

Model Checking et systèmes Ouverts

M2 Informatique UPSAY, parcours CILS

Serenella Cerrito, Tarek Melliti

Laboratoire IBISC et Département d'Informatique – Université d'Évry

premier semestre 2016-2017

1. Introduction, Motivations et Objectifs (S. Cerrito)
2. Syntaxe et Sémantique de la Logique ATL (S. Cerrito)
3. Décision de la Satisfiabilité pour ATL (S. Cerrito)
4. Model Checking pour ATL (T. Melliti)

Merci à Amélie David, docteur en informatique (depuis septembre 2015) pour des exemples tirés de son mémoire de thèse.

Système (informatique) **ouvert** = interagit avec un environnement dont le comportement est inconnu (ou partiellement connu).

Modélisation : système **multi-agents**.

Composantes du système = des agents qui essayient de «gagner» à coût sûr contre l'agent ennemi environnement.

Ici :

Interaction système-environnement représentée par un **CGM** (*Concurrent Game Model*, modèle de jeu concurrent);

Expression des propriétés du système par des formules de la logique **ATL** (*Alternating-time Temporal Logic*, R. Alur, Th. A. Henzinger, O. Kupferman 2002)

Exemple intuitif de CGM : charriot et robots

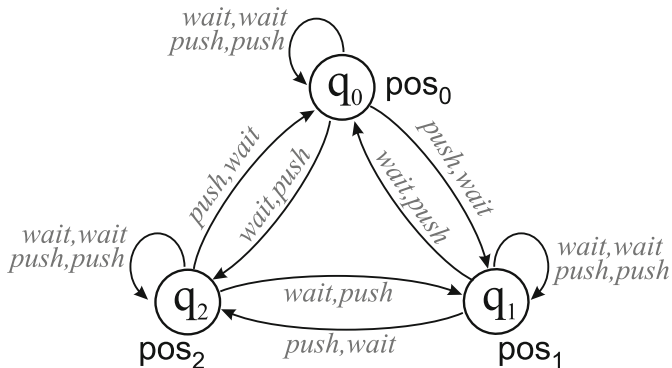


Figure 1

Deux robots, un à la gauche d'un charriot et l'autre à sa droite, peuvent pousser le charriot. Donc le charriot peut bouger de la gauche vers la droite ou de la droite vers la gauche.

- ▶ On veut vérifier (automatiquement) qu'un CGM donnée a une propriété donnée : par exemple, que le système peut atteindre un état donné.

Notre exemple : en partant de q_0 , le charriot peut-il arriver en q_1 ? Ou encore : peut-il aller en q_1 , puis en q_2 , puis à nouveau en q_0 , et y rester pour toujours ?

On fait alors du *model checking* ;

- ▶ On veut vérifier (automatiquement) qu'un CGM donnée a une propriété donnée : par exemple, que le système peut atteindre un état donné.

Notre exemple : en partant de q_0 , le charriot peut-il arriver en q_1 ? Ou encore : peut-il aller en q_1 , puis en q_2 , puis à nouveau en q_0 , et y rester pour toujours ?

On fait alors du *model checking* ;

- ▶ On veut tester (automatiquement) si une spécification formelle d'un système $\mathcal{S}$ (une formule de la logique ATL décrivant des propriétés de $\mathcal{S}$) est *cohérente* et, si possible, *extraire automatiquement un CGM décrivant le comportement d'un système qui respecte $\mathcal{S}$* .

On fait alors de la *décision de la satisfiabilité* et de la *synthèse de modèles*.

- ▶ On veut vérifier (automatiquement) qu'un CGM donnée a une propriété donnée : par exemple, que le système peut atteindre un état donné.

Notre exemple : en partant de q_0 , le charriot peut-il arriver en q_1 ? Ou encore : peut-il aller en q_1 , puis en q_2 , puis à nouveau en q_0 , et y rester pour toujours ?

On fait alors du *model checking* ;

- ▶ On veut tester (automatiquement) si une spécification formelle d'un système $\mathcal{S}$ (une formule de la logique ATL décrivant des propriétés de $\mathcal{S}$) est *cohérente* et, si possible, *extraire automatiquement un CGM décrivant le comportement d'un système qui respecte $\mathcal{S}$* .

On fait alors de la *décision de la satisfiabilité* et de la *synthèse de modèles*.

Ce cours : on étudie les deux tâches, en partant de la deuxième.

CGS, définition formelle)

Une **CGS** (*Concurrent Game Structure*) est une 4-ple

$\mathcal{S} = \langle \mathbb{A}, \mathbb{S}, \{\text{Act}_a\}_{a \in \mathbb{A}}, \text{out} \rangle$ où :

- $\mathbb{A} = \{1, \dots, k\}$ est un ensemble non vide d'*agents* (ou *joueurs*) ;
- $\mathbb{S}$ est un ensemble non vide d'*états* ;
- Pour tout $a \in \mathbb{A}$, Act_a est un ensemble non vide d'*actions*. Si $A \subseteq \mathbb{A}$, alors A est une *coalition*, $\text{Act}_A = \prod_{a \in A} \text{Act}_a$ et σ_A indique un élément de Act_A (une *action de la coalition A*). Si $a \in A$, $\sigma_A(a)$ indique l'action de l'agent a dans le vecteur d'actions σ_A .
- Si $a \in \mathbb{A}$ et $s \in \mathbb{S}$, une fonction act associe à chaque état s un sous-ensemble non vide de Act_a : $\text{act}_a(s)$ indique l'ensemble des *actions disponibles pour l'agent a à l'état s*. De même, une fonction Act associe à s un sous-ensemble non vide de Act_A et $\text{act}_A(s)$ indique l'ensemble des *actions disponibles pour la coalition A à l'état s*. On a : $\text{act}_A(s) = \prod_{a \in A} \text{act}_a(s)$.

Alors : **out** est une *fonction de transition*, qui associe à tout $s \in \mathbb{S}$ et tout $\sigma_{\mathbb{A}} = \langle \sigma_1, \dots, \sigma_k \rangle \in \text{act}_{\mathbb{A}}(s)$ un état $\text{out}(s, \sigma_{\mathbb{A}}) \in \mathbb{S}$: l'état atteint quand chaque agent $a \in \mathbb{A}$ fait (joue) l'action σ_a à s .

Un **CGM** (*Concurrent Game Model*) est une 6-ple

$\mathcal{M} = \langle \mathbb{A}, \mathbb{S}, \{\text{Act}_a\}_{a \in \mathbb{A}}, \text{out}, P, L \rangle$ où :

- $\langle \mathbb{A}, \mathbb{S}, \{\text{Act}_a\}_{a \in \mathbb{A}}, \text{out} \rangle$ est une CGS ;
- P est un ensemble non vide de symboles de propositions (propositions atomiques) ;
- L (*labelling*) est une fonction qui associe à chaque état $s \in \mathbb{S}$ un sous-ensemble de P (l'ensemble des propositions vraies à s).

N.B. : Différence entre une CGS $\mathcal{S}$ et un CGM $\mathcal{M}$?

$\mathcal{M}$ est obtenu à partir de $\mathcal{S}$ en y ajoutant une déclaration de quelles propositions atomiques sont vraies aux états.

Exemple du charriot et rdes obots (fig. 1), à nouveau

Ici, pour la CGS $\langle \mathbb{A}, \mathbb{S}, \{\text{Act}_a\}_{a \in \mathbb{A}}, \text{out} \rangle$:

- $\mathbb{A} = \{1, 2\}$ (robot1 et robot2)
- $\mathbb{S} = \{q_0, q_1, q_2\}$
- $\text{Act}_1 = \text{Act}_2 = \{\text{wait}, \text{push}\}$.

$\text{Act}_{\mathbb{A}} = \{\langle \text{wait}, \text{wait} \rangle, \langle \text{wait}, \text{push} \rangle, \langle \text{push}, \text{wait} \rangle, \langle \text{push}, \text{push} \rangle\}$ (et chacun de ces vecteurs d'actions est disponible pour $\mathbb{A}$ à chaque état)

- $\text{out}(q_0, \langle \text{wait}, \text{wait} \rangle) = \text{out}(q_0, \langle \text{push}, \text{push} \rangle) = q_0$
 $\text{out}(q_0, \langle \text{push}, \text{wait} \rangle) = q_1$ et $\text{out}(q_0, \langle \text{wait}, \text{push} \rangle) = q_2$
 $\text{out}(q_1, \langle \text{wait}, \text{wait} \rangle) = \text{out}(q_1, \langle \text{push}, \text{push} \rangle) = q_1$
 $\text{out}(q_1, \langle \text{wait}, \text{push} \rangle) = q_0$ et $\text{out}(q_1, \langle \text{push}, \text{wait} \rangle) = q_2$
 $\text{out}(q_2, \langle \text{wait}, \text{wait} \rangle) = \text{out}(q_2, \langle \text{push}, \text{push} \rangle) = q_2$
 $\text{out}(q_2, \langle \text{push}, \text{wait} \rangle) = q_0$ et $\text{out}(q_2, \langle \text{wait}, \text{push} \rangle) = q_1$

Exemple du charriot et rdes obots (fig. 1), à nouveau

Ici, pour la CGS $\langle \mathbb{A}, \mathbb{S}, \{\text{Act}_a\}_{a \in \mathbb{A}}, \text{out} \rangle$:

- $\mathbb{A} = \{1, 2\}$ (robot1 et robot2)
- $\mathbb{S} = \{q_0, q_1, q_2\}$
- $\text{Act}_1 = \text{Act}_2 = \{\text{wait}, \text{push}\}$.

$\text{Act}_{\mathbb{A}} = \{\langle \text{wait}, \text{wait} \rangle, \langle \text{wait}, \text{push} \rangle, \langle \text{push}, \text{wait} \rangle, \langle \text{push}, \text{push} \rangle\}$ (et chacun de ces vecteurs d'actions est disponible pour $\mathbb{A}$ à chaque état)

- $\text{out}(q_0, \langle \text{wait}, \text{wait} \rangle) = \text{out}(q_0, \langle \text{push}, \text{push} \rangle) = q_0$
 $\text{out}(q_0, \langle \text{push}, \text{wait} \rangle) = q_1$ et $\text{out}(q_0, \langle \text{wait}, \text{push} \rangle) = q_2$
 $\text{out}(q_1, \langle \text{wait}, \text{wait} \rangle) = \text{out}(q_1, \langle \text{push}, \text{push} \rangle) = q_1$
 $\text{out}(q_1, \langle \text{wait}, \text{push} \rangle) = q_0$ et $\text{out}(q_1, \langle \text{push}, \text{wait} \rangle) = q_2$
 $\text{out}(q_2, \langle \text{wait}, \text{wait} \rangle) = \text{out}(q_2, \langle \text{push}, \text{push} \rangle) = q_2$
 $\text{out}(q_2, \langle \text{push}, \text{wait} \rangle) = q_0$ et $\text{out}(q_2, \langle \text{wait}, \text{push} \rangle) = q_1$

On peut transformer $\mathcal{S}$ en un CGM $\mathcal{M}$ en posant :

$$P = \{\text{pos}_0, \text{pos}_1, \text{pos}_2\};$$

$$L(q_0) = \{\text{pos}_0\}, L(q_1) = \{\text{pos}_1\}, \text{ et } L(q_2) = \{\text{pos}_2\}.$$

CGM : les actions des agents $\mathbb{A}$ sont **synchrones** par définition.

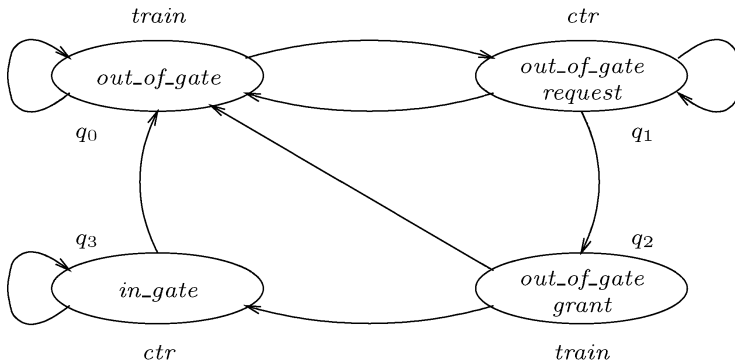
Mais on peut simuler des jeux où chacun joue à son tour : l'action faite par un agent à un état est : *do_nothing*



TURN-BASED synchronous CGM, où, à chaque état, un seul agent peut effectuer une véritable action.

Exemple de turn-based sync CGM : train et passage à niveau

Deux agents, un train et un contrôleur (*ctr*). L'agent *train* a le droit de jouer aux états q_0 et q_2 , et *ctr* aux deux autres. La figure ne montre pas les vecteurs d'actions étiquettes des transitions.



Exemple de turn-based sync. CGM : train et passage à niveau

$\mathbb{S} = \{q_0, q_1, q_2, q_3\}$, $\mathbb{A} = \{train, ctr\}$

$P = \{out_of_gate, request, grant, in_gate\}$

$L(q_0) = \{out_of_gate\}$, $L(q_1) = \{out_of_gate, request\}$,

$L(q_2) = \{out_of_gate, grant\}$, $L(q_3) = \{in_gate\}$

Lire : *out_of_gate* = le train n'est pas rentré (par la porte),
request = le train a demandé la permission de rentrer, *grant* = le
contrôleur a donné la permission de rentrer, *in_gate* = le train est
rentré par la porte (qui s'est refermée).

Le train (joueur 1) peut jouer à q_0 et q_2 ; aux autres états :

do_nothing : c'est le tour du contrôleur.

Le contrôleur (joueur 2) peut jouer à q_1 et q_3 ; aux autres états :

do_nothing : c'est le tour du train.

Exemple de turn-based sync CGM : train et passage à niveau

Actions collectives possibles, état par état

$$\text{act}_{\text{train}, \text{ctr}}(q_0) = \underbrace{\{\langle \text{stay_out}, \text{do_nothing} \rangle\}}_{\sigma_1}, \underbrace{\{\langle \text{request_enter}, \text{do_nothing} \rangle\}}_{\sigma_2}$$

$$\text{act}_{\text{train}, \text{ctr}}(q_1) = \underbrace{\{\langle \text{do_nothing}, \text{delay} \rangle\}}_{\sigma_3}, \underbrace{\{\langle \text{do_nothing}, \text{refuse} \rangle\}}_{\sigma_4}, \underbrace{\{\langle \text{do_nothing}, \text{accept} \rangle\}}_{\sigma_5}$$

$$\text{act}_{\text{train}, \text{ctr}}(q_2) = \underbrace{\{\langle \text{enter}, \text{do_nothing} \rangle\}}_{\sigma_6}, \underbrace{\{\langle \text{renounce}, \text{do_nothing} \rangle\}}_{\sigma_7}$$

$$\text{act}_{\text{train}, \text{ctr}}(q_3) = \underbrace{\{\langle \text{do_nothing}, \text{keep_closed} \rangle\}}_{\sigma_8}, \underbrace{\{\langle \text{do_nothing}, \text{reopen} \rangle\}}_{\sigma_9}$$

Exemple de turn-based sync CGM : train et passage à niveau

Transitions

- ▶ $\text{out}(q_0, \sigma_1) = q_0$, $\text{out}(q_0, \sigma_2) = q_1$
- ▶ $\text{out}(q_1, \sigma_3) = q_1$, $\text{out}(q_1, \sigma_4) = q_0$, $\text{out}(q_1, \sigma_5) = q_2$
- ▶ $\text{out}(q_2, \sigma_6) = q_3$, $\text{out}(q_2, \sigma_7) = q_0$
- ▶ $\text{out}(q_3, \sigma_8) = q_3$, $\text{out}(q_3, \sigma_9) = q_0$

Une formule de ATL permet de décrire un comportement des agents d'une CGS.

Opérateurs temporels : $\bigcirc$ (à l'état suivant), $\Diamond$ (maintenant ou à un état future), $\Box$ (maintenant et toujours dans le future), U (until).

Explication intuitive : au tableau

Ce sont les opérateurs temporels de LTL (*Linear Temporal Logic*) !

Quantificateur (existantiel) de stratégie :

$\langle\langle A \rangle\rangle O$ où A est un ensemble de joueurs (*coalition*) et O est une formule représentant un objectif : la coalition A a une **stratégie** telle que, **peu importe ce que les autres joueurs font**, l'objectif O est atteint.

$\langle\langle robot1 \rangle\rangle \bigcirc pos_1$: fausse à q_0 . Pourquoi ?

$\langle\langle robot1 \rangle\rangle \bigcirc pos_1$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si *robot1* décide de jouer l'action *push* à q_0 , si *robot2* décide de jouer *push* le chariot reste sur place.

Formules de ATL : Intuition, avec l'exemple robots et chariot

$\langle\langle robot1 \rangle\rangle \bigcirc pos_1$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si *robot1* décide de jouer l'action *push* à q_0 , si *robot2* décide de jouer *push* le chariot reste sur place.

$\langle\langle robot1, robot2 \rangle\rangle \bigcirc pos_1$: vraie à q_0 . **Pourquoi ?**

Formules de ATL : Intuition, avec l'exemple robots et chariot

$\langle\langle robot1 \rangle\rangle \bigcirc pos_1$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si *robot1* décide de jouer l'action *push* à q_0 , si *robot2* décide de jouer *push* le chariot reste sur place.

$\langle\langle robot1, robot2 \rangle\rangle \bigcirc pos_1$: vraie à q_0 . **Pourquoi ?** Par-ce-que la coalition des deux robots peut décider de jouer $\langle push, wait \rangle$ à q_0 .

Formules de ATL : Intuition, avec l'exemple robots et chariot

$\langle\langle robot1 \rangle\rangle \bigcirc pos_1$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si *robot1* décide de jouer l'action *push* à q_0 , si *robot2* décide de jouer *push* le chariot reste sur place.

$\langle\langle robot1, robot2 \rangle\rangle \bigcirc pos_1$: vraie à q_0 . **Pourquoi ?** Par-ce-que la coalition des deux robots peut décider de jouer $\langle push, wait \rangle$ à q_0 .

$\langle\langle robot1, robot2 \rangle\rangle \Box (pos_2 \vee pos_3)$: fausse à q_0 . **Pourquoi ?**

Formules de ATL : Intuition, avec l'exemple robots et chariot

$\langle\langle robot1 \rangle\rangle \bigcirc pos_1$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si *robot1* décide de jouer l'action *push* à q_0 , si *robot2* décide de jouer *push* le chariot reste sur place.

$\langle\langle robot1, robot2 \rangle\rangle \bigcirc pos_1$: vraie à q_0 . **Pourquoi ?** Par-ce-que la coalition des deux robots peut décider de jouer $\langle push, wait \rangle$ à q_0 .

$\langle\langle robot1, robot2 \rangle\rangle \Box (pos_2 \vee pos_3)$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si les deux robots peuvent agir de façon à rester toujours dans q_1 ou q_2 , dès qu'ils sont dans un de ces deux états, à l'état q_0 la formule $(pos_2 \vee pos_3)$ est fausse.

Formules de ATL : Intuition, avec l'exemple robots et chariot

$\langle\langle robot1 \rangle\rangle \bigcirc pos_1$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si *robot1* décide de jouer l'action *push* à q_0 , si *robot2* décide de jouer *push* le chariot reste sur place.

$\langle\langle robot1, robot2 \rangle\rangle \bigcirc pos_1$: vraie à q_0 . **Pourquoi ?** Par-ce-que la coalition des deux robots peut décider de jouer $\langle push, wait \rangle$ à q_0 .

$\langle\langle robot1, robot2 \rangle\rangle \Box (pos_2 \vee pos_3)$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si les deux robots peuvent agir de façon à rester toujours dans q_1 ou q_2 , dès qu'ils sont dans un de ces deux états, à l'état q_0 la formule $(pos_2 \vee pos_3)$ est fausse.

$(\langle\langle robot1, robot2 \rangle\rangle \Diamond pos_1) \wedge (\langle\langle robot1, robot2 \rangle\rangle \Diamond pos_2)$: vraie à q_0 .
Pourquoi ?

Formules de ATL : Intuition, avec l'exemple robots et chariot

$\langle\langle robot1 \rangle\rangle \bigcirc pos_1$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si *robot1* décide de jouer l'action *push* à q_0 , si *robot2* décide de jouer *push* le chariot reste sur place.

$\langle\langle robot1, robot2 \rangle\rangle \bigcirc pos_1$: vraie à q_0 . **Pourquoi ?** Par-ce-que la coalition des deux robots peut décider de jouer $\langle push, wait \rangle$ à q_0 .

$\langle\langle robot1, robot2 \rangle\rangle \Box (pos_2 \vee pos_3)$: fausse à q_0 . **Pourquoi ?** Par-ce-que même si les deux robots peuvent agir de façon à rester toujours dans q_1 ou q_2 , dès qu'ils sont dans un de ces deux états, à l'état q_0 la formule $(pos_2 \vee pos_3)$ est fausse.

$(\langle\langle robot1, robot2 \rangle\rangle \Diamond pos_1) \wedge (\langle\langle robot1, robot2 \rangle\rangle \Diamond pos_2)$: vraie à q_0 . **Pourquoi ?** Par-ce-que *une stratégie* où la coalition des deux robots joue $\langle push, wait \rangle$ à q_0 assure d'aller en q_1 et *une (autre) stratégie* où la coalition des deux robot joue $\langle wait, push \rangle$ à q_0 assure d'aller en q_2 .

Grammaire des formules ATL

P = ensemble de symboles de propositions (variables booléennes)

$Atomes$ (formules atomiques) $:= P \cup \{T\}$

$F := a \mid (\neg F) \mid (F \wedge F) \mid \langle\langle A \rangle\rangle \bigcirc F \mid \langle\langle A \rangle\rangle \Box F \mid$
 $\langle\langle A \rangle\rangle F \cup F$

où :

$a \in Atomes$, A = un sous ensemble des joueurs, dit *coalition* .

Grammaire des formules ATL

P = ensemble de symboles de propositions (variables booléennes)

$Atomes$ (formules atomiques) $:= P \cup \{T\}$

$F := a \mid (\neg F) \mid (F \wedge F) \mid \langle\langle A \rangle\rangle \bigcirc F \mid \langle\langle A \rangle\rangle \Box F \mid$
 $\langle\langle A \rangle\rangle F \cup F$

où :

$a \in Atomes$, A = un sous ensemble des joueurs, dit *coalition* .

$A \vee B =_{Def} \neg(\neg A \wedge \neg B)$, $A \rightarrow B =_{Def} \neg A \vee B$ et

$\Diamond F =_{Def} T \cup F$

Conventions usuelles pour réduire le nombre de parenthèses

Grammaire des formules ATL

P = ensemble de symboles de propositions (variables booléennes)

$Atomes$ (formules atomiques) $:= P \cup \{T\}$

$F := a \mid (\neg F) \mid (F \wedge F) \mid \langle\langle A \rangle\rangle \bigcirc F \mid \langle\langle A \rangle\rangle \Box F \mid \langle\langle A \rangle\rangle F \cup F$

où :

$a \in Atomes$, A = un sous ensemble des joueurs, dit *coalition* .

$A \vee B =_{Def} \neg(\neg A \wedge \neg B)$, $A \rightarrow B =_{Def} \neg A \vee B$ et

$\Diamond F =_{Def} T \cup F$

Conventions usuelles pour réduire le nombre de parenthèses

NB : Un quantificateur stratégique ($\langle\langle A \rangle\rangle$) est **toujours** suivi par un opérateur temporel ($\bigcirc, \Box, \cup, \Diamond$), et un opérateur temporel est **toujours** précédé par un quantificateur stratégique.

Exemples de formules

Ensemble des joueurs : $\{1, 2, 3, 4\}$.

$$P = \{p, q, r\}$$

$$\langle\langle 1, 3 \rangle\rangle \bigcirc (p \wedge q)$$

$$\neg \langle\langle 1 \rangle\rangle \Box (p \wedge q)$$

$$\langle\langle 1, 2, 3, 4 \rangle\rangle r \cup (p \vee q)$$

$$(\langle\langle 1, 3 \rangle\rangle r \cup (p \vee q)) \wedge (\neg \langle\langle 1, 4 \rangle\rangle \Box r)$$

$$\langle\langle \emptyset \rangle\rangle \Box r$$

tandis que

$$\langle\langle 3, 4 \rangle\rangle (\Box (p \wedge q) \wedge \Diamond r)$$

n'est pas une formule ATL. Pourquoi ?

Exemples de formules

Ensemble des joueurs : $\{1, 2, 3, 4\}$.

$P = \{p, q, r\}$

$\langle\langle 1, 3 \rangle\rangle \bigcirc (p \wedge q)$

$\neg \langle\langle 1 \rangle\rangle \Box (p \wedge q)$

$\langle\langle 1, 2, 3, 4 \rangle\rangle r \cup (p \vee q)$

$(\langle\langle 1, 3 \rangle\rangle r \cup (p \vee q)) \wedge (\neg \langle\langle 1, 4 \rangle\rangle \Box r)$

$\langle\langle \emptyset \rangle\rangle \Box r$

tandis que

$\langle\langle 3, 4 \rangle\rangle (\Box (p \wedge q) \wedge \Diamond r)$

n'est pas une formule ATL. Pourquoi ? Par-ce-que le quantificateur domine $\wedge$, qui n'est pas une opérateur temporel.

Mots Clés :

CGM

exécution

stratégie

Soit un cgm $\mathcal{M} = \langle \mathbb{A}, \mathbb{S}, \{\text{Act}_a\}_{a \in \mathbb{A}}, \text{out}, P, L \rangle$

Une **exécution** (chémin) λ (ayant un état s_0 comme départ) est une suite **infinie** d'éléments de $\mathbb{S} : s_0, s_1, s_2, \dots$ telle que, pour chaque $i \geq 0$, $\exists$ un vecteur d'actions $\sigma_{\mathbb{A}} \in \text{act}_{\mathbb{A}}(s_i)$ tel que $\text{out}(s_i, \sigma_{\mathbb{A}}) = s_{i+1}$.

Autre façon d'indiquer une exécution $\lambda : \lambda_0, \lambda_1, \lambda_2, \dots$

Etant donné une coalition A : une **stratégie sans mémoire** F_A est une fonction qui associe à tout $s \in \mathbb{S}$ un vecteur d'actions σ_A tel que l'action de σ_A appartient à $\text{act}_A(s)$ (l'ensemble d'actions disponibles pour la coalition A à l'état s).

Idée : Une stratégie pour A indique les choix d'actions faites par A , état par état. Attention : chaque fois que A passe par un état donné s , elle doit faire le même choix (pas de mémoire de comment on est arrivé à s).

Etant donné une coalition A : une **stratégie avec mémoire parfaite** F_A est une fonction qui associe à toute sequence initiale finie d'exécution $s_0, s_1, s_2, \dots, s_n$ un vecteur d'actions σ_A tel que l' action σ_A appartient à $\text{act}_A(s_n)$ (l'ensemble d'actions disponibles pour la coalition A à l'état s_n).

Idée : Une stratégie avec mémoire parfaite pour A peut faire dépendre les choix d'actions de A , à un état donné s , non seulement de s , mais aussi des états déjà traversés pour arriver à s : ainsi, chaque fois qu'on passe par s , la coalition A peut choisir un vecteur d'actions différent. Cela signifie que *le choix d'actions de A à s peut même changer chaque fois qu'on passe par s !*. En fait, chaque fois qu'on passe par s on aura un « passé » différent.

Pour toutes les définitions qui suivent, on se place dans le cadre où toutes les stratégies sont sans mémoire. On parle alors de **sémantique sans mémoire**

Ce choix est justifié, pour ATL, et on verra après pourquoi.

Soit σ_A une action collective de la coalition A . La notation $\text{Out}(s, \sigma_A)$ (*outcome de σ_A à s*) indique l'ensemble de ces états $\text{out}(s, \sigma_{\mathbb{A}})$ où $\sigma_{\mathbb{A}}$ est n'importe quel vecteur global d'actions ayant la propriété : pour chaque joueur a de la coalition A , $\sigma_A(a) = \sigma_{\mathbb{A}}(a)$.

Autrement dit : c'est l'ensemble de tous les états qui sont successeurs de s (selon la fonction de transition de $\mathcal{M}$) quand la coalition A joue l'action collective σ_A (et les autres joueurs jouent n'importe quelle action légale).

Exemple 1. Reprennons l'exemple des robots et du chariot, soit $A = \{\text{robot1}\}$ et soit $\sigma_A = \text{push}$. Alors $\text{Out}(q_0, \sigma_A) = \{q_0, q_1\}$.

Exemple 2. Soit $\mathcal{M}$ un cgm avec trois joueurs, 1, 2 et 3, et quatre états : q_0, q_1, q_2 et q_3 .

Supposons : à l'état q_0 le joueur 1 peut jouer juste une action a , le joueur 2 peut jouer soit b_1 soit b_2 , et le joueur 3 peut jouer soit c_1 soit c_2 .

Soit : $\text{out}(q_0, \langle a, b_1, c_1 \rangle) = q_0$,

$\text{out}(q_0, \langle a, b_2, c_1 \rangle) = \text{out}(q_0, \langle a, b_1, c_2 \rangle) = q_3$,

$\text{out}(q_0, \langle a, b_2, c_2 \rangle) = q_2$.

Alors :

- pour la coalition $A = \{1, 3\}$ et l'action $\sigma_A = \langle a, c_2 \rangle$ de cette coalition, $\text{Out}(q_0, \sigma_A) = \{q_2, q_3\}$.

Exemple 2. Soit $\mathcal{M}$ un cgm avec trois joueurs, 1, 2 et 3, et quatre états : q_0, q_1, q_2 et q_3 .

Supposons : à l'état q_0 le joueur 1 peut jouer juste une action a , le joueur 2 peut jouer soit b_1 soit b_2 , et le joueur 3 peut jouer soit c_1 soit c_2 .

Soit : $\text{out}(q_0, \langle a, b_1, c_1 \rangle) = q_0$,

$\text{out}(q_0, \langle a, b_2, c_1 \rangle) = \text{out}(q_0, \langle a, b_1, c_2 \rangle) = q_3$,

$\text{out}(q_0, \langle a, b_2, c_2 \rangle) = q_2$.

Alors :

- pour la coalition $A = \{1, 3\}$ et l'action $\sigma_A = \langle a, c_2 \rangle$ de cette coalition, $\text{Out}(q_0, \sigma_A) = \{q_2, q_3\}$.
- pour la coalition $A = \{b_1\}$ et l'action $\sigma_A = \langle b_1 \rangle$ de cette coalition, $\text{Out}(q_0, \sigma_A) = \{q_0, q_3\}$

Soit A une coalition.

Une exécution $\lambda = \lambda_0, \lambda_1, \dots$ est dite **cohérente avec une stratégie F_A** ssi quelque soit $j \geq 0$, $\lambda_{j+1} \in \text{Out}(\lambda_j, \sigma_A)$, où σ_A est le vecteur d'actions choisi par F_A à l'état λ_j

Idee : Une exécution λ est cohérente avec une stratégie F_A si c'est une exécution où la coalition A suit la stratégie F_A .

Si F est une formule, lire $\mathcal{M}, s \models F$ comme : l'état s de $\mathcal{M}$ satisfait la formule F .

- ▶ $\mathcal{M}, s \models T$
- ▶ $\mathcal{M}, s \models p$, où $p \in P$, ssi $p \in L(s)$
- ▶ $\mathcal{M}, s \models \neg F$ ssi $\mathcal{M}, s \not\models F$
- ▶ $\mathcal{M}, s \models F \wedge G$ ssi $\mathcal{M}, s \models F$ et $\mathcal{M}, s \models G$
- ▶ $\mathcal{M}, s \models \langle\langle A \rangle\rangle \bigcirc F$ ssi $\exists$ un vecteur d'actions $\sigma_A \in \text{act}_A(s)$ tel que, qqe soit $s' \in \text{Out}(s, \sigma_A)$, $\mathcal{M}, s' \models F$

- ▶ $\mathcal{M}, s \models \langle\langle A \rangle\rangle \Box F$ ssi $\exists$ une stratégie F_A telle que pour toute exécution λ commençant à $s = \lambda_0$ et cohérente avec F_A , $\mathcal{M}, \lambda_i \models F$, quelque soit $i \geq 0$.
- ▶ $\mathcal{M}, s \models \langle\langle A \rangle\rangle FUG$ ssi $\exists$ une stratégie F_A telle que pour toute exécution λ commençant à $s = \lambda_0$ et cohérente avec F_A , il existe un $i \geq 0$ tel que :
 $\mathcal{M}, \lambda_i \models G$ et
 quelque soit j , $0 \leq j < i$, $\mathcal{M}, \lambda_j \models F$

- ▶ $\mathcal{M}, s \models \langle\langle A \rangle\rangle \Box F$ ssi $\exists$ une stratégie F_A telle que pour toute exécution λ commençant à $s = \lambda_0$ et cohérente avec F_A , $\mathcal{M}, \lambda_i \models F$, quelque soit $i \geq 0$.
- ▶ $\mathcal{M}, s \models \langle\langle A \rangle\rangle FUG$ ssi $\exists$ une stratégie F_A telle que pour toute exécution λ commençant à $s = \lambda_0$ et cohérente avec F_A , il existe un $i \geq 0$ tel que :
 $\mathcal{M}, \lambda_i \models G$ et
 quelque soit j , $0 \leq j < i$, $\mathcal{M}, \lambda_j \models F$

Question : pourquoi poser $\Diamond F =_{Def} TUF$ est sensé ?

Soit une formule de la forme $\langle\langle A \rangle\rangle H$. Intuitivement, elle est vraie à un état s de $\mathcal{M}$ ssi $\exists$ une stratégie F_A pour la coalition A telle que $\forall$ façon de jouer des autres agents, toute exécution λ commençant à $s = \lambda_0$ et cohérente avec F_A rendra vraie la formule H .

Que veut dire alors que $\neg\langle\langle A \rangle\rangle H$ est vraie à s ?

Il veut dire que $\nexists$ une stratégie F_A pour la coalition A telle que $\forall$ façon de jouer des autres agents, toute exécution λ commençant à $s = \lambda_0$ et cohérente avec F_A rendra vraie la formule temporelle H .

Dit autrement : $\forall$ stratégie F_A pour la coalition A , $\exists$ une façon de jouer des autres agents assurant l'existence d'au moins une exécution λ commençant à $s = \lambda_0$, cohérente avec F_A , et rendant fausse la formule temporelle H .

On dit : *la coalition A ne peut pas éviter $\neg H$.*

Cas particulier où A est la coalition **vide** : $\mathcal{M}, s \models \langle\langle \emptyset \rangle\rangle G$ veut dire que **toute** exécution qui part de s rend vraie la formule temporelle G . Pourquoi ?

Cas particulier où A est la coalition **vide** : $\mathcal{M}, s \models \langle\langle \emptyset \rangle\rangle G$ veut dire que **toute** exécution qui part de s rend vraie la formule temporelle G . Pourquoi ?

Si A est vide alors la stratégie F_A ne fait aucun choix, tandis que les «autres agents» sont tout le monde !

En effet, toute exécution sera cohérente avec F_A , donc

$\mathcal{M}, s \models \langle\langle \emptyset \rangle\rangle G$ ssi toute exécution qui part de s rend vraie la formule temporelle G . .

Soit à nouveau l'exemple du charriot et des deux robots

Évaluer à l'état q_0 :

- ▶ $\langle\langle robot1 \rangle\rangle \bigcirc pos_1$
- ▶ $\langle\langle robot1 \rangle\rangle \Box pos_0$
- ▶ $\langle\langle robot1, robot2 \rangle\rangle \Box pos_0$
- ▶ $\langle\langle robot1, robot2 \rangle\rangle (pos_0 \vee pos_1) \cup pos_2$
- ▶ $\langle\langle robot1 \rangle\rangle \Box (pos_0 \vee pos_1 \vee pos_2)$
- ▶ $\langle\langle \emptyset \rangle\rangle \Box (pos_0 \vee pos_1 \vee pos_2)$

Un CGM $\mathcal{M}$ *satisfait* une formule F si $\mathcal{M}, s \models F$ pour quelque état s de $\mathcal{M}$. On dit que $\mathcal{M}$ est un *modèle* de F .

Une formule F est *vraie* dans un CGM $\mathcal{M}$ si $\mathcal{M}, s \models F$ pour tout état s de $\mathcal{M}$.

Une formule F est (*tightly*) **satisfiable** s'il existe un CGM $\mathcal{M}$, dont les agents sont ceux qui apparaissent dans F , qui est un modèle de F . Une formule F est (*tightly*) **insatisfiable** si F n'est pas (*tightly*) satisfiable.

Une formule F est (*tightly*) **valide** si elle est vraie pour toute CGM $\mathcal{M}$ dont les agents sont ceux qui apparaissent dans F .

Dans la suite, on omet le mot *tightly*.

NB : F est valide si et seulement si $\neg F$ est insatisfiable.

Pourquoi ?

On a donné les définitions sémantiques en supposant que les stratégies sont sans mémoire.

Pour ATL la sémantique sans mémoire et celle avec mémoire parfaite sont équivalentes au sens suivant :

une formule F est satisfiable dans le cas sans mémoire ssi elle l'est dans le cas avec mémoire parfaite (ce n'est pas vrai pour des extensions de ATL, comme ATL*!).

Deux formules F et G sont (*logiquement*) *équivalentes* ssi, quelque soit le CGM $\mathcal{M}$, quelque soit l'état s :

$\mathcal{M}, s \models F$ si et seulement si $\mathcal{M}, s \models G$.

Exercice 1. Considérons à nouveau le CGM du train et du passage à niveau (diapositives 11, 12 et 13).

Traduire par des formules de ATL les énoncés suivants :

1. Il existe une stratégie du contrôleur qui assure que le train est toujours en dehors de la porte (*out_of_gate*).
2. Il existe une stratégie du train qui lui assure de rentrer (*in_gate*) à un moment donné.
3. Le train ne peut pas éviter de se trouver toujours en dehors de la porte.
4. Le train ne peut pas éviter de se trouver en dehors de la porte (*out_of_gate*) à un moment donné.
5. Chaque fois que le train est en dehors de la porte et n'a pas la permission de rentrer (*grant*), le contrôleur peut lui empêcher de rentrer dans la porte

Exercice 2. Un automate de réservation de la SNCF fonctionne ainsi : un utilisateur peut rentrer son numéro de réservation et ou bien demander l'impression de son ticket et l'obtenir, ou bien demander un remboursement et l'obtenir ; dans les deux cas le numéro de réservation n'est plus disponible. Jean a réussi à tricher avec l'aide de son ami Hugo de la façon suivante. Hugo a utilisé un automate avec le numéro de réservation de Jean et il a imprimé le ticket. Au même temps, Jean a utilisé un *autre* automate, il y a rentré le même numéro de réservation et il a obtenu le remboursement.

Modélisez avec un CGM ce « jeux » permettant à Jean et Hugo de coopérer pour tricher .

Suggestion : Utiliser deux agents, Hugo et Jean, poser

$Act_{Hugo} = Act_{Jean} =$

$\{idle(do_nothing), login, ask_print, ask_reimbursement\}$

$P = \{loggedH, loggedJ, ticket_printed,$

$ticket_reimbursed, reservation_unavailable\}$.

Le problème de la satisfiabilité pour ATL

Une méthode de décision du problème de la satisfiabilité d'une formule

=

Un algorithme qui termine et sait répondre à la question « F est satisfiable ? », pour n'importe quelle formule F .

La méthode de décision de la satisfiabilité pour ATL que nous allons voir utilise implicitement des stratégies avec mémoire parfaite.

Mais : on a déjà remarqué que les deux sortes de mémoire ne font pas de différence, par rapport à la satisfiabilité.

Tableaux : une méthode automatique pour décider le problème de la satisfiabilité pour ATL.

C'est une méthode constructive : si elle répond "Oui, la formule F est satisfiable", elle fournit aussi un modèle de F .

Ici : une variante des tableaux pour ATL de V. Goranko et D. Shkatov (*Tableau-based decision procedures for logics of strategic ability in multiagent systems*, 2009 : voir la bibliographie à la fin).

Deux phases : **Construction** et **Elimination**.

La phase de construction crée un graphe dirigé et ayant une racine, dont les sommets sont étiquetés par des ensembles de formules.

Celle d'élimination élimine des sommets.

Construction : Intuition

On construit un graphe qui décrit un *ensemble de CGM candidats à être des modèles* de F .

On a deux sortes de sommets : les *pre-états* Γ_i et les *états* Δ_j .

Un *pre-état* est un ensemble de formules Γ contenant des informations implicites.

Par ex. si $F \wedge G \in \Gamma$, mais $F \notin \Gamma$ ou $G \notin \Gamma$, la formule $F \wedge G$ n'est pas complètement analysée : elle contient de l'information implicite. On explicite en décomposant cette formule en F et G , qui seront ajoutées à Γ .

Un *état du tableau* est un ensemble de formules complètement analysées et correspondra à un état d'un modèle éventuel de F .

NB : surcharge sémantique du mot *état* : d'un tableau, d'un CGM. Mais lien intuitif entre les 2 significations.

Pour tester si une formule F est satisfiable :

- **Initialisation** : On crée la racine, qui est le pre-état $\Gamma_1 = \{F\}$.

Dans la construction du tableau, les joueurs considérés sont ceux qui apparaissent dans F .

- On applique à tour de rôle la règle **SR** qui génère un ou plusieurs états **à partir d'un pre-état**, et la règle **Next** qui génère un ou plusieurs pre-états **à partir d'un état**.
- On fait de retours en arrière si nécessaire $\Rightarrow$ graphe fini.

Une formule de ATL est dite *primitive* si elle a une des formes suivantes :

T , p , ou bien $\neg p$ ($p \in P$)

$\langle\langle A \rangle\rangle \bigcirc F$ (formule de type *successeur* **positive**)

$\neg\langle\langle A \rangle\rangle \bigcirc F$ avec $A \neq \mathbb{A}$ (formule de type *successeur* **négative**).

Pour les formules de type successeur, F (resp. $\neg F$) est dite *composante successeur* positive (resp. négative).

Pour la règle **SR** : décomposition

Les formules **non-primitives** sont classées comme formules- α et formules- β ayant composantes, respectivement, α_1 , α_2 et β_1 , β_2 :

α	α_1	α_2
$F_1 \wedge F_2$	F_1	F_2
$\langle\langle A \rangle\rangle \Box F$	F	$\langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \Box F$
$\neg\neg F$	F	F
$\neg\langle\langle A \rangle\rangle \bigcirc F$	$\langle\langle \emptyset \rangle\rangle \bigcirc \neg F$	$\langle\langle \emptyset \rangle\rangle \bigcirc \neg F$

β	β_1	β_2
$\neg(F_1 \wedge F_2)$	$\neg F_1$	$\neg F_2$
$\langle\langle A \rangle\rangle F_1 \cup F_2$	F_2	$F_1 \wedge \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle F_1 \cup F_2$
$\neg\langle\langle A \rangle\rangle F_1 \cup F_2$	$\neg F_1 \wedge \neg F_2$	$\neg F_2 \wedge \neg\langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle F_1 \cup F_2$
$\neg\langle\langle A \rangle\rangle \Box F$	$\neg F$	$\neg\langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \Box F$

Intuition : les α sont conjonctives, le β sont disjonctives, et :

$\alpha \equiv \alpha_1 \wedge \alpha_2$, $\beta \equiv \beta_1 \vee \beta_2$.

Pour la règle **SR** : Convention

Dans la suite :

Si F est une formule alors, par abus de notation, $\neg F$ indiquera la formule obtenue en prefixant d'abord F par $\neg$, mais en effaçant ensuite chaque double négation (c.à.d. en réécrivant $\neg\neg G$ comme G).

Exemples :

$$\neg(\neg\langle\langle A \rangle\rangle\Box p) = \langle\langle A \rangle\rangle\Box p$$

$$\neg\neg(\neg\langle\langle A \rangle\rangle\Box p) = \neg\langle\langle A \rangle\rangle\Box p$$

$$\neg\neg\neg(\neg\langle\langle A \rangle\rangle\Box p) = \langle\langle A \rangle\rangle\Box p$$

Donc en prefixant F avec peu importe quel nombre de $\neg$ on obtient au max une nouvelle formule, pas un nombre infini !

Pour la règle SR : Cloture

Soit F une formule ATL.

La *cloture* de F , notée $cl(F)$ est le plus petit ensemble de formules tel que :

1. $F \in cl(F)$ et si G est une sous-formule d'une formule $F_1 \in cl(F)$, alors $G \in cl(F)$;
2. Si G est une formule- α et $G \in cl(F)$, alors ses composantes α_1 et α_2 sont dans $cl(F)$;
3. Si G est une formule- β et $G \in cl(F)$, alors ses composantes β_1 et β_2 sont dans $cl(F)$;
4. Si $G \in cl(F)$ alors $\neg G \in cl(F)$;
5. $T \in cl(F)$ et $\langle\langle A \rangle\rangle \bigcirc T \in cl(F)$.

Intuition : $cl(F)$ est un ensemble **fini**, mais il contient toutes les formules qui pourront apparaître dans un tableau de racine $\{F\}$.

Si Γ est un ensemble de formules, $cl(\Gamma) =_{DEF} \bigcup_{F \in \Gamma} cl(F)$.

Exemple de calcul de $cl(F)$

Soit $F = (\neg\langle\langle 1 \rangle\rangle \Box p \wedge \langle\langle 1, 2 \rangle\rangle \bigcirc p) \wedge \neg\langle\langle 2 \rangle\rangle \bigcirc \neg p$
où $p \in P$.

Calcul de $cl(F)$:

au tableau

Pour la règle **SR** : Saturation

Soient Γ et Δ des ensembles de formules ATL avec $\Gamma \subseteq \Delta \subseteq cl(\Gamma)$.

1. Δ est dit *banalement incohérent* s'il contient une paire de formules $F, \neg F$;
2. Δ est une *saturation* de Γ ssi il n'est pas banalement incohérent et :

Si une formule- $\alpha \in \Gamma$, alors $\alpha_1 \in \Gamma$ **et** $\alpha_2 \in \Gamma$

Si une formule- $\beta \in \Gamma$, alors $\beta_1 \in \Gamma$ **ou** $\beta_2 \in \Gamma$

Intuition ?

Soient Γ et Δ des ensembles de formules ATL avec $\Gamma \subseteq \Delta \subseteq cl(\Gamma)$.

1. Δ est dit *banalement incohérent* s'il contient une paire de formules $F, \neg F$;
2. Δ est une *saturation* de Γ ssi il n'est pas banalement incohérent et :

Si une formule- $\alpha \in \Gamma$, alors $\alpha_1 \in \Gamma$ **et** $\alpha_2 \in \Gamma$

Si une formule- $\beta \in \Gamma$, alors $\beta_1 \in \Gamma$ **ou** $\beta_2 \in \Gamma$

Intuition ?

Δ est une saturation de Γ quand il représente de façon explicite un état d'un CGM où toutes les formules de Γ sont vraies.

Exemple de calcul de saturations

Soit $F = \neg\langle\langle 1 \rangle\rangle \Box p \wedge \langle\langle 1, 2 \rangle\rangle \bigcirc p \wedge \neg\langle\langle 2 \rangle\rangle \bigcirc \neg p$

où $p \in P$.

(On omet des $()$ car le $\wedge$ est associatif, et on décompose $A \wedge B \wedge C$ en A, B, C .)

Calcul des ensembles Δ_1 et Δ_2 qui sont les saturations de $\{F\}$: au tableau.

Exemple de calcul de saturations

Soit $F = \neg\langle\langle 1 \rangle\rangle\Box p \wedge \langle\langle 1, 2 \rangle\rangle\bigcirc p \wedge \neg\langle\langle 2 \rangle\rangle\bigcirc \neg p$

où $p \in P$.

(On omet des $()$ car le $\wedge$ est associatif, et on décompose $A \wedge B \wedge C$ en A, B, C .)

Calcul des ensembles Δ_1 et Δ_2 qui sont les saturations de $\{F\}$: au tableau.

On obtient :

- $\Delta_1 = \{F, \neg\langle\langle 1 \rangle\rangle\Box p, \langle\langle 1, 2 \rangle\rangle\bigcirc p, \neg\langle\langle 2 \rangle\rangle\bigcirc \neg p, \neg p\}$
- $\Delta_2 = \{F, \neg\langle\langle 1 \rangle\rangle\Box p, \langle\langle 1, 2 \rangle\rangle\bigcirc p, \neg\langle\langle 2 \rangle\rangle\bigcirc \neg p, \neg\langle\langle 1 \rangle\rangle\bigcirc \langle\langle 1 \rangle\rangle\Box p\}$

NB : $\neg\langle\langle 1 \rangle\rangle\Box p$ est une β -formule est a donné lieu à 2 possibilités.

Règle SR

Pour chaque **pre-état** Γ , faire :

1. Ajouter au graphe courant autant d'**états** que de saturations de Γ ;
2. Pour chaque état Δ que l'on vient d'ajouter, si Δ ne contient aucune formule de type successeur, y ajouter $\langle\langle A \rangle\rangle \bigcirc T$ (ceci assure : l'ensemble des formule successeurs de $\Delta \neq \emptyset$) ;
3. Pour chaque état Δ obtenu aux deux étapes précédentes, créer une arête $\Rightarrow$ de Γ à Δ ;
4. Dans le cas particulier où le noeud Δ existait déjà, **ne pas en créer une seconde copie**, mais lier juste Γ à ce noeud par une arête $\Rightarrow$ (**loop check**).

Soit $F = (\neg\langle\langle 1 \rangle\rangle \Box p \wedge \langle\langle 1, 2 \rangle\rangle \bigcirc p) \wedge \neg\langle\langle 2 \rangle\rangle \bigcirc \neg p$

Initialisation du tableau pour tester la satisfiabilité de F et application de SR :

- ▶ Le sommet initial est le **pre-état** $\Gamma_1 = \{F\}$.
- ▶ Par SR, ce sommet est relié par deux arêtes $\Rightarrow$ à **deux états possibles** résultant de l'analyse : Δ_1 et Δ_2 .

Vers la règle **Next**

La règle **SR** se limite à analyser l'information implicite d'un pre-état Γ , en obtenant des états $\Delta_1, \Delta_2, \dots$ (correspondant à des états des modèles candidats de la racine).

La règle suivante, **Next**, a un rôle **dynamique** : intuitivement, elle doit créer les états accessibles à partir de l'état Δ auquel elle s'applique. En réalité, elle crée des nouveaux pre-états, qui seront analysés par **SR** afin de devenir des véritables états.

Pour s'appliquer à un état Δ , **Next** **arrange d'abord toutes les formules successeur de Δ en une liste L** où toutes les formules successeur positives ($\langle\langle A \rangle\rangle \bigcirc F$) précèdent toutes les formules successeur négatives ($\neg\langle\langle A \rangle\rangle \bigcirc G$).

Idée Générale : une action d'un agent sera ici le choix d'une formule de L . Cette action est codée par un nombre correspondant à la position de cette formule dans L .

Soit Δ un état du tableau. Faire (σ abrège $\sigma_{\mathbb{A}}$) :

1. Créer une liste $\mathbf{L}$ des formules successeur de Δ telle que toutes les positives précèdent toutes les négatives :

$\mathbf{L} =$

$$[\langle\langle A_0 \rangle\rangle \circ F_0, \dots, \langle\langle A_{m-1} \rangle\rangle \circ F_{m-1}, \neg\langle\langle A'_0 \rangle\rangle \circ G_0, \dots, \neg\langle\langle A'_{l-1} \rangle\rangle \circ G_{l-1}]$$

Soit $r_{\Delta} = m + l$ (c'est la taille de $\mathbf{L} + 1$) .

Soit $D(\Delta)$ l'ensemble de vecteurs d'actions $\{0, \dots, r_{\Delta} - 1\}^{|\mathbb{A}|}$.

Pour chaque vecteur $\sigma \in D(\Delta)$, soit :

- $N(\sigma) = \{i \mid \sigma_i \geq m\}$, où σ_i est la i -ème composante de σ ,
- $\text{co}(\sigma) = [\sum_{i \in N(\sigma)} (\sigma_i - m)] \bmod l$.

2. Pour chaque vecteur $\sigma \in D(\Delta)$ créer un **pre-état** Γ_σ :

$$\Gamma_\sigma =$$

$$\{F_p \mid \langle\langle A_p \rangle\rangle \circ F_p \in \Delta \text{ et } \sigma_a = p \text{ pour chaque agent } a \text{ de la coalition } A_p \}$$

$$\cup$$

$$\{\neg G_q \mid \neg\langle\langle A'_q \rangle\rangle \circ G_q \in \Delta, \text{co}(\sigma) = q \text{ et } \mathbb{A} \setminus A'_q \subseteq N(\sigma)\}$$

Si Γ_σ est vide, y ajouter T .

Puis, connecter Δ à Γ_σ par l'arête $\xrightarrow{\sigma}$.

Dans le cas particulier où le noeud Γ_σ existait déjà, **ne pas en créer une seconde copie** mais lier juste Δ à ce noeud par $\xrightarrow{\sigma}$ (**loop check**).

Soit $F = (\neg\langle\langle 1 \rangle\rangle \Box p \wedge \langle\langle 1, 2 \rangle\rangle \bigcirc p) \wedge \neg\langle\langle 2 \rangle\rangle \bigcirc \neg p$.

Soit Δ_2 l'état déjà créé :

$$\Delta_2 = \{F, \neg\langle\langle 1 \rangle\rangle \Box p, \langle\langle 1, 2 \rangle\rangle \bigcirc p, \neg\langle\langle 2 \rangle\rangle \bigcirc \neg p, \neg\langle\langle 1 \rangle\rangle \bigcirc \langle\langle 1 \rangle\rangle \Box p\}.$$

$$\mathbf{L} = [\underbrace{\langle\langle 1, 2 \rangle\rangle \bigcirc p}_0, \underbrace{\neg\langle\langle 2 \rangle\rangle \bigcirc \neg p}_1, \underbrace{\neg\langle\langle 1 \rangle\rangle \bigcirc \langle\langle 2 \rangle\rangle \Box p}_2].$$

Ici : $m = 1$, $l = 3$, $r_{\Delta_2} = m + l = 4$, $r_{\Delta_2} - 1 = 3$.

En bas : le numéro d'une formule (successeur) dans $\mathbf{L}$.

En haut : le numéro d'une formule négative dans la sous-liste des formules négatives.

Règle **Next**, Suite Exemple

$$\mathbf{L} = [\underbrace{\langle\langle 1, 2 \rangle\rangle \bigcirc p}_0, \underbrace{\neg\langle\langle 2 \rangle\rangle \bigcirc \neg p}_1, \underbrace{\neg\langle\langle 1 \rangle\rangle \bigcirc \langle\langle 1 \rangle\rangle \Box p}_2].$$

$D(\Delta_2)$ contient $3^2 = 9$ vecteurs : le joueur 1 peut choisir parmi 3 actions (= formules), et de même pour le joueur 2.

Pour chaque $\sigma \in D(\Delta_2)$, $N(\sigma)$ ($= \{i \mid \sigma_i \geq m\}$) est l'ensemble des positions de σ correspondant à des "actions négatives" (il nous dit quels joueurs ont joué négatif dans σ)

Exemples (les numéros des formules, ici, sont ceux de l'énumération principale, celle de $\mathbf{L}$)

- Si $\sigma = \langle 0, 1 \rangle$: le joueur 1 joue la formule 0 (positive), le joueur 2 la formule 1 (négative), et $N(\sigma) = \{2\}$.
- Si $\sigma = \langle 1, 2 \rangle$: le joueur 1 joue la formule 1 (négative), le joueur 2 la formule 2 (négative), et $N(\sigma) = \{1, 2\}$.

Règle **Next**, Suite Exemple

$$\mathbf{L} = [\underbrace{\langle\langle 1, 2 \rangle\rangle \bigcirc p}_0, \underbrace{\neg\langle\langle 2 \rangle\rangle \bigcirc \neg p}_1, \underbrace{\neg\langle\langle 1 \rangle\rangle \bigcirc \langle\langle 1 \rangle\rangle \Box p}_2].$$

$m = 1$, $l = 3$. Rappel de la déf. de $\text{co}(\sigma)$:

$$\text{co}(\sigma) = [\sum_{i \in N(\sigma)} (\sigma_i - m)] \bmod l.$$

Si $\sigma = \langle 1, 1 \rangle$, alors $N(\sigma) = \{1, 2\}$ et

$$\text{co}(\sigma) = [\sigma_1 - 1 + \sigma_2 - 1] \bmod 3 = [0 + 0] \bmod 3 = 0.$$

Si $\sigma = \langle 0, 1 \rangle$, alors $N(\sigma) = \{2\}$ et .

$$\text{co}(\sigma) = [\sigma_2 - 1] \bmod 3 = [1 - 1] \bmod 3 = [0] \bmod 3 = 0$$

La valeur $\text{co}(\sigma)$ est la somme des numéros des actions négatives (selon l'énumération propre à la sous-liste des actions négatives) effectuées selon le vecteur d'actions σ .

Règle **Next**, Suite Exemple

$$L = [\underbrace{\langle\langle 1, 2 \rangle\rangle \bigcirc p}_0, \underbrace{\neg\langle\langle 2 \rangle\rangle \bigcirc \neg p}_1, \underbrace{\neg\langle\langle 1 \rangle\rangle \bigcirc \langle\langle 1 \rangle\rangle \Box p}_2].$$

$$m = 1, l = 3.$$

Si $\sigma = \langle 0, 1 \rangle$, alors $N(\sigma) = \{2\}$ et $\text{co}(\sigma) = 0$

$$\Gamma_{\langle 0, 1 \rangle} = \emptyset \cup \emptyset \cup T.$$

Pourquoi ? NB : ici, $\mathbb{A} \setminus \{2\} = \{1\} \not\subseteq N(\sigma)$.

Si $\sigma = \langle 1, 1 \rangle$, alors $N(\sigma) = \{1, 2\}$ et $\text{co}(\sigma) = 0$.

$$\Gamma_{\langle 1, 1 \rangle} = \emptyset \cup \{\neg\neg p\} = \{p\}$$

Pourquoi ?

Si $\sigma = \langle 0, 0 \rangle$, alors $N(\sigma) = \emptyset$ et $\text{co}(\sigma) = 0$.

$$\Gamma_{\langle 0, 0 \rangle} = \{p\} \cup \emptyset = \{p\}.$$

Pourquoi ? (NB : ici, $\mathbb{A} \setminus \{2\} = \{1\} \not\subseteq N(\sigma)$).

Règle **Next**, Suite Exemple

$$L = [\underbrace{\langle\langle 1, 2 \rangle\rangle \bigcirc p}_0, \underbrace{\neg\langle\langle 2 \rangle\rangle \bigcirc \neg p}_1, \underbrace{\neg\langle\langle 1 \rangle\rangle \bigcirc \langle\langle 1 \rangle\rangle \Box p}_2].$$

σ	$N(\sigma)$	$\text{co}(\sigma)$	Γ_σ
0,0	$\emptyset$	0	$\{p\}$
0,1	$\{2\}$	0	$\{T\}$
0,2	$\{2\}$	1	$\{\neg\langle\langle 1 \rangle\rangle \Box p\}$
1,0	$\{1\}$	0	$\{p\}$
1,1	$\{1, 2\}$	0	$\{p\}$
1,2	$\{1, 2\}$	1	$\{\neg\langle\langle 1 \rangle\rangle \Box p\}$
2,0	$\{1\}$	1	$\{T\}$
2,1	$\{1, 2\}$	1	$\{\neg\langle\langle 1 \rangle\rangle \Box p\}$
2,2	$\{1, 2\}$	0	$\{p\}$

Exemple d'application de **Next**

Suite de la construction du tableau pour tester la satisfiabilité de F ,
avec d'abord une application de **Next**, à Δ_2 , puis une
application de **Next**, à Δ_1 :



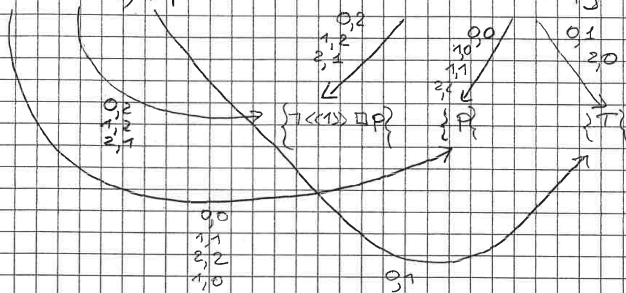
Exemple d'application de Next

$$F = \neg \langle \langle 1 \rangle \rangle \sqcap p \wedge \langle \langle 1, 2 \rangle \rangle op \wedge \neg \langle \langle 2 \rangle \rangle op \neg p$$

$$\Gamma_1 = \{F\}$$

$$\Delta_1 = \{F, \neg \langle \langle 1 \rangle \rangle \sqcap p, \langle \langle 1 \rangle \rangle op, \neg \langle \langle 2 \rangle \rangle op \neg p\}$$

$$\Delta_2 = \{F, \neg \langle \langle 1 \rangle \rangle \sqcap p, \langle \langle 1 \rangle \rangle op, \neg \langle \langle 2 \rangle \rangle op \neg p\}$$



Pourquoi **Next** fonctionne ?

La construction de **Next** assure les propriétés suivantes, pour chaque pre-état Γ successeur d'un état Δ :

- ▶ **P1** : Si $\{\langle A_i \rangle F_i, \langle A_i \rangle F_j\} \subseteq \Delta$ et $\{F_i, F_j\} \subseteq \Gamma$ (avec $i \neq j$), alors $A_i \cap A_j = \emptyset$.
- ▶ **P2** : Γ contient au max une formule $\neg G$ telle que $\langle A' \rangle G \in \Delta$.
- ▶ **P3** : Si $\{\langle A_i \rangle F_i, \neg \langle A' \rangle G\} \subseteq \Delta$ et $\{F_i, \neg G\} \subseteq \Gamma$, alors $A_i \subseteq A'$ (donc $(A \setminus A') \cap A_i = \emptyset$).

Pourquoi ? En trouver les raisons : **Exercice 3**

Pourquoi **Next** fonctionne ? Suite

P1 permet d'aller de Δ à Γ par le même vecteur d'actions σ : pas d'incohérence possible entre l'action choisie par l'agent a car il appartient à la coalition A_i et celle choisie par a car il appartient à A_j .

P2 assure que les formules négatives de Δ sont distribuées parmi les Γ_i successeurs, une par successeur : chacune peut être vérifiée, sans problèmes de conflits.

P3 sert à expliquer la gestion des formules négatives (la moins intuitive !) 

Pourquoi **Next** fonctionne ? Suite

Si $\neg \langle\langle A_q \rangle\rangle G_q \in \Delta$, $\Delta \xrightarrow{\sigma} \Gamma$ et $\neg G_q \in \Gamma$, alors la coalition adverse à A_q pourra riposter à l'action de A dans σ de façon à assurer $\neg G_q$.

Car :

- Par construction de **Next**, on sait que $q = \text{co}(\sigma)$;
- Par construction de **Next**, on sait aussi que tous les agents de $\mathbb{A} \setminus A_q$ jouent négatif ;

Alors, si la somme des actions négatives de A_q dans σ vaut k_1 , la coalition adverse peut riposter ainsi : un joueur b joue un k_2 tel que $k_1 + k_2$ vaut q , et tous les autres (s'il en a) jouent l'action négative 0. Ce choix assure que $\text{co}(\sigma) = q$, donc que σ code une façon de jouer qui produit $\neg G_q$. Cette façon de jouer de la coalition adverse de A_q est possible grâce à P3, sinon il y aurait risque de conflit avec les actions des coalitions A_i jouant positif responsables de la présence des F_i dans Γ !

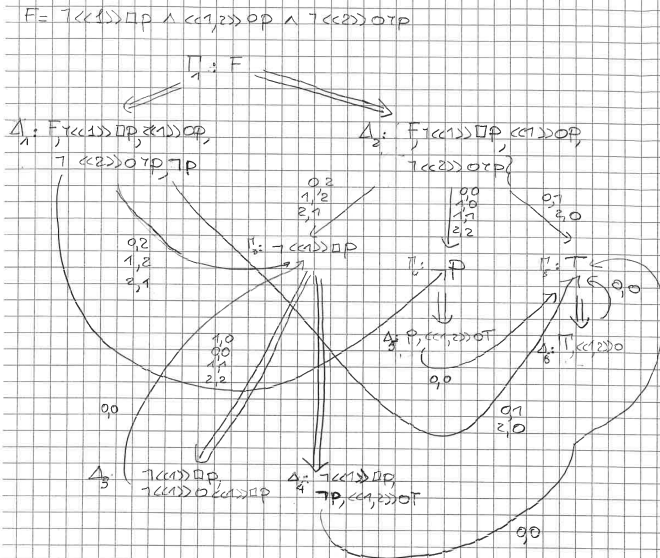
- NB : on peut toujours compléter une action quelconque de A_q de façon à obtenir au moins un σ tel que $\text{co}(\sigma) = q$.

Intuitivement, $co(\sigma)$ est une sorte de **vote collectif** des adversaires de la coalition A_q de $\neg \langle\langle A_q \rangle\rangle G_q$:
chacun fait d'abord son propre choix d'une formule négative, puis il s'accordent pour décider que la formule collectivement choisie pour riposter à l'action de A_q est celle dont le numéro est la somme de leur actions individuelles (modulo I).

Résultat de la phase de construction pour tester la satisfiabilité de F :



Tableau construit pour F



La phase de construction produit un graphe fini, noté $\mathcal{T}^F_0$ et dit *tableau initial pour F*..

Dans la phase d'élimination on applique à tour de rôle deux règles, **ER1** et **ER2**, afin d'éliminer certain états (sommets du tableau) qui ne pourront jamais être des états d'un modèle de F .

Chaque étape s'applique à un graphe $\mathcal{T}^F_n$ ($n \geq 0$) pour créer un nouveau graphe $\mathcal{T}^F_{n+1}$. Mais la procédure termine.

Soit N un sommet de $\mathcal{T}_n^F$ (état ou pre-état).

- ▶ Si N est un pre-état : si tous les sommets Δ_i tels que $N \Rightarrow \Delta_i$ ont été déjà éliminés, éliminer N aussi ;

Intuition : on a éliminé tous les états possibles issus de N : par exemple, N contient $A \vee B$ mais on a déjà éliminé la possibilité A et la B .

- ▶ Si N est un état : si pour quelque σ le sommet Γ_i tel que $N \xrightarrow{\sigma} \Gamma_i$ a été déjà éliminé, éliminer N aussi.

Intuition : par définition de CGM, tout vecteur possible doit être joué à l'état N .

Vers la règle **ER2** : *Eventualities*

Les formules de la forme $\langle\langle A \rangle\rangle F_1 U F_2$ ou de la forme $\neg\langle\langle A \rangle\rangle \Box G$ sont dites *eventualities* : elles font des promesses.

$\langle\langle A \rangle\rangle F_1 U F_2$ promet que F_2 sera vrai, tôt ou tard.

$\neg\langle\langle A \rangle\rangle \Box G$ promet que $\neg G$ sera vrai, tôt ou tard.

Danger dans la construction du tableau : **toujours procrastiner** : on promet, on promet, on promet... Et on boucle.

Intuition sur le rôle de la règle **ER2 : contrôler que toute promesse est tenue**

Vers la règle **ER2** : Notation

Soit Δ un état de $\mathcal{T}_n^F$ et soit

$\mathbf{L} =$

$[\langle\langle A_0 \rangle\rangle \circ F_0, \dots, \langle\langle A_{m-1} \rangle\rangle \circ F_{m-1}, \neg\langle\langle A'_0 \rangle\rangle \circ G_0, \dots, \neg\langle\langle A'_{l-1} \rangle\rangle \circ G_{l-1}]$

la liste de ses formules successeurs créé pour pouvoir appliquer la règle **Next**.

$Succ(\Delta, \langle\langle A_p \rangle\rangle \circ F_p) =$

$\{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma \text{ et } \sigma_a = p \text{ pour chaque agent } a \text{ de la coalition } A_p \}$

$Succ(\Delta, \neg\langle\langle A'_p \rangle\rangle \circ G_p) =$

$\{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma, \text{ co}(\sigma) = q \text{ et } \mathbb{A} \setminus A'_q \subseteq N(\sigma)\}$

C'est à dire :

$Succ(\Delta, \langle\langle A_p \rangle\rangle \circ F_0)$ est l'ensemble des successeurs de Δ créés (par **Next**) car $\langle\langle A_p \rangle\rangle \circ F_p \in \Delta$

$Succ(\Delta, \neg\langle\langle A'_p \rangle\rangle \circ G_p)$ est l'ensemble des successeurs de Δ créés (par **Next**) car $\neg\langle\langle A'_p \rangle\rangle \circ G_p \in \Delta$.

Vers la règle ER2 : Réalisation

Intuition : on dit qu'une *eventuality* Ev ($\langle\langle A \rangle\rangle F_1 U F_2$ ou $\neg\langle\langle A \rangle\rangle \Box G$) est *réalisée*, à un sommet du tableau courant, quand sa promesse est tenue, soit tout de suite soit après.

La définition formelle de réalisation de Ev à l'état Δ (de $\mathcal{T}^F_n$) est récursive :

1. • *Base* : $\langle\langle A \rangle\rangle F_1 U F_2$ est réalisée à Δ si $\{\langle\langle A \rangle\rangle F_1 U F_2, F_2\} \subseteq \Delta$
• *Cas Récursif* : $\langle\langle A \rangle\rangle F_1 U F_2$ est réalisée à Δ si $\{\langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle F_1 U F_2, F_1\} \subseteq \Delta$ et, pour tout $\Gamma \in Succ(\Delta, \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle F_1 U F_2)$ il existe un état Δ' (de $\mathcal{T}^F_n$) tel que $\Gamma \Rightarrow \Delta'$ et $\langle\langle A \rangle\rangle F_1 U F_2$ est réalisée à Δ' .
2. • *Base* : $\neg\langle\langle A \rangle\rangle \Box G$ est réalisée à Δ si $\{\neg\langle\langle A \rangle\rangle \Box G, \neg G\} \subseteq \Delta$
• *Cas Récursif* : $\neg\langle\langle A \rangle\rangle \Box G$ est réalisée à Δ si $\{\neg\langle\langle A \rangle\rangle \Box G, \neg\langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \Box G\} \subseteq \Delta$ et pour tout $\Gamma \in Succ(\Delta, \neg\langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \Box G)$ il existe un état Δ' (de $\mathcal{T}^F_n$) tel que $\Gamma \Rightarrow \Delta'$ et $\neg\langle\langle A \rangle\rangle \Box G$ est réalisée à Δ' .

Si Δ est un état de $\mathcal{T}_n^F$ qui contient au moins une *eventuality* qui n'est pas réalisée à Δ , alors supprimer Δ , en obtenant ainsi $\mathcal{T}_{n+1}^F$

Elimination : Exemple 1

Le tableau pour F issu de la phase de construction de notre exemple (celui de la diapositive 65) **n'est pas modifié du tout par l'Elimination**. La seule *eventuality* est $\neg\langle\langle 1 \rangle\rangle\Box p$ et

- ▶ $\neg p \in \Delta_1$ et $\neg p \in \Delta_4$: réalisation immédiate à ces états ;
- ▶ $\neg\langle\langle 1 \rangle\rangle\Box p \notin \Delta_5$ et $\neg\langle\langle 1 \rangle\rangle\Box p \notin \Delta_6$: ces états ne contiennent pas d' *eventuality* ;
- ▶ : $\neg\langle\langle 1 \rangle\rangle\Box p \in \Delta_3$, mais son seul pre-état successeur contenant $\neg\langle\langle 1 \rangle\rangle\Box p$ c'est Γ_3 , qui a Δ_4 comme $\Rightarrow$ -successeur, où il y a réalisation immédiate. De même, $\neg\langle\langle 1 \rangle\rangle\Box p \in \Delta_2$, mais son seul pre-état successeur contenant $\neg\langle\langle 1 \rangle\rangle\Box p$ c'est Γ_3 .

↪ **ER2** n'efface rien, donc **ER1** non plus.

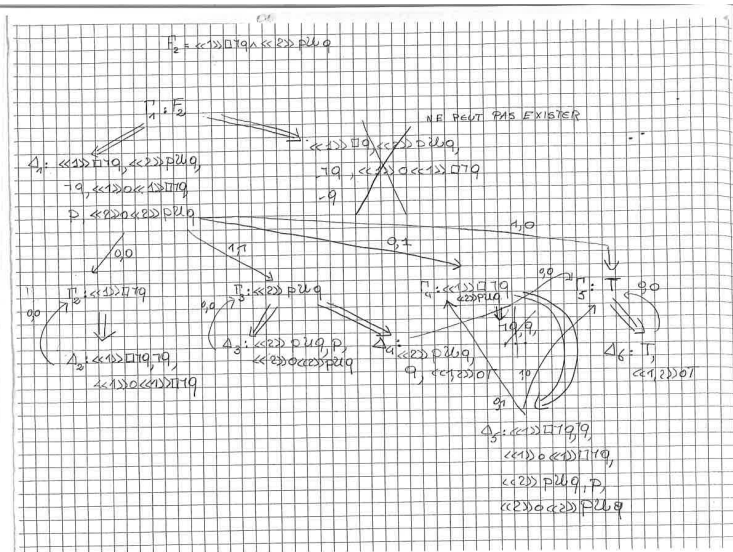
Elimination : Exemple 2

$$F_2 = \langle\langle 1 \rangle\rangle \Box \neg q \wedge \langle\langle 2 \rangle\rangle p \cup q$$

Le tableau pour F_2 issu de la phase de Construction :



Elimination : Suite Exemple 2



Elimination : Suite Exemple 2

Dans le tableau construit pour F_2 :

- ▶ L'état Δ_5 contient l'*eventuality* $\langle\langle 2 \rangle\rangle pUq$ mais son seul pre-état successeur contenant $\langle\langle 2 \rangle\rangle pUq$ est Γ_4 , dont le seul état qui en est un $\Rightarrow$ -successeur est Δ_5 à nouveau. **Cycle, procrastination pour toujours!** Δ_5 est supprimé par **ER2**.
- ▶ Le pre-état Γ_4 a comme unique successeur Δ_5 , qui a été éliminé. Γ_4 est supprimé par **ER1**.
- ▶ L'état Δ_1 est tel que son pre-état successeur selon $\langle 0, 1 \rangle$ est Γ_4 qui a été éliminé. Δ_1 est supprimé par **ER1**. Par conséquent, la racine Γ_0 est éliminée aussi par **ER1**.

N.B : Peu importe ce qui arrive aux autres sommets : la racine est supprimée et F_2 est déclarée insatisfiable ! (voir après).

Après q'un tableau pour une formule F a été constuit, et que la phase d'élimination a eu lieu :

Si la racine a été supprimée, le tableau est dit *clos* et F est déclarée *insatisfiable*.

Sinon, le tableau est dit *ouvert* et F est déclarée satisfiable.

Terminaison et Complexité : la procédure termine en temps exponentiel (par rapport à la taille de F).

Idée de la preuve : On démontre que un tableau pour F a au maximum autant de sommets que l'ensemble des sous-ensembles de la cloture de F , et que la cardinalité de la cloture de F est un polynôme de la taille de F .

Correction par rapport à l'insatisfiabilité : Si un tableau pour F est clos, alors F n'est pas satisfiable.

Complétude par rapport à l'insatisfiabilité : Si F n'est pas satisfiable, alors tout tableau pour F est clos.

Idée de la preuve : On démontre (V. Goranko, D. Shkatov, 2009) : si un tableau pour F est ouvert, alors il existe une méthode pour **extraire** un modèle de F à partir du tableau. *L'état où F sera vraie sera un des états issus de l'analyse du pre-état racine ($\{F\}$).*

NB. Ce n'est pas vrai que tout sous-graphe sera un modèle !

- Le graphe-tableau lui même, déjà est non-déterministe, car il décrit **plusieurs cgm candidats** ; la même chose peut arriver pour un sous-graphe.
- Puis, même si la structure d'un sous-graphe est ok, il faut assurer que toutes les *eventualities* soient réalisées.
- Ce n'est pas vrai non plus que cette méthode assure un modèle minimal par rapport au nombre d'états.

TATL : les tableaux pour ATL implementés

1. Aller à la page

http://atila.ibisc.univ-evry.fr/tableau_ATL/index.php

2. Regarder ce que l'application permet de faire.

Warning :

- 1) Cette implementation affiche aussi des états banalement incohérents (par ex. contenant p et $\neg p$), pour ensuite les supprimer.

- 2) Quand une formule est traitée, on affiche à gauche le graphe contenant pre-états et états, en l'appelant *pretableau* : les P_i sont des pre-états, les S_i des états.

A droite, on affiche le graphe épuré des pre-états, si la formule est satisfiable, et rien du tout sinon.

- En utilisant TATL, tester la satisfiabilité des formules suivantes :

1. $\langle\langle 1 \rangle\rangle p \cup q$
2. $\neg \langle\langle 1 \rangle\rangle p \cup q$
3. $\langle\langle 1 \rangle\rangle \bigcirc p \wedge \langle\langle 1 \rangle\rangle \bigcirc \neg p$
4. $\langle\langle 1 \rangle\rangle \Box p \wedge \langle\langle 1 \rangle\rangle \Box \neg p$
5. $\langle\langle 1, 2 \rangle\rangle p \cup (\neg \langle\langle 1 \rangle\rangle \Box p)$

- Quand la formule testée est satisfiable, dessiner le graphe des états décrit dans la droite de la page et, à partir de la, indiquer au moins un modèle de la formule.

(Utiliser ce graphe comme base, et puis l'intuition, car on ne vous a pas donné l'algorithme d'extraction de modèles de la preuve de complétude).