)
- Il pourra être nécessaire de renommer certains atomes afin de mettre des clauses sous forme de Horn

Étape 2 : Modélisation du problème/de la base de connaissance KB en logique propositionnelle

- Problème d'ambiguïté du langage...
- C'est l'étape faisant le plus intervenir de savoir-faire et d'intelligence humaine...

VI. Modéliser et résoudre des problèmes en logique propositionnelle

Étape 3 : Mise sous forme clausale

- On reprend la méthode mécanique présentée précédemment...
- On peut avoir besoin d'un programme pour générer entièrement le problème...

Étape 4 : Vérifier la cohérence de KB (via SAT)

- ÉTAPE ABSOLUMENT NÉCESSAIRE
- En cas d'incohérence, on pourra tout déduire et son contraire

Étape 5 : Encoder une requête en se ramenant à un problème SAT

- Test de satisfiabilité (SAT)
- Trouver une conséquence
- Trouver une conséquence conditionnelle
- Compter les modèles
- ...

VI. Modéliser et résoudre des problèmes en logique propositionnelle

Différentes requêtes possibles

Satisfiabilité

- découverte d'au moins 1 modèle, c.-à-d. une assignation solution du problème modélisé
- absolument nécessaire avant de lancer des requêtes depuis KB
- difficilement faisable à la main sur un problème réel... On utilisera des solveurs SAT dédiés

VI. Modéliser et résoudre des problèmes en logique propositionnelle

Trouver une conséquence logique

- $KB \vDash C$?
- $KB \cup \{\neg C\} \vDash \bot$
- $KB \cup \{\neg C\} \vdash \bot$
- *remarque 1 :* $\neg C$ doit être mise sous forme clausale
- *remarque 2 :* on peut inférer (déduire) C, $\neg C$ ou ni l'un ni l'autre
- *remarque 3 :* on peut toujours ajouter une conséquence à la base de départ, cela peut aider le solveur (surtout dans le cas de clauses unitaires)

VI. Modéliser et résoudre des problèmes en logique propositionnelle

Trouver une conséquence conditionnelle

- Si H est vrai, puis-je déduire C ?
- $KB \vDash H \rightarrow C$?
- $KB \cup \{\neg(H \rightarrow C)\} \vdash \bot$
- $KB \cup \{\neg(\neg H \vee C)\} \vdash \bot$
- $KB \cup \{(H \wedge \neg C)\} \vdash \bot$
- $KB \cup \{H, \neg C\} \vdash \bot$
- Ce qui revient à $KB \cup \{H\} \vdash C$!

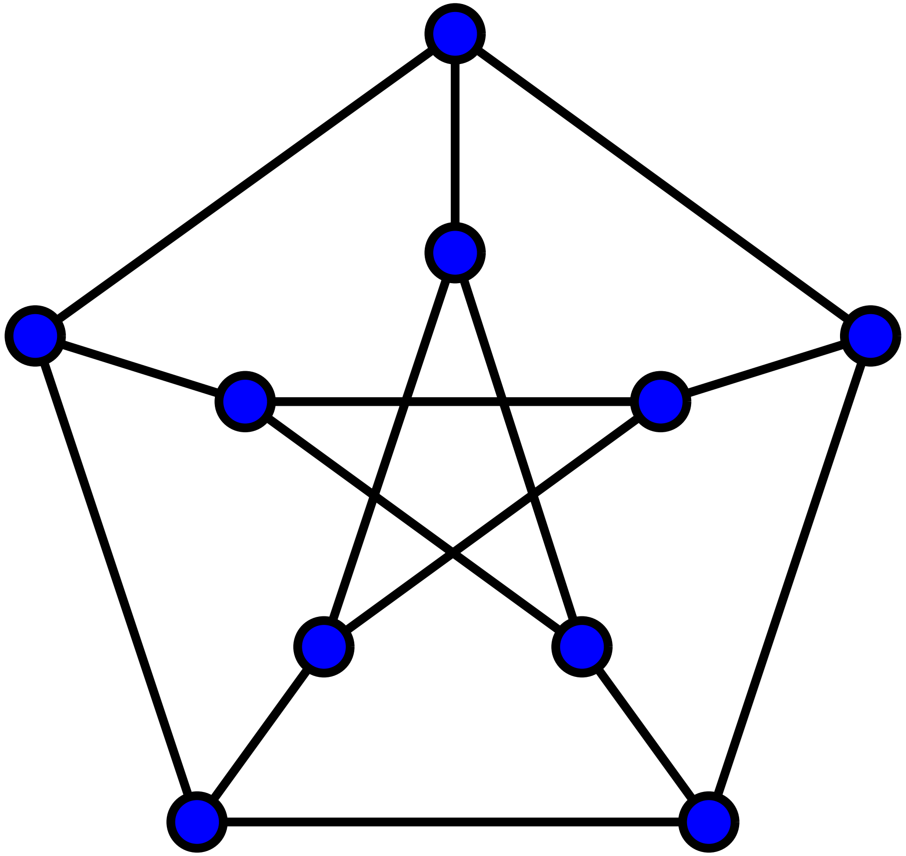
VI. Modéliser et résoudre des problèmes en logique propositionnelle

Compter les modèles/avoir l'ensemble des modèles

- Si KB est cohérente (satisfiable), le solveur renvoie une interprétation
- Un modèle peut être vu comme une conjonction de littéraux, ex. : $a \wedge \neg b \wedge c$
- Pour avoir l'ensemble des modèles on ajoute la négation de la conjonction à KB, ex. :
  - $\neg(a \wedge \neg b \wedge c)$
  - $\neg a \vee b \vee \neg c$ (c'est une clause !)
- On recommence jusqu'à tomber sur l'incohérence...

Exemple : coloration d'un graphe avec 3 couleurs

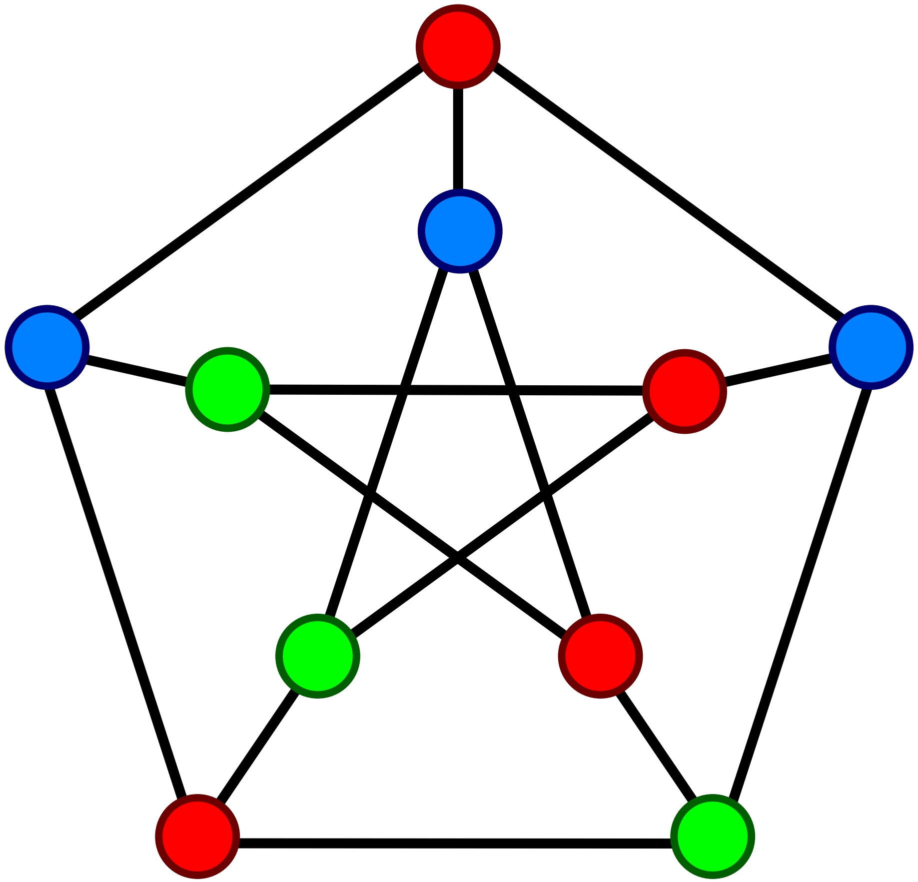
- Un **graphe** est un ensemble de nœuds/sommets et d'arêtes/arcs.



- Réprésentation informatique : matrice d'adjacence ou liste de sommets adjacents...

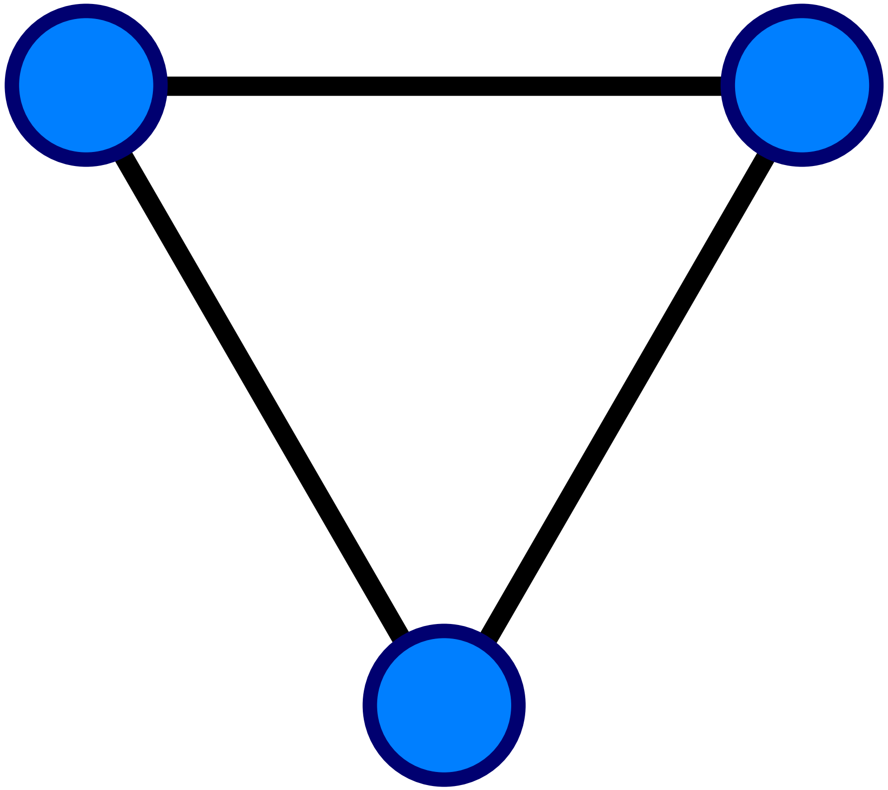
Exemple : coloration d'un graphe avec 3 couleurs

- **Problème de coloration d'un graphe :** 2 sommets adjacents (reliés par un arc) ne peuvent pas avoir la même couleur.
- Dualement, on peut colorier les arcs...



Exemple : coloration d'un graphe avec 3 couleurs

- **Question** : comment encoder le problème de coloration à 3 couleurs du graphe suivant ?



Exemple : coloration d'un graphe avec 3 couleurs

Étape 1 : choix des variables

- On considère 3 couleurs (R, G, B).
- Problème : les variables booléennes ne peuvent prendre que 2 valeurs...
- Couleurs du sommet 1 : $S1R$, $S1G$, $S1B$
- Couleurs du sommet 2 : $S2R$, $S2G$, $S2B$
- Couleurs du sommet 3 : $S3R$, $S3G$, $S3B$

Exemple : coloration d'un graphe avec 3 couleurs

Étape 2 : modélisation du problème

- Chaque sommet doit être colorié par une couleur :

$$S1R \lor S1G \lor S1B$$

$$S2R \lor S2G \lor S2B$$

$$S3R \lor S3G \lor S3B$$

- Chaque sommet ne peut être colorié qu'avec une seule couleur :

$$S1R \leftrightarrow \neg S1G \land \neg S1B$$

$$S1G \leftrightarrow \neg S1R \land \neg S1B$$

$$S1B \leftrightarrow \neg S1R \land \neg S1G$$

$$\ldots$$

$$S3B \leftrightarrow \neg S3R \land \neg S3G$$

Exemple : coloration d'un graphe avec 3 couleurs

- Chaque sommet a une couleur différente des sommets adjacents :

$$S1R \rightarrow \neg S2R \land \neg S3R$$

$$S1G \rightarrow \neg S2G \land \neg S3G$$

$$S1B \rightarrow \neg S2B \land \neg S3B$$

$$\ldots$$

$$S3B \rightarrow \neg S1B \land \neg S2B$$

Étape 3 : Mise sous forme clausale

$$S1R \lor S1G \lor S1B$$

$$S1R \leftrightarrow \neg S1G \land \neg S1B$$

$$(S1R \rightarrow \neg S1G \land \neg S1B) \land (\neg S1G \land \neg S1B \rightarrow S1R)$$

$$(\neg S1R \lor (\neg S1G \land \neg S1B)) \land (\neg(\neg S1G \land \neg S1B) \lor S1R)$$

$$(\neg S1R \lor \neg S1G) \land (\neg S1R \lor \neg S1B) \land (S1G \lor S1B \lor S1R)$$

Exemple : coloration d'un graphe avec 3 couleurs

$$S1R \lor S1G \lor S1B$$

$$\neg S1R \lor \neg S1G$$

$$\neg S1R \lor \neg S1B$$

$$\neg S1G \lor \neg S1B$$

$$S2R \lor S2G \lor S2B$$

$$\neg S2R \lor \neg S2B$$

$$\neg S2G \lor \neg S2B$$

$$\neg S2G \lor \neg S2R$$

$$S3R \lor S3G \lor S3B$$

$$\neg S3R \lor \neg S3B$$

$$\neg S3G \lor \neg S3B$$

$$\neg S3G \lor \neg S3R$$

Exemple : coloration d'un graphe avec 3 couleurs

$$S1R \rightarrow \neg S2R \wedge \neg S3R$$

$$\neg S1R \vee (\neg S2R \wedge \neg S3R)$$

$$(\neg S1R \vee \neg S2R) \wedge (\neg S1R \vee \neg S3R)$$

Exemple : coloration d'un graphe avec 3 couleurs

$$\neg S1R \lor \neg S2R$$

$$\neg S1R \lor \neg S3R$$

$$\neg S2R \lor \neg S3R$$

$$\neg S1G \lor \neg S2G$$

$$\neg S1G \lor \neg S3G$$

$$\neg S2G \lor \neg S3G$$

$$\neg S1B \lor \neg S2B$$

$$\neg S1B \lor \neg S3B$$

$$\neg S2B \lor \neg S3B$$

La base de clause entière

$S1R \lor S1G \lor S1B$

$\neg S1R \lor \neg S1B$

$\neg S1G \lor \neg S1B$

$\neg S1G \lor \neg S1R$

$S2R \lor S2G \lor S2B$

$\neg S2R \lor \neg S2B$

$\neg S2G \lor \neg S2B$

$\neg S2G \lor \neg S2R$

$S3R \lor S3G \lor S3B$

$\neg S3R \lor \neg S3B$

$\neg S3G \lor \neg S3B$

$\neg S3G \lor \neg S3R$

$\neg S1R \lor \neg S2R$

$\neg S1R \lor \neg S3R$

$\neg S2R \lor \neg S3R$

$\neg S1G \lor \neg S2G$

$\neg S1G \lor \neg S3G$

$\neg S2G \lor \neg S3G$

$\neg S1B \lor \neg S2B$

$\neg S1B \lor \neg S3B$

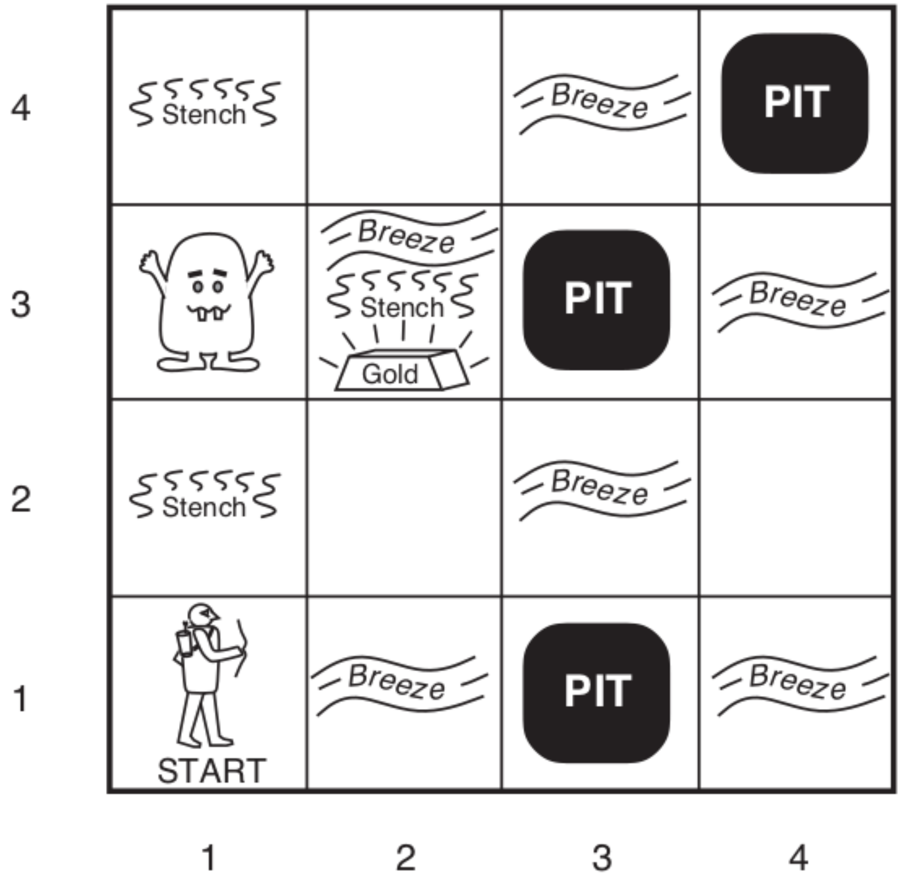
$\neg S2B \lor \neg S3B$

Les requêtes possibles :

- Existe-t-il une solution ?
- Existe-t-il une solution en coloriant le sommet 1 en bleu ?
- Combien existe-t-il de colorations différentes ?
- Donner toutes les solutions possibles

Exercice conclusif

Encoder entièrement le problème du Wumpus.



VII. Conclusion/synthèse

Définition de la logique propositionnelle

- Définition formelle du langage propositionnel
- Définition de l'ensemble des formules valides à partir des modèles/interprétations et des tables de vérité $\vDash \varphi$
- Définition de l'ensemble des théorèmes à partir d'axiomes et du *Modus Ponens* $\vdash \varphi$
- Équivalence des 2 approches, complétude et adéquation de la logique propositionnelle

Définition de la conséquence logique à partir d'un ensemble de faits/d'hypothèses

- Définition d'une conséquence logique sémantique basée sur les interprétations $H_1, H_2, \ldots, H_n \vDash C$
- Définition d'une conséquence logique syntaxique basée sur des règles de réécritures $H_1, H_2, \ldots, H_n \vdash C$
- Équivalence des 2 approches

VII. Conclusion/synthèse

Calcul clausal et modélisation de problème

- Toute formule peut s'écrire sous la forme d'une CNF
- À partir de cette forme, seul le principe de résolution et la sous-sommation sont utiles
- Les clause de Horn permettent de résoudre le problème en temps polynomial
- On peut utiliser des solveurs SAT pour résoudre le problème de satisfiabilité ainsi que pour répondre à d'autres requêtes

Quelques limites

- Explosion du nombre de clauses, de variables et problème de lisibilité...
- On ne peut pas encoder des règles du type :

> *Tous les humains sont mortels. Socrate est un humain. Donc Socrate est mortel.*

The end…

To be continued…