Énigmes logiques

Problème

Naufragés sur la planète Golique, le capitaine Lagedu et son équipier Cogito découvrent six astronefs abandonnés. Par un raisonnement logique, ils trouvent lequel de ces appareils est paré à décoller.
Saurez-vous en faire autant ?

– Cogito : *Ces astronefs semblent en bon état.*
– Lagedu : *Ne vous y fiez pas, Cogito ! Depuis le temps, les moteurs doivent être hors-service ! !*
– Cogito : *Tout de même, capitaine, sur six appareils, il y en a sûrement un capable de décoller !*
– Lagedu : *Certes ! mais allez savoir lequel !*
– Cogito : *Il suffit de consulter pour chaque appareil ses deux ordinateurs de bord...*
– Lagedu : *Mais voyons ! Ces ordinateurs de bord aussi peuvent être déréglés ! Et vous savez qu'un ordinateur déréglé donne au moins une réponse fausse !*

Dans l'appareil n°1
– Ordinateur gauche : appareil n°1 paré à décoller ; ordinateur connexe hors service.
– Ordinateur droite : appareil n°1 paré à décoller ; ordinateur connexe fonctionne 5 sur 5.
– Lagedu : *Qu'est-ce que je vous disais ? Nous voilà bien avancés !*
– Cogito : *Il faut voir...*

Dans l'appareil n°2
– Ordinateur gauche : appareil n°2 paré à décoller ; ordinateurs connexes tous deux hors service.
– Ordinateur droite : ordinateur connexe fonctionne 5 sur 5.

Dans l'appareil n°3
– Ordinateur gauche : appareil n°3 paré à décoller ; au moins un des ordinateurs connexes hors service.
– Ordinateur droite : ordinateur connexe fonctionne 5 sur 5.

Dans l'appareil n°4
– Ordinateur gauche : appareil n°4 interdit de vol ; ordinateur connexe hors service.
– Ordinateur droite : appareil n°4 interdit de vol ; ordinateur connexe fonctionne 5 sur 5.

Dans l'appareil n°5
– Ordinateur gauche : appareil n°5 interdit de vol ; au moins un des ordinateurs connexes fonctionne 5 sur 5.
– Ordinateur droite : appareil n°5 paré à décoller ; ordinateurs connexes tous deux hors service.
– Lagedu : *Il n'y a rien à tirer de ces inepties, Cogito ! Vous perdez votre temps ! !*

Dans l'appareil n°6
– Ordinateur gauche : appareil n°6 paré à décoller ; ordinateurs connexes tous deux hors service.
– Ordinateur droite : appareil n°6 interdit de vol ; au moins un des ordinateurs connexes hors service.
– Lagedu : *Je vous l'avais bien dit : ces appareils n'ont pas été abandonnés sans raison, ils sont inutilisables !*
– Cogito : *Pas tous ! Venez, quittons ce coin perdu !*

Question 1

Voici un extrait de la solution parue dans « Jeux et Stratégies ».
Supposons que le premier astronef soit en état de marche (première supposition). Si l'ordinateur de gauche est en panne (deuxième supposition), il donne au moins une affirmation fausse. Sa première affirmation étant vraie (appareil paré à décoller), la deuxième est fausse, c'est-à-dire que l'ordinateur connexe fonctionne normalement : ce qu'il dit est vrai. Or, il affirme que l'ordinateur de gauche fonctionne bien. Il y a incompatibilité. Changeons donc notre deuxième supposition et posons à présent que l'ordinateur de gauche marche bien. S'il dit vrai, l'ordinateur de droite est déréglé et donne au moins une fausse indication. Or il affirme que l'appareil est prêt à décoller (ce qui est conforme à la première supposition) et que l'ordinateur de gauche fonctionne (ce qui est conforme à la deuxième). II y a de nouveau incompatibilité. Quelle que soit donc notre deuxième supposition, nous avons abouti à une impossibilité : la première supposition est donc fausse et le premier astronef n'est pas en état de marche.

En vous inspirant de ce style, indiquez sur lequel des six astronefs se sont embarqués Lagedu et Cogito.

Question 2

On va formaliser un peu le problème précédent.
Pour chaque astronef, on note $\mathcal{P}$ la proposition « *l'appareil est prêt à décoller* ».
On note aussi $\mathcal{G}$ (resp. $\mathcal{D}$) pour « *l'ordinateur de gauche (resp. de droite) fonctionne 5 et 5* ».
Traduire les données du problème à l'aide de ce formalisme, et répondre à la question.

Question 3

On va utiliser un formalisme plus efficace encore.
A toute proposition $\mathcal{A}$, on associe la *variable booléenne* a définie par $\begin{cases} a=1 & \text{si } \mathcal{A} \text{ est vraie} \\ a=0 & \text{si } \mathcal{A} \text{ est fausse} \end{cases}$

1. Soient a et b les variables associées à deux propositions $\mathcal{A}$ et $\mathcal{B}$.
   Avec ces notations, les propositions $\mathcal{A} \Leftrightarrow \mathcal{B}$ et $a=b$ sont bien sûr synonymes.
   Quelles sont les variables associées aux propositions $\overline{\mathcal{A}}$, $\mathcal{A}$ et $\mathcal{B}$, $\mathcal{A}$ ou $\mathcal{B}$, $\mathcal{A} \Rightarrow B$, $\mathcal{A} \Leftrightarrow \mathcal{B}$?
2. Traduire les données du problème à l'aide de ce formalisme, et répondre à la question.

Question 4

Les habitants de la planète Olgique sont de deux types.
– Les « anciens » qui mentent le mardi, le mercredi, le jeudi, et qui disent la vérité les autres jours.
– Les « modernes » qui mentent le vendredi, samedi, dimanche, et qui disent la vérité les autres jours.

Dans un long voyage vers le temple de Vérisonge, vous rencontrez des anciens et des modernes, qui vous indiquent à leur manière quel jour on est dans la semaine.
– Le jour du départ, un moderne : *j'ai menti hier* ; un ancien : *j'ai menti hier.*
– Première étape, un ancien : *je mentais hier* ; un ancien : *je mentirai dans trois jours.*
– Deuxième étape, un moderne : *hier, je mentais* ; un ancien : *ce moderne ment.*
– A l'arrivée, un moderne : *je mentirai demain comme hier* ; un ancien : *on n'est pas vendredi.*

Précisez le jour de la semaine, pour chacune des quatre étapes.

Question 5

Le capitaine Lagedu et son équipier Cogito sont chargés d'enquêter sur différents braquages, commis (enfin on sait que les coupables sont parmi ces trois là) par Alfred, Baptiste et Charly. Ils n'ont à leur disposition que des indices détournés mais certains, fournis par les mutants visionnaires du puits de la vérité.

– Pour le hold-up de la bijouterie de Gemma : *Si Charly est innocent, Alfred est coupable. Si Alfred est coupable, il a agi avec un complice et un seul. Si Baptiste n'a pas trempé dans cette affaire, Charly non plus. S'il y a deux responsables dans cette affaire, Alfred est l'un d'eux.*

– Pour le casse de la banque de Prokur : *Si Baptiste a trempé dans cette affaire, Charly aussi. Pour les hold-up de banques, Alfred a horreur de faire équipe avec Charly. Si Alfred est coupable et Baptiste innocent, alors Charly est coupable. Charly n'a pas pu faire ce genre de boulot tout seul.*

– Pour le vol dans la pharmacie de Narcozia : *Si Alfred a trempé dans cette affaire, Baptiste non. Si Baptiste est coupable, il avait un complice et un seul. Si Charly est coupable, Alfred et Baptiste le sont aussi.*

Précisez le ou les coupables dans chacune de ces trois affaires.

Question 6

Les habitants de la planète Giloque sont de deux types.
Les « menteurs » mentent toujours, et les « changeants » parfois mentent et parfois disent la vérité.
Vous rencontrez quatre habitants X, Y, Z, T (dont au moins deux menteurs) de cette planète qui vous disent :

– X : de deux choses l'une, ou bien Y est changeant, ou bien T est changeant.
– Y : si je suis un menteur, X est un changeant.
– Z : ou bien je suis un menteur et Y est un changeant, ou bien je suis un changeant et Y est un menteur.
– T : ce problème n'était pas intéressant.

Ce problème était-il intéressant ?

Corrigé du problème

Question 1

- Pour l'appareil n°1 :

  Une solution a été donnée dans l'énoncé. Voici une autre rédaction possible : Supposons que l'ordinateur de droite fonctionne normalement. Ce qu'il dit est vrai, et notamment l'ordinateur de gauche fonctionne 5 sur 5. Ce que dit celui-ci est donc vrai, notamment au sujet de l'ordinateur de droite : mais c'est contradictoire. Donc l'ordinateur de droite est hors-service.

  Si l'appareil était prêt à décoller, alors tout ce que dit l'ordinateur de gauche serait vrai. Celui-ci fonctionnerait donc 5 sur 5, et tout ce que dirait l'ordinateur de droite (déréglé) serait vrai : c'est contradictoire.

  Autrement dit, l'appareil n°1 n'est pas paré à décoller.

- Pour l'appareil n°2 :

  Si l'ordinateur de gauche fonctionnait 5 sur 5, il ne déclarerait pas qu'il est déréglé (comme il le dit aussi de son voisin). L'ordinateur de gauche est donc déréglé, ce qui prouve que la phrase prononcée par l'ordinateur de droite est fausse. Celui-ci est donc également déréglé.

  Ainsi la deuxième partie de la phrase prononcée par l'ordinateur de gauche est vraie. Comme celui-ci est déréglé, c'est que la première partie de cette même phrase est fausse.

  Autrement dit, l'appareil n°2 n'est pas paré à décoller.

- Pour l'appareil n°3 :

  Si l'ordinateur de droite fonctionnait 5 sur 5, il dirait vrai en affirmant que l'ordinateur de gauche fonctionne lui aussi 5 sur 5 : la deuxième partie de la phrase prononcée par ce dernier serait donc fausse et il en résulterait une contradiction.

  Donc l'ordinateur de droite est déréglé, et la phrase qu'il prononce est fausse : l'ordinateur de gauche est déréglé. La deuxième partie de la phrase prononcée par ce dernier étant vraie, c'est que la première partie de cette même phrase est fausse. Autrement dit, l'appareil n°3 n'est pas paré à décoller.

- Pour l'appareil n°4 :

  Supposons que l'ordinateur de droite fonctionne 5 sur 5. D'après ses déclarations, on en déduit que l'ordinateur de gauche fonctionne aussi. Il en découle que l'ordinateur de droite est hors service : c'est contradictoire.

  Ainsi l'ordinateur de droite est déréglé.

  Dans ces conditions, supposons que l'ordinateur de gauche fonctionne 5 sur 5. Alors l'appareil est interdit de vol, et tout ce que déclare l'ordinateur de droite est exact : c'est absurde pour un ordinateur déréglé.

  L'ordinateur de gauche est donc lui aussi déréglé. Or la deuxième partie de sa déclaration est vraie. C'est que la première partie est fausse. Autrement dit, l'appareil n°4 est paré à décoller.

- Pour l'appareil n°5 :

  L'ordinateur de droite est déréglé, sinon il en résultait une contradiction manifeste avec la deuxième partie de la phrase qu'il prononce.

  Si l'ordinateur de gauche fonctionne 5 sur 5, alors l'apparail n°5 n'est pas paré à décoller.

  Supposons au contraire que l'ordinateur de gauche soit déréglé. Alors la deuxième partie de la phrase prononcée par l'ordinateur de droite est vraie. C'est nécessairement que la première partie est fausse.

  Autrement dit, l'appareil n°5 n'est pas paré à décoller.

  Remarque : contrairement aux questions précédentes, on voit qu'on peut conclure sans pour autant connaître l'état de fonctionnement d'un des deux ordinateurs. Celui de gauche peut fonctionner ou être déréglé : dans les deux cas, on trouve la réponse à la question posée.

- Pour l'appareil n°6 :

  L'ordinateur de gauche est déréglé, sinon il en résulterait une contradiction manifeste avec la deuxième partie de la phrase qu'il prononce.

  Supposons que l'appareil soit paré à décoller. Alors nécessairement la deuxième partie de ce que dit l'ordinateur de gauche est fausse. Il en résulte que l'ordinateur de droite fonctionne normalement, ce qui implique que l'appareil est interdit de vol : c'est contradictoire.

  Autrement dit, l'appareil n°6 n'est pas paré à décoller.

Question 2

Chaque ordinateur émet une assertion $\mathcal{A}$, consistant en un ou deux renseignements (séparés alors par un *et* logique, par exemple : *l'appareil n°1 est paré à décoller* **et** *l'ordinateur connexe est hors-service.*)

On nous dit qu'un ordinateur déréglé donne au moins un renseignement faux. Cela signifie tout simplement que la proposition $\mathcal{A}$ équivaut logiquement à la proposition $\mathcal{O}$: « l'ordinateur fonctionne 5 sur 5» .

– Pour l'appareil n°1 :

La phrase prononcée par l'ordinateur de gauche peut se traduire par : $\mathcal{P}$ et $\overline{\mathcal{D}}$.

On a donc l'équivalence $\mathcal{G} \Leftrightarrow (\mathcal{P}$ et $\overline{\mathcal{D}})$.

La phrase prononcée par l'ordinateur de droite se traduit par : $\mathcal{P}$ et $\mathcal{G}$.

On a donc l'équivalence $\mathcal{D} \Leftrightarrow (\mathcal{P}$ et $\mathcal{G})$, et on doit résoudre le système (S) $\begin{cases} \mathcal{G} \Leftrightarrow (\mathcal{P} \text{ et } \overline{\mathcal{D}}) \\ \mathcal{D} \Leftrightarrow (\mathcal{P} \text{ et } \mathcal{G}) \end{cases}$

On a successivement :

$$(S) \Leftrightarrow \begin{cases} \mathcal{G} \Leftrightarrow (\mathcal{P} \text{ et } (\overline{\mathcal{P}} \text{ ou } \overline{\mathcal{G}})) \\ \mathcal{D} \Leftrightarrow (\mathcal{P} \text{ et } \mathcal{G}) \end{cases} \Leftrightarrow \begin{cases} \mathcal{G} \Leftrightarrow ((\mathcal{P} \text{ et } \overline{\mathcal{P}}) \text{ ou } (\mathcal{P} \text{ et } \overline{\mathcal{G}})) \\ \mathcal{D} \Leftrightarrow (\mathcal{P} \text{ et } \mathcal{G}) \end{cases} \Leftrightarrow \begin{cases} \mathcal{G} \Leftrightarrow (\mathcal{P} \text{ et } \overline{\mathcal{G}}) \\ \mathcal{D} \Leftrightarrow (\mathcal{P} \text{ et } \mathcal{G}) \end{cases}$$

L'équivalence « $\mathcal{G} \Leftrightarrow (\mathcal{P}$ et $\overline{\mathcal{G}})$ » n'est possible que si la proposition $\mathcal{P}$ est fausse.

Cela signifie que l'appareil n°1 n'est pas paré à décoller.

– Pour l'appareil n°2 :

De ce que dit l'ordinateur de droite on tire l'équivalence : $\mathcal{D} \Leftrightarrow \mathcal{G}$.

De ce que dit l'ordinateur de gauche on tire : $\mathcal{G} \Leftrightarrow (\mathcal{P}$ et $\overline{\mathcal{G}}$ et $\overline{\mathcal{D}})$, donc : $\mathcal{G} \Leftrightarrow (\mathcal{P}$ et $\overline{\mathcal{G}})$.

Cela n'est possible que si $\mathcal{P}$ est fausse : l'appareil n°2 n'est pas paré à décoller.

– Pour l'appareil n°3 :

De ce que dit l'ordinateur de droite on tire l'équivalence : $\mathcal{D} \Leftrightarrow \mathcal{G}$.

De ce que dit l'ordinateur de gauche on tire : $\mathcal{G} \Leftrightarrow (\mathcal{P}$ et $\overline{(\mathcal{D} \text{ et } \mathcal{G})})$, donc : $\mathcal{G} \Leftrightarrow (\mathcal{P}$ et $\overline{\mathcal{G}})$.

Cela n'est possible que si $\mathcal{P}$ est fausse : l'appareil n°3 n'est pas paré à décoller.

– Pour l'appareil n°4 :

De ce que dit l'ordinateur de gauche on tire l'équivalence : $\mathcal{G} \Leftrightarrow (\overline{\mathcal{P}}$ et $\overline{\mathcal{D}})$.

De ce que dit l'ordinateur de droite on tire : $\mathcal{D} \Leftrightarrow (\overline{\mathcal{P}}$ et $\mathcal{G})$.

Si $\mathcal{P}$ était fausse, on aurait $\begin{cases} \mathcal{G} \Leftrightarrow \overline{\mathcal{D}} \\ \mathcal{D} \Leftrightarrow \mathcal{G} \end{cases}$, ce qui est absurde.

On en déduit que $\mathcal{P}$ est vraie : l'appareil n°4 est paré à décoller.

– Pour l'appareil n°5 :

De ce que dit l'ordinateur de droite on tire l'équivalence : $\mathcal{D} \Leftrightarrow (\mathcal{P}$ et $\overline{\mathcal{D}}$ et $\overline{\mathcal{G}})$.

Cela implique nécessairement que la proposition $\mathcal{D}$ est fausse (par l'absurde.)

Il en résulte que la proposition $(\mathcal{P}$ et $\overline{\mathcal{G}})$ est fausse.

De ce que dit l'ordinateur de gauche on tire l'équivalence : $\mathcal{G} \Leftrightarrow (\overline{\mathcal{P}}$ et $(\mathcal{G}$ ou $\mathcal{D}))$.

Puisque $\mathcal{D}$ est fausse, cela se simplifie en : $\mathcal{G} \Leftrightarrow (\overline{\mathcal{P}}$ et $\mathcal{G})$.

Si $\mathcal{P}$ était vraie, $\mathcal{G}$ serait donc fausse, et $(\mathcal{P}$ et $\overline{\mathcal{G}})$ serait vraie (et ce serait contradictoire.)

On en déduit que $\mathcal{P}$ est fausse : l'appareil n°5 n'est pas paré à décoller.

– Pour l'appareil n°6 :

De ce que dit l'ordinateur de gauche on tire l'équivalence : $\mathcal{G} \Leftrightarrow (\mathcal{P}$ et $\overline{\mathcal{G}}$ et $\overline{\mathcal{D}})$.

Cela implique nécessairement que la proposition $\mathcal{G}$ est fausse (par l'absurde.)

Il en résulte que la proposition $(\mathcal{P}$ et $\overline{\mathcal{D}})$ est fausse.

De ce que dit l'ordinateur de droite on tire l'équivalence : $\mathcal{D} \Leftrightarrow (\overline{\mathcal{P}}$ et $(\overline{\mathcal{D}}$ ou $\overline{\mathcal{G}}))$

Puisque $\mathcal{G}$ est fausse, cela se simplifie en : $\mathcal{D} \Leftrightarrow \overline{\mathcal{P}}$.

Il en résulte que $(\mathcal{P}$ et $\overline{\mathcal{D}})$ est équivalente à $\mathcal{P}$.

Cette dernière proposition est donc fausse : l'appareil n°6 n'est pas paré à décoller.

Question 3

1. – La variable associée à $\overline{\mathcal{A}}$ est bien sûr $1-a$.
   – Celle qui est associée à ($\mathcal{A}$ et $\mathcal{B}$) est ab (elle vaut 1 si et seulement si a et b valent 1 toutes les deux.)
   – On sait que ($\mathcal{A}$ ou $\mathcal{B}$) est équivalente à : $\overline{\overline{\mathcal{A}} \text{ et } \overline{\mathcal{B}}}$.
   La variable associée à ($\mathcal{A}$ ou $\mathcal{B}$) est donc : $1-(1-a)(1-b) = a+b-ab$.
   – On sait que ($\mathcal{A} \Rightarrow \mathcal{B}$) est équivalente à ($\overline{\mathcal{A}}$ ou $\mathcal{B}$).
   La variable associée à ($\mathcal{A} \Rightarrow \mathcal{B}$) est donc : $(1-a)+b-(1-a)b = 1-a+ab$.
   – La proposition ($\mathcal{A} \Leftrightarrow \mathcal{B}$) est équivalente à : ($\mathcal{A} \Rightarrow \mathcal{B}$) et ($\mathcal{B} \Rightarrow \mathcal{A}$).
   La variable associée à ($\mathcal{A} \Leftrightarrow \mathcal{B}$) est donc : $c = (1-a+ab)(1-b+ab)$.
   Sachant que $a^2 = a$ et $b^2 = b$, on trouve $c = (1-a)(1-b)+ab = 1-a-b+2ab$.
   On peut aussi écrire $c = 1-a^2-b^2+2ab = 1-(a-b)^2$.
   Il en résulte que $c = 1$ si et seulement si $a = b$, ce qui est rassurant.

2. Nous allons traduire par un système (S) les déclarations des ordinateurs de chaque appareil.
   On note p la variable associée à la proposition « *l'appareil est prêt à décoller* ».
   On note g (resp. d) les variables associées au bon fonctionnement de chaque ordinateur.
   – Pour l'appareil n°1 :
   Le système des deux équivalences (S) $\begin{cases} \mathcal{G} \Leftrightarrow (\mathcal{P} \text{ et } \overline{\mathcal{D}}) \\ \mathcal{D} \Leftrightarrow (\mathcal{P} \text{ et } \mathcal{G}) \end{cases}$ s'écrit maintenant (Σ) : $\begin{cases} g = p(1-d) \\ d = pg \end{cases}$
   Ce sytème implique $d = p^2(1-d) = p(1-d)$, donc $p = 0$ (car $p = 1$ conduirait à $d = 1-d$.)
   Cela signifie que l'appareil n°1 n'est pas paré à décoller.
   – Pour l'appareil n°2 :
   On a $d = g$ et $g = p(1-d)(1-g)$, donc $g = p(1-g)^2 = p(1-g)$.
   Là encore, il est nécessaire que p soit nul : l'appareil n°2 n'est pas paré à décoller.
   – Pour l'appareil n°3 :
   On a $d = g$ et $g = p(1-dg)$ donc $g = p(1-g^2) = p(1-g)$.
   La conclusion est la même : l'appareil n°3 n'est pas paré à décoller.
   – Pour l'appareil n°4 :
   On a $g = (1-p)(1-d)$ et $d = (1-p)g$. On en déduit l'égalité $d = (1-p)(1-d)$.
   Celle-ci implique $p = 1$ (car $p = 0$ conduirait à $d = 1-d$.)
   Conclusion : l'appareil n°4 est paré à décoller.
   – Pour l'appareil n°5 :
   De ce que dit l'ordinateur de droite on tire : $d = p(1-d)(1-g)$, qui exige $d = 0$.
   Il en découle $0 = p(1-g)$ donc $p = 0$ ou $g = 1$.
   De ce que dit l'ordinateur de gauche on tire : $g = (1-p)(g+d-gd)$.
   Puisque $d = 0$, cela s'écrit $g = (1-p)g$, donc $pg = 0$.
   Les deux égalités $p(1-g) = 0$ et $pg = 0$ exigent $p = 0$.
   Conclusion : l'appareil n°5 n'est pas paré à décoller.
   – Pour l'appareil n°6 :
   De ce que dit l'ordinateur de gauche on tire : $g = p(1-d)(1-g)$, qui implique $g = 0$.
   De ce que dit l'ordinateur de droite on tire : $d = (1-p)(1-dg)$ donc $d = 1-p$.
   Dans ces conditions $g = p(1-d)(1-g)$ donne $0 = p(1-d)$ puis $0 = p^2 = p$.
   Conclusion : l'appareil n°6 n'est pas paré à décoller.

Question 4

Première méthode :
Pour résoudre cette question, on va utiliser un formalisme ensembliste. Celui-ci peut sembler un peu lourd au départ, mais c'est un investissement rentable et il devient rapidement très efficace.
Notons $\mathcal{W} = (Lu, Ma, Me, Je, Ve, Sa, Di)$ le 7-uplet des jours de la semaine.
Notons $\mathcal{M} = \{Ve, Sa, Di\}$ l'ensemble des jours où les modernes mentent.
Notons $\mathcal{A} = \{Ma, Me, Je\}$ l'ensemble des jours où les anciens mentent.
Si $\mathcal{X}$ est un sous-ensemble de jours de la semaine, on notera $\mathcal{X}^+$ (resp. $\mathcal{X}^-$) l'ensemble obtenu par permutation circulaire d'une journée vers le futur (resp. vers le passé).
Si par exemple $\mathcal{X} = \{Lu, Sa, Di\}$, alors $\begin{cases} \mathcal{X}^+ = \{Ma, Di, Lu\} \\ \mathcal{X}^- = \{Di, Ve, Sa\} \end{cases}$ et $\begin{cases} \mathcal{X}^{++} = \{Me, Lu, Ma\} \\ \mathcal{X}^{3-} = X^{4+} = \{Ve, Me, Je\} \end{cases}$
Pour une étape donnée, notons J le jour de la semaine, J^+ le lendemain, J^- la veille, etc.
Chaque moderne prononce une phrase qui (fausse ou vraie) décrit un sous-ensemble $\mathcal{X}$ des jours de la semaine.
De deux choses l'une : ou bien le moderne ment (autrement dit $J \in \mathcal{M}$) et alors J n'appartient pas à $\mathcal{X}$, ou bien il dit la vérité ($J \notin \mathcal{M}$) et alors J appartient à $\mathcal{X}$.
De ce que prétend ce moderne, on conclut donc que J appartient à l'ensemble $(\mathcal{M} \cap \overline{\mathcal{X}}) \cup (\overline{\mathcal{M}} \cap \mathcal{X}) = \mathcal{M}\Delta\mathcal{X}$ (c'est la définition de la *différence symétrique* Δ.)
De même, si la phrase prononcée par un ancien (qu'elle soit vraie ou fausse) décrit un sous-ensemble $\mathcal{X}$ de la semaine, alors de ce qu'il prétend on conclut que J appartient à $\mathcal{A}\Delta\mathcal{X}$.

– Le jour du départ :

  Le moderne prétend que J^- est dans $\mathcal{M}$, c'est-à-dire que J est dans $\mathcal{M}^+ = \{Sa, Di, Lu\}$.
  On en tire que J est dans $\mathcal{M}\Delta\mathcal{M}^+ = \{Ve, Lu\}$.
  L'ancien prétend que J^- est dans $\mathcal{A}$ donc que J est dans $\mathcal{A}^+ = \{Me, Je, Ve\}$.
  On en tire que J est dans $\mathcal{A}\Delta\mathcal{A}^+ = \{Ma, Ve\}$.
  Finalement J est dans $\{Ve, Lu\} \cap \{Ma, Ve\} = \{Ve\}$.
  Conclusion : le jour du départ est un vendredi.

– Première étape :

  De ce que prétend le premier ancien, on tire que J est dans $\{Ma, Ve\}$ (question précédente.)
  Le second ancien prétend J^{3+} est dans $\mathcal{A}$, donc J est dans $\mathcal{A}^{3-} = \{Sa, Di, Lu\}$.
  On en tire que J est dans $\mathcal{A}\Delta\{Sa, Di, Lu\} = \{Ma, Me, Je, Sa, Di, Lu\}$.
  On en déduit que J est dans $\{Ma, Ve\} \cap \{Ma, Me, Je, Sa, Di, Lu\} = \{Ma\}$.
  Conclusion : le jour de la première étape est un mardi.

– Deuxième étape :

  De ce que prétend le moderne, on tire que J est dans $\{Ve, Lu\}$ (voir «jour du départ»)
  L'ancien prétend que J est dans l'ensemble $\mathcal{M}$.
  On en tire que J est dans $\mathcal{A}\Delta\mathcal{M} = \{Ma, Me, Je, Ve, Sa, Di\}$.
  Finalement, J est dans $\{Ve, Lu\} \cap \{Ma, Me, Je, Ve, Sa, Di\} = \{Ve\}$.
  Conclusion : le jour de la seconde étape est un vendredi.

– A l'arrivée :

  Le moderne prétend que J est dans $\mathcal{M}^- \cap \mathcal{M}^+ = \{Sa\}$.
  On en tire que J est dans $\mathcal{M}\Delta\{Sa\} = \{Ve, Di\}$.
  L'ancien prétend que J est dans $\mathcal{X} = \{Lu, Ma, Me, Je, Sa, Di\}$.
  On en tire que J est dans $\mathcal{A}\Delta X = \{Lu, Sa, Di\}$.
  Finalement, J est dans $\{Ve, Di\} \cap \{Lu, Sa, Di\} = \{Di\}$.
  Conclusion : le jour d'arrivée est un dimanche.

Deuxième méthode :
On note cette fois $di, lu, ma, me, je, ve, sa$ des variables booléennes (valant donc 0 ou 1) caractérisant chacune un jour de la semaine : par exemple $di = 1$ si on est un dimanche et $di = 0$ sinon.

Pour chaque «variable journalière» j, on a $j^2 = j$. Bien sûr : $di + lu + ma + me + je + ve + sa = 1$.

Considérons la déclaration $\mathcal{P}$ d'un moderne par exemple, décrivant (sans se préoccuper de savoir s'il ment ou dit la vérité) un ensemble $\mathcal{X}$ de jours de la semaine, et soit p la variable booléenne associée à cette proposition. Il est clair que p est la somme des variables journalières associées aux différents jours possibles de $\mathcal{X}$.

Considérons ensuite la proposition $\mathcal{V}_m$: «*on est un jour de vérité pour les modernes*».
La variable booléenne associée est $v_m = lu + ma + me + je$.

Avec ces notations, $\mathcal{P}$ et $\mathcal{V}_m$ sont équivalentes. On a donc l'égalité $p = v_m = lu + ma + me + je$.

De même, la proposition $\mathcal{Q}$ (de variable booléenne q) prononcée par un ancien est équivalente à la proposition de variable booléenne $v_a = ve + sa + di + lu$ (qui caractérise les jours de vérité des anciens.) De ce que dit un ancien, on tirera donc une égalité du type $q = ve + sa + di + lu$.

– Le jour du départ :
  De ce que dit le moderne on tire : $lu + ma + me + je = sa + di + lu$, donc $ma + me + je = sa + di$.
  Cette égalité implique nécessairement $ma = me = je = sa = di = 0$.
  De ce que dit l'ancien on tire : $ve + sa + di + lu = me + je + ve$ donc $sa = di = lu = me = je = 0$.
  Il ne reste que $ve = 1$: on est un vendredi.

– Première étape :
  Le premier ancien nous apprend que : $sa = di = lu = me = je = 0$ (voir ci-dessus).
  Le second nous apprend que $ve + sa + di + lu = sa + di + lu$ donc $ve = 0$.
  Il ne reste que $ma = 1$: on est un mardi.

– Deuxième étape :
  Le moderne nous apprend que $lu + ma + me + je = sa + di + lu$, donc $ma = me = je = sa = di = 0$.
  L'ancien nous apprend que $ve + sa + di + lu = ve + sa + di$ donc $lu = 0$.
  Il ne reste que $ve = 1$: on est un vendredi.

– A l'arrivée :
  Le moderne nous apprend que $lu + ma + me + je = sa$, donc $lu = ma = me = je = sa = 0$.
  L'ancien nous apprend que $ve + sa + di + lu = lu + ma + me + je + sa + di$ donc $ve = ma = me = je = 0$.
  Il ne reste que $di = 1$: on est un dimanche.

Remarque : cette méthode se prête bien à l'utilisation de Maple (voir la feuille Maple en annexe.)

Question 5

On va utiliser ici des variables booléennes. Pour chaque braquage, soient $\mathcal{A}, \mathcal{B}, \mathcal{C}$ les propositions énonçant respectivement la culpabilité d'Alfred, Baptiste, Charly, et soient a, b, c les variables booléennes associées.

On va utiliser la question (3.1). Chaque renseignement $\mathcal{R}$ étant vrai, sa variable booléenne vaut 1, et il suffit de l'écrire en fonction de a, b, c pour obtenir une égalité. Le système obtenu donne alors le(s) coupable(s).

Puisqu'il y a au moins un coupable, on a toujours $a + b + c \geq 1$.

– Pour le hold-up de la bijouterie de Gemma :

  Le premier renseignement s'écrit «$\overline{\mathcal{C}} \Rightarrow \mathcal{A}$», ou encore «$\mathcal{C}$ ou $\mathcal{A}$», c'est-à-dire : $1 = a + (1 - a)c$.
  Le second s'écrit «$\mathcal{A} \Rightarrow \overline{(\mathcal{B} \Leftrightarrow \mathcal{C})}$», ou encore «$\overline{\mathcal{A}}$ ou $\overline{(\mathcal{B} \Leftrightarrow \mathcal{C})}$».
  Mais la variable associée à $\overline{(\mathcal{B} \Leftrightarrow \mathcal{C})}$ est $b + c - 2bc$: c'est le *ou exclusif* des propositions $\mathcal{B}, \mathcal{C}$.
  On peut donc écrire : $1 = (1 - a) + a(b + c - 2bc)$, c'est-à-dire : $a(b + c - 2bc) = a$.
  Le troisième renseignement s'écrit «$\overline{\mathcal{B}} \Rightarrow \overline{\mathcal{C}}$» ou encore «$\mathcal{C} \Rightarrow \mathcal{B}$», c'est-à-dire : $c = bc$.
  On interprète le quatrième en disant que si Alfred est innocent, alors il y a un coupable et un seul parmi Baptiste et Charly. Autrement dit : «$\overline{\mathcal{A}} \Rightarrow \overline{(\mathcal{B} \Leftrightarrow \mathcal{C})}$», c'est-à-dire «$\mathcal{A}$ ou $\overline{(\mathcal{B} \Leftrightarrow \mathcal{C})}$».
  Joint au deuxième renseignement, cela donne «$\overline{(\mathcal{B} \Leftrightarrow \mathcal{C})}$», c'est-à-dire $b + c - 2bc = 1$.
  Finalement, on aboutit à : $a + b + c \geq 1$, $a + c - ac = 1$, $c = bc$ et $b + c - 2bc = 1$.
  Puisque $c = bc$, l'égalité $b + c - 2bc = 1$ implique $b - c = 1$ donc $b = 1$ et $c = 0$.
  Enfin $a + c - ac = 1$ donne $a = 1$. Les coupables sont Alfred et Charly.

– Pour le casse de la banque de Prokur :

Remarquons (on l'a utilisé dans la question précédente) que si X, Y sont deux propositions de variables booléennes x, y, alors dire que l'implication « $X \Rightarrow Y$ » est vraie c'est dire qu'on a l'égalité $x(1-y)=0$.

Le premier renseignement donne « $\mathcal{B} \Rightarrow \mathcal{C}$ », donc : $b(1-c)=0$.
Le deuxième donne « $\mathcal{A} \Rightarrow \overline{\mathcal{C}}$ » donc $ac=0$.
Le troisième donne « $(\mathcal{A}$ et $\overline{\mathcal{B}}) \Rightarrow \mathcal{C}$ » donc $a(1-b)(1-c)=0$.
Le quatrième donne « $\mathcal{C} \Rightarrow (\mathcal{A}$ ou $\mathcal{B})$ » donc $c(1-a-b+ab)=0$.

Compte tenu de $ac=0$, tout cela se réduit à : $b(1-c)=0$, $ac=0$, $a(1-b)=0$, et $c(1-b)=0$.
L'égalité $a=1$ impliquerait $c=0$ et $b=1$, en contradiction avec $b(1-c)=0$.
On a donc $a=0$, puis $b=c$. Sachant qu'il y a au moins un coupable, on obtient : $b=c=1$.
Conclusion : dans cette affaire, les coupables sont Baptiste et Charly.

– Pour le vol dans la pharmacie de Narcozia :

Le premier renseignement donne « $\mathcal{A} \Rightarrow \overline{\mathcal{B}}$ », donc : $ab=0$.
Le deuxième donne « $\mathcal{B} \Rightarrow \overline{(\mathcal{A} \Leftrightarrow \mathcal{C})}$ », donc $b(1-a-c+2ac)=0$.
Le troisième donne « $\mathcal{C} \Rightarrow (\mathcal{A}$ et $\mathcal{B})$ », donc $c(1-ab)=0$.

Compte tenu de $ab=0$, la deuxième égalité se simplifie en $b(1-c)=0$.
Si on avait $a=0$, il en résulterait $c=0$ (troisième égalité) puis $b=0$.
Comme il y a au moins un coupable, $a=1$, puis $b=0$ et $c=0$.
Conclusion : dans cette affaire, le seul coupable est Alfred.

Question 6

Considérons un individu A, et notons $\mathcal{P}$ la proposition qu'il énonce.

Soit a la variable valant 1 si A est un menteur, et 0 sinon.
Soit p la variable associée à $\mathcal{P}$ ($p=1$ si $\mathcal{P}$ est vraie, $p=0$ sinon.)

Il est certain que ou bien « A est menteur et $\mathcal{P}$ est fausse », ou bien « A est changeant » (et dans ce dernier cas on ne peut rien dire de la proposition $\mathcal{P}$.)

Cela se traduit par l'égalité : $1=a(1-p)+(1-a)$, c'est-à-dire $ap=0$.

Notons x (resp. y, z, t) les variables associées aux propositions « X (resp Y, Z, T) est un menteur ».

– La proposition énoncée par X a pour variable associée $(1-y)+(1-t)-2(1-y)(1-t)=y+t-2yt$.
De ce que dit X on en tire donc $x(y+t-2yt)=0$.
– La proposition énoncée par Y a pour variable associée $1-y+y(1-x)=1-xy$.
De ce que dit X on en tire donc $y(1-xy)=0$.
– La proposition énoncée par Z a pour variable associée $z(1-y)+(1-z)y-2z(1-y)(1-z)y=y+z-2yz$.
De ce que dit Z on en tire donc $z(y+z-2yz)=0$.
– Notons i la variable associée à la proposition « le problème était intéressant ».
La proposition énoncée par T a pour variable associée $1-i$.
De ce que dit T on en tire donc $t(1-i)=0$.

Finalement on doit résoudre le système : $x(y+t-2yt)=0$, $y(1-xy)=0$, $z(y+z-2yz)=0$, $t(1-i)=0$.

Si on avait $t=0$, la première égalité donnerait $xy=0$.
La deuxième donnerait alors $y=0$, et ensuite la troisième donnerait $z=0$.
Mais c'est impossible car il y a au moins deux menteurs ($x+y+z+t \geq 2$.)

On en déduit l'égalité $t=1$, puis $x(1-y)=0$, $y(1-xy)=0$, $z(y+z-2yz)=0$, $i=1$.
Là encore $x=0$ est exclu car il conduirait à $y=z=0$.

Finalement $t=1$, $x=1$, $y=1$, $z(1-z)=0$, et $i=1$.

Donc X, Y, T sont des menteurs (pour Z on ne sait pas), mais surtout :

Le problème était intéressant !

Utilisation de Maple

Position du problème

Les opérateurs not, and, or, xor, et implies permettent de construire des expressions de type logical.

Nous appelerons *expression logique formelle* toute expression de type logical ou equation, dont les arguments sont des noms ou les constantes booléennes true ou false.

Par exemple, on place une *expression logique formelle* dans la variable E :

```
> E:= (X or Y) implies (Z and T);
```

$$E := X \textbf{ or } Y \textbf{ implies } Z \textbf{ and } T$$

Maple effectue certaines simplifications ou conversions automatiques :

```
> X or false, Y and false, Z xor true, false implies T, U implies
> true;
```

$$X,\ \mathit{false},\ \textbf{not } Z,\ \mathit{true},\ \mathit{true}$$

Voici quatre autres possibilités. Les deux premières simplifications sont prévisibles. En revanche, on peut s'étonner que les deux autres expressions ne se simplifient pas.

En effet, « Z xor Z » devrait donner false, et « T implies T » devrait donner true.

```
> X or X, Y and Y, Z xor Z, T implies T;
```

$$X,\ Y,\ Z \textbf{ xor } Z,\ T \textbf{ implies } T$$

De même, l'expression suivante devrait se simplifier en *true*. L'utilisation de simplify ne donne rien.

```
> X or not(X); simplify(%);
```

$$X \textbf{ and or}(X)$$
$$X \textbf{ or not}(X)$$

L'opérateur = permet quant à lui de former des expressions de type equation.

On considérera ici l'opérateur d'égalité comme synonyme de l'équivalence.

Par exemple, $X = Y$ est une façon plus simple de noter (X implies Y) and (Y implies X).

On peut alors former des *systèmes d'expressions logiques formelles* (en abrégé : *self*).

En voici un exemple :

```
> sys:={X = (Y implies Z), Y = (not(X) and Z), Z = (Y implies
> not(X))};
```

$$sys := \{X = (Y \textbf{ implies } Z),\ Y = (\textbf{ not } X \textbf{ and } Z),\ Z = (Y \textbf{ implies not } X)\}$$

Le système précédent est la traduction de l'énigme logique suivante :

Proposition X : « si la proposition Y est vraie, alors la proposition Z est vraie »

Proposition Y : « la proposition X est fausse, et la proposition Z est vraie »

Proposition Z : « si la proposition Y est vraie, alors la proposition Y est fausse »

Il n'est pas évident de deviner que la seule solution est ici « X vraie », « Y fausse », « Z vraie ».

Nous souhaitons programmer la résolution d'un tel système.

Par exemple, nous aimerions pouvoir obtenir, avec l'exemple précédent :

```
> bsolve(sys);
```

$$\{Z = \mathit{true},\ Y = \mathit{false},\ X = \mathit{true}\}$$

Une approche algébrique

L'idée consiste à associer à chaque proposition logique X une variable booléenne x. Cette variable x prendrait la valeur 1 quand X est vraie et la valeur 0 quand X est fausse.

Or les opérations logiques sur les propositions se traduisent en opérations algébriques sur ces variables.
La résolution d'un *self* conduira donc à celle d'un système d'équations algébriques.

« Algébrisation » d'une expression logique

Supposons que x et y soient les variables associées aux propositions X et Y.
Alors la variable associée à not(X) est 1-x. De même celle associée à « X and Y » est xy.
On sait que « X or Y » est équivalent à not(not(X) and not(Y)).
La variable associée à « X or Y » est donc 1-(1-x)(1-y) = x+y-xy.
« X implies Y » est équivalent à « not(X) or Y ». Sa variable est donc (1-x)+y-(1-x)y = 1-x+xy.
« A xor B » équivaut à « (A et non(B)) ou (B et non (A)) ».
La variable associée à « A xor B » est donc : a(1-b)+b(1-a)-a(1-b)b(1-a) = a(1-b)+b(1-a) = a+b-2ab.
Enfin la variable associée à « $A = B$ » est : (1-a+ab)(1-b+ab) = (1-a)(1-b)+ab = 1-a-b+2ab.
On peut remarquer que $1 - a - b + 2ab = 1 - (a - b)^2$. Dans ces conditions dire que l'équivalence « $A = B$ » est vraie, c'est dire $a = b$, ce qui est rassurant.

Idée générale du programme

On rappelle qu'une expression est codée sous une forme arborescente. Elle possède en général un opérateur racine, qui possède lui-même des arguments. Ceux-ci sont souvent des expressions, avec opérateurs racines et arguments. Cette structure arborescente se termine par des feuilles qui dans notre cas sont des noms ou des contantes (true ou false).

Prenons par exemple le cas de l'expression E suivante :

```
> E:=(not(X xor Y)) implies Y;
```

$$E := \textbf{not}\ (X\ \textbf{xor}\ Y)\ \textbf{implies}\ Y$$

La racine r de E est l'opérateur implies, dont les deux arguments sont g et d, avec :

```
> r:=op(0,E); g:=op(1,E); d:=op(2,E);
```

$$r := implies$$
$$g := \textbf{not}\ (X\ \textbf{xor}\ Y)$$
$$d := Y$$

L'argument d est une feuille de l'expression initiale, puisqu'il se réduit au nom Y. La racine gr de g est l'opérateur not, dont le seul argument est gg, avec :

```
> gr:=op(0,g); gg:=op(1,g);
```

$$gr := not$$
$$gg := X\ \textbf{xor}\ Y$$

La racine ggr de gg est l'opérateur xor, dont les arguments sont les feuilles ggg=X et ggd=Y.
Notons x, y, e les variables booléennes associées aux « propositions » X, Y, E.
Alors la variable associée à « X xor Y » est x+y-2xy.
Celle qui est associée à « not(X xor Y) » est donc t = 1-x-y+2xy.
Enfin celle associée à E est donc e = 1-t+ty = x+y-2xy+(1-x-y+2 xy)y.
Bien sûr, on a $y^2 = y$, donc e = x+y-2xy+y-xy-y+2xy = x+y-xy.
Finalement, on observe que E est en fait équivalent à « X or Y ».

Pour algébriser une expression logique ayant un opérateur racine r ayant lui-même deux arguments G et D, il suffit d'algébriser ces deux arguments (de manière à obtenir des expressions algébriques g et d) et à combiner entre elles ces deux expressions en fonction de l'opérateur racine r (c'est clair ?) Une approche récursive s'impose d'elle-même. Pour stopper la « plongée » récursive, il suffit de se dire les choses suivantes :

a) Les feuilles true et false sont converties respectivement en 1 et en 0.

b) Les feuilles qui se réduisent à des noms sont inchangées.

On traitera à part le cas de not, qui est ici le seul opérateur *unaire*.

Quelques simplifications utiles

Il est important de noter que le traitement des opérateurs binaires conduit à effectuer des produits de variables booléennes. Il est possible notamment qu'apparaissent des carrés. Or pour tout variable booléenne v, on a l'égalité $v^2 = v$. Il convient donc d'effectuer toutes les simplifications des carrés au fur et à mesure de leur apparition. Or pour cela, il faut connaître tous les noms (indéterminées) qui figurent dans une expression. L'instruction indets est faite pour ça :

```
> E:=(1-x*y)*(z+y-x)+(x+z*y)*(x-y); n:=indets(E);
```

$$E := (1 - x\,y)\,(z + y - x) + (x + z\,y)\,(x - y)$$

$$n := \{x,\, y,\, z\}$$

En reprenant l'exemple ci-dessus, voici comment former les égalités qui expriment que x, y, z sont des variables booléennes.

```
> n:=map(t->(t^2=t),n);
```

$$n := \{x^2 = x,\, y^2 = y,\, z^2 = z\}$$

Et voici maintenant la forme développée de E, avant et après simplification.

```
> E:=expand(E);
```

$$E := z + y - x - x\,y^2 + x^2\,y + x^2 - x\,y - z\,y^2$$

```
> E:=subs(n,E);
```

$$E := z + y - x\,y - z\,y$$

Un programme de conversion d'expression logique

Voici une procédure **bconv**, dont le rôle est de convertir une expression logique en une expression algébrique, en utilisant les idées exposées précédemment. Les noms A, B, etc... qui figurent dans cette expressions sont inchangés : tout se passe comme si une proposition A (censée prendre les valeurs false ou true) était transformée en une variable booléenne A (prenant la valeur 0 ou 1). On voit que l'expression attendue doit être un nom, une des deux constantes true ou false, une équation ou un expression logique (au sens donné par Maple, c'est-à-dire formée avec les opérateurs and, or, xor, implies et not.)

```
> bconv:=proc(E::{name,logical,equation,truefalse})
>   local r,g,d,p,s;
>   if type(E,truefalse) then
>     return(`if`(E,1,0)) # 1 pour true, et 0 pour false
>    fi;
>    if type(E,name) then return(E) fi; # noms inchangés
>    r:=op(0,E); # l'opérateur racine
>    g:=bconv(op(1,E)); # convertit opérande gauche (ou unique pour «not»)
>    if r=`not` then
>      return(1-g) # traitement d'une expression de type «not»
>    fi;
>    d:=bconv(op(2,E)); # convertit opérande droite
>    p:=expand(d*g);   # développe le produit des deux conversions.
>    s:=map(x->(x^2=x),indets(p)); # les équations du type A^2=A
>    p:=subs(s,p);     # simplifie les carrés de variables booléennes
>    if r=`and` then p        # traitement du « and »
>      elif r=`or` then d+g-p # traitement du « or »
>      elif r=`xor` then d+g-2*p # traitement du « xor »
>      elif r=`=` then 1-g-d+2*p # traitement de l'équivalence « = »
>      else 1-g+p      # seul cas restant: l'opérateur « implies »
>    fi;
> end:
```

Voici quelques exemples très très simples :

```
> bconv(true), bconv(false), bconv(X);
```

$$1,\ 0,\ X$$

Voici quelques exemple très simples :

```
> bconv(not(X)); bconv(X and Y); bconv(X or Y);
> bconv(X implies Y); bconv(X xor Y);
```

$$1 - X$$
$$Y\,X$$
$$Y + X - Y\,X$$
$$1 - X + Y\,X$$
$$Y + X - 2\,Y\,X$$

Voici un exemple simple :

```
> E:=(X and Y) or (Y and Z); bconv(E);
```

$$E := X \textbf{ and } Y \textbf{ or } Y \textbf{ and } Z$$
$$Z\,Y + Y\,X - Z\,Y\,X$$

Voici un exemple qui traduit la figure dite du *syllogisme*. Le résultat est 1, ce qui signifie que la proposition logique E est toujours vraie.

```
> E:=((X implies Y) and (Y implies Z)) implies (X implies Z);
> bconv(E);
```

$$E := (X \textbf{ implies } Y) \textbf{ and } (Y \textbf{ implies } Z) \textbf{ implies } (X \textbf{ implies } Z)$$
$$1$$

Voici un dernier exemple :

```
> E:=((Y xor Z) or (Z xor X) or (X xor Y)); bconv(E);
```

$$E := (Y \textbf{ xor } Z) \textbf{ or } (Z \textbf{ xor } X) \textbf{ or } (X \textbf{ xor } Y)$$
$$X + Z - Z\,X + Y - Z\,Y - Y\,X$$

Résolution de systèmes logiques formels

On a s'intéresse ici à des « self » (systèmes d'équations logiques formels.)

Chaque équation dans un self peut être une égalité reliant deux expressions logiques (auquel cas, comme on l'a vu, cette égalité est interprétée comme une équivalence), ou un expression logique X qui n'est pas une égalité. Dans ce dernier cas, on considère que cette expression logique X sous-entend l'égalité X = true.

Sur cet exemple, on voit que le programme **bconv** effectue déjà ce genre de sous-entendu.

```
> bconv(X), bconv(X=true);
```

$$X,\ X$$

On sait que le programme bconv traduit une expression logique en termes algébriques. Il faut par exemple imaginer que la proposition X est transformée en sa variable booléenne x (même si le nom X est conservé, son statut n'est plus le même). Mais si on considère que X est vraie, alors on doit considérer que la variable x vaut 1.

C'est pourquoi il faut transformer les résultats x, y, z de **bconv** en autant d'égalités $x = 1$, $y = 1$, $z = 1$, etc.

Voici par exemple la conversion de l' ensemble E en un ensemble d'égalités avec second membre égal à 1.

```
> E:={X, Y, Z}; E:=map(t->(t=1),E);
```

$$E := \{X,\ Y,\ Z\}$$
$$E := \{X = 1,\ Y = 1,\ Z = 1\}$$

Quand on résout un système dont les indéterminées représentent des variables booléennes x, y, z, il ne faut pas oublier de préciser les conditions $x^2 = x$, $y^2 = y$, $z^2 = z$, sans lesquelles la recherche des solutions pourrait se révéler impossible ou s'égarer dans des domaines de valeurs sans rapport avec la nature de x, y, z.

A partir du système E suivant, voici le système des conditions qu'il conviendra d'ajouter.

```
> E:={X*Y+X*Z-X-Y=1, X+Y*Z-Y=1, Z*(X+Y-X*Y)=1};
> map(t->(t^2=T),indets(E));
```

$$E := \{Y\,X + Z\,X - X - Y = 1,\ X + Z\,Y - Y = 1,\ Z\,(Y + X - Y\,X) = 1\}$$
$$\{X^2 = T,\ Y^2 = T,\ Z^2 = T\}$$

Voici maintenant le programme bsolve :

```
> bsolve:=proc(S::set)
>   local s,c;
>   # On converitit les équations du système.
>   s:=map(bconv,S);
>   # On convertit les résultats x en équations x=1
>   s:=map(x->(x-1),s);
>   # Voici les conditions disant qu'on a des variables booléennes.
>   c:=map(x->(x^2=x),indets(s));
>   # on ajoute ces conditions et on résout le système.
>   s:=solve({op(c),op(s)});
> end:
```

Voici par exemple la résolution du système logique formel E. Au départ E est la traduction de l'énigme suivante :

Proposition X : « si la proposition Y est vraie, alors la proposition Z est vraie »

Proposition Y : « la proposition X est fausse, et la proposition Z est vraie »

Proposition Z : « si la proposition Y est vraie, alors la proposition Y est fausse »

Question : pour chacune des propositions X, Y, Z, dire si elle est vraie ou fausse.

```
> sys:={X = (Y implies Z), Y = (not(X) and Z), Z = (Y implies
> not(X))};
> res:=bsolve(sys);
```

$$sys := \{X = (Y\ \textbf{implies}\ Z),\ Y = (\ \textbf{not}\ X\ \textbf{and}\ Z),\ Z = (Y\ \textbf{implies}\ \ \textbf{not}\ X)\}$$
$$res := \{Z = 1,\ Y = 0,\ X = 1\}$$

Application au problème «énigmes logiques»

On écrit les systèmes correspondant aux déclarations des deux ordinateurs de bord de chaque astronef.

Notons g la variable booléenne associée à la proposition « *l'ordinateur de gauche fonctionne 5 sur 5* ».

On définit de même la variable booléenne d pour l'ordinateur de droite.

Soit p la variable booléenne associée à la proposition « *l'astronef est paré à décoller* ».

Un ordinateur (par exemple celui de gauche) émet une proposition A. Cette proposition est vraie si l'ordinateur fonctionne 5 sur 5, et elle globalement fausse (car l'une au moins des deux affirmations qu'elle contient l'est) quand l'ordinateur est déréglé. Si on note a la variable booléenne associée à la proposition A, on a donc $g = a$.

Dans le premier astronef, l'ordinateur de gauche déclare que l'appareil est paré à décoller et que l'ordinateur de droite est hors-service. Ainsi on a l'égalité $g = (p \text{ and } \text{not}(d))$.

De même l'ordinateur de droite déclare que l'appareil est paré à décoller et que l'ordinateur de gauche fonctionne 5 sur 5. Cela se traduit par l'égalité $d = (p \text{ and } g)$.

Voici donc le système obtenu, et sa résolution grâce à **bsolve**. On constate que p est égal à 0. Donc l'appareil n'est pas prêt à décoller.

```
> sys1:={g = (p and not(d)), d = (p and g)};
> bsolve(sys1);
```

$$sys1 := \{g = (p \text{ and } \text{not } d), d = (p \text{ and } g)\}$$
$$\{d = 0, p = 0, g = 0\}$$

Pour la suite, on se reportera à l'énoncé du DM. L'appareil n°2 n'est pas prêt à décoller :

```
> sys2:={g =(p and not(g) and not(d)), d = g};
> bsolve(sys2);
```

$$sys2 := \{g = (p \text{ and } \text{not } g \text{ and } \text{not } d), d = g\}$$
$$\{g = 0, d = 0, p = 0\}$$

L'appareil n°3 n'est pas prêt à décoller :

```
> sys3:={g=(p and (not(g) or not(d))), d=g};
> bsolve(sys3);
```

$$sys3 := \{d = g, g = (p \text{ and } \text{not } (g \text{ and } d))\}$$
$$\{g = 0, d = 0, p = 0\}$$

On voit que l'appareil n°4 est prêt à décoller :

```
> sys4:={g=(not(p) and not(d)), d=(not(p) and g)};
> bsolve(sys4);
```

$$sys4 := \{g = \text{not } (p \text{ or } d), d = (\text{not } p \text{ and } g)\}$$
$$\{g = 0, d = 0, p = 1\}$$

L'appareil n°5 n'est pas prêt à décoller :

```
> sys5:={g=(not(p) and (g or d)), d=(p and (not(g or d)))};
> bsolve(sys5);
```

$$sys5 := \{d = (p \text{ and } \text{not } (g \text{ or } d)), g = (\text{not } p \text{ and } (g \text{ or } d))\}$$
$$\{g = 0, d = 0, p = 0\}, \{d = 0, p = 0, g = 1\}$$

L'appareil n°6 n'est pas prêt à décoller :

```
> sys6:={g=(p and not(g) and not(d)), d=(not(p) and (not(g) or
> not(d)))};
> bsolve(sys6);
```

$$sys6 := \{d = \text{not } (p \text{ or } g \text{ and } d), g = (p \text{ and } \text{not } g \text{ and } \text{not } d)\}$$
$$\{d = 1, g = 0, p = 0\}$$

Questions sur les anciens et les modernes (problème «énigmes logiques»)

La variable *Week* exprime qu'on est l'un (et l'un seulement !) des jours de la semaine.

```
> Week := Lu + Ma + Me + Je + Ve + Sa + Di=1;
```

$$Week := Lu + Ma + Me + Je + Ve + Sa + Di = 1$$

La variable *Cond* donne la séquence des conditions exprimant que les variables *Lu*, *Ma*, etc. sont booléennes, en ce sens qu'elles ne peuvent prendre que la valeur 0 ou 1.

```
> Cond := op(map(t->(t^2=t),indets(Week)));
```

$$Cond := Me^2 = Me, Sa^2 = Sa, Ma^2 = Ma, Ve^2 = Ve, Lu^2 = Lu, Je^2 = Je, Di^2 = Di$$

La variable booléenne *Vmod* indique si on est dans un jour de vérité pour un moderne (cette variable vaut 1 si et seulement si on est un lundi, un mardi, un mercredi ou un jeudi.)

La variable *Vanc* indique si on est dans un jour de vérité pour un ancien.

```
> Vmod := Lu + Ma + Me + Je;
> Vanc := Ve + Sa + Di + Lu;
```

$$Vmod := Lu + Ma + Me + Je$$
$$Vanc := Ve + Sa + Di + Lu$$

Le jour du départ. La variable *Pmod* exprime ce que dit le moderne (à savoir on est un samedi, un dimanche ou un lundi, c'est-à-dire le lendemain d'un jour de mensonge pour les modernes). La variable *Panc* exprime ce que dit l'ancien (à savoir on est un mercredi, un jeudi ou un vendredi, c'est-à-dire le lendemain d'un jour de mensonge pour les anciens).

```
> Pmod := Sa + Di + Lu;
> Panc := Me + Je + Ve;
```

$$Pmod := Sa + Di + Lu$$
$$Panc := Me + Je + Ve$$

La variable *Pmod* vaut 1 (càd représente une vérité) si et seulement si on est un jour de vérité pour les modernes, càd si et seulement si la variable *Vmod* vaut 1. On traduit celà par l'égalité $Vmod = Pmod$. On écrit de même l'égalité Vanc = Panc. Voici donc le système résolu par l'instruction **solve**. On voit qu'on est un vendredi.

```
> solve({ Vmod = Pmod, Vanc = Panc, Week, Cond});
```

$$\{Sa = 0,\ Lu = 0,\ Ve = 1,\ Di = 0,\ Me = 0,\ Je = 0,\ Ma = 0\}$$

Première étape (se reporter au DM). On est un mardi.

```
> Panc1 := Me + Je + Ve:
> Panc2 := Sa + Di + Lu:
> solve({ Vanc = Panc1, Vanc = Panc2, Week, Cond});
```

$$\{Sa = 0,\ Lu = 0,\ Di = 0,\ Ve = 0,\ Me = 0,\ Je = 0,\ Ma = 1\}$$

Deuxième étape. On est un vendredi.

```
> Pmod := Sa + Di + Lu:
> Panc := Ve + Sa + Di:
> solve({ Vmod = Pmod, Vanc = Panc, Week, Cond});
```

$$\{Ve = 1,\ Lu = 0,\ Di = 0,\ Je = 0,\ Sa = 0,\ Ma = 0,\ Me = 0\}$$

Le jour d'arrivée. On est dimanche.

```
> Pmod := Sa: Panc := 1-Ve:
> solve({ Vmod = Pmod, Vanc = Panc, Week, Cond});
```

$$\{Lu = 0,\ Je = 0,\ Ve = 0,\ Sa = 0,\ Ma = 0,\ Me = 0,\ Di = 1\}$$

Enquête sur différentes braquages (suite du problèmes «énigmes logiques»)

Hold-up de la bijouterie de Gemma. Les coupables sont Alfred et Charly.

```
> sys:={A or B or C,not(C) implies A, A implies (B xor C), C implies
> B, not(A) implies (B xor C)};
```

$$sys := \{A \textbf{ or } B \textbf{ or } C,\ \textbf{not } C \textbf{ implies } A,\ A \textbf{ implies } B \textbf{ xor } C,\ C \textbf{ implies } B,\ \textbf{not } A \textbf{ implies } B \textbf{ xor } C\}$$

```
> bsolve(sys);
```

$$\{C = 0,\ B = 1,\ A = 1\}$$

Casse de la banque de Prokur. Les coupables sont Baptiste et Charly.


```
> sys:={A or B or C, B implies C, A implies not(C), (A and not(B))
> implies C, C implies (A or B)};
```

$$sys := \{A \textbf{ implies } \textbf{ not } C,\ C \textbf{ implies } A \textbf{ or } B,\ A \textbf{ and } \textbf{ not } B \textbf{ implies } C,\ B \textbf{ implies } C,\ A \textbf{ or } B \textbf{ or } C\}$$

```
> bsolve(sys);
```

$$\{C = 1,\ A = 0,\ B = 1\}$$

Vol dans la pharmacie de Narcozia. Il y a un seul coupable, c'est Alfred.

```
> sys:={A or B or C, A implies not(B), B implies (A xor C), C implies
> (A and B)};
```

$$sys := \{A \textbf{ implies } \textbf{ not } B,\ C \textbf{ implies } A \textbf{ and } B,\ B \textbf{ implies } A \textbf{ xor } C,\ A \textbf{ or } B \textbf{ or } C\}$$

```
> bsolve(sys);
```

$$\{C = 0,\ B = 0,\ A = 1\}$$

Menteurs et changeants (fin du problème «énigmes logiques»)

On exprime les propositions PX, PY, PZ énoncées par X, Y, Z :

```
> PX:=(not(Y) xor not(T));
> PY:=(Y implies not(X));
> PZ:=(not(Y) and Z) or (Y and not(Z));
```

$$PX := Y \textbf{ xor } T$$
$$PY := Y \textbf{ implies } \textbf{ not } X$$
$$PZ := \textbf{ not } Y \textbf{ and } Z \textbf{ or } Y \textbf{ and } \textbf{ not } Z$$

Pour X, par exemple, on exprime qu'on ne peut avoir en même temps «X vraie» (X menteur) et «PX vraie». On obtient un système de trois équations.

```
> sys:={not(X and PX), not(Y and PY), not(Z and PZ)}:
```

On résout ce système avec **bsolve**.

```
> res:=bsolve(sys);
```

$$res := \{T = 0,\ X = 0,\ Z = 0,\ Y = 0\},\ \{X = 0,\ Z = 0,\ T = 1,\ Y = 0\},$$
$$\{T = 0,\ X = 1,\ Z = 0,\ Y = 0\},\ \{X = 1,\ Z = 0,\ T = 1,\ Y = 1\},$$
$$\{X = 1,\ Z = 1,\ T = 1,\ Y = 1\}$$

Comme il y a au moins deux menteurs, il ne reste que deux solutions, qui impliquent toutes deux T=1.

Puisque T est un menteur, c'est que le problème était intéressant.