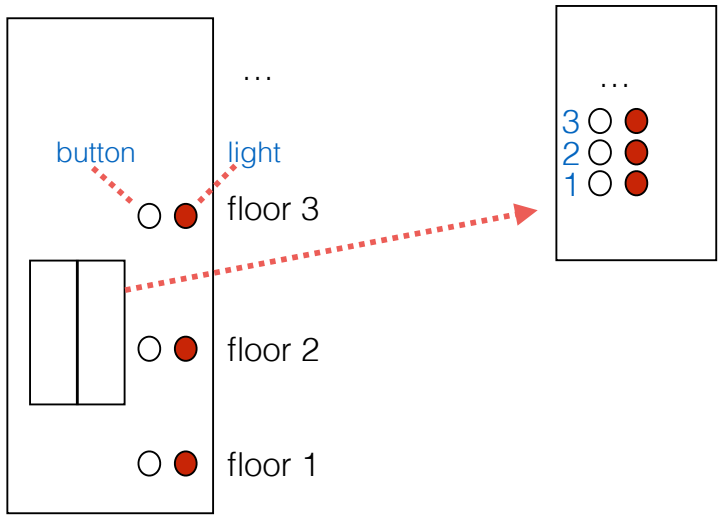
Spécification d'un ascenseur

Specification of a lift



H. Barringer ("Up and down, the temporal way", 1985)

Specification of a lift

Hypothesis:

- A floor door is open or closed.
- A button is pressed or depressed.
- An indicator light is on or off.
- The cabin is present at floor i, or it is absent.

Specification of a lift

P1. Safe doors:
A floor door is never opened if the cabin is not present at the given floor.

P2. Indicator lights:
The indicator lights correctly reflect the current requests.

P3. Services:
All requests are eventually satisfied.

P4. Smart service:
The cabin only services the requested floors and does not move when there is no request.

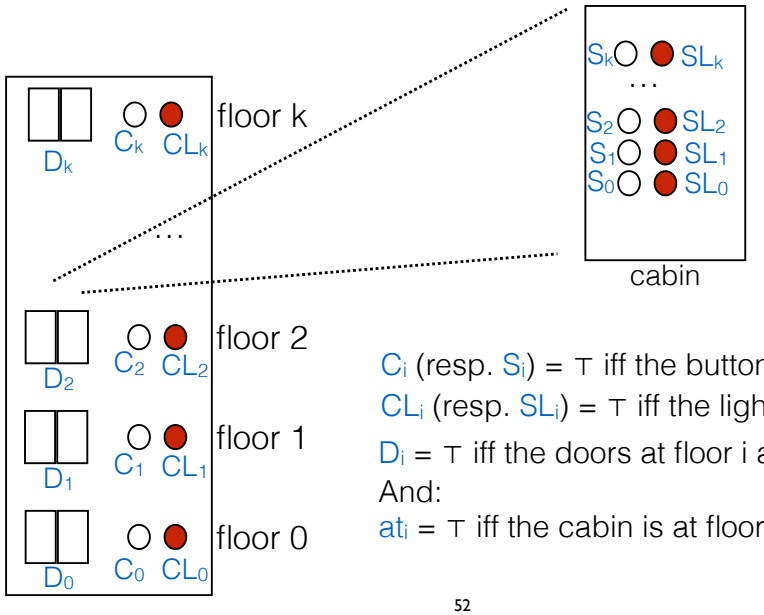
Specification of a lift

P5. Diligent service:
The cabin does not pass by a floor for which it has a request without servicing it.

P6. Direct movements:
The cabin always moves directly from previous to next serviced floor.

P7. Priorities:
The cabin services in priority requests that do not imply a change of direction (upward or downward).

Specification of a lift: the atomic prop.



C_i (resp. S_i) = $\top$ iff the button is pressed
CL_i (resp. SL_i) = $\top$ iff the light is on
D_i = $\top$ iff the doors at floor i are opened
And:
at_i = $\top$ iff the cabin is at floor i

Specification of a lift

P1. Safe doors:
A floor door is never opened if the cabin is not present at the given floor.

$$\mathbf{G}\,(\,D_i \Rightarrow at_i)$$

Specification of a lift

P2. Indicator lights:
The indicator lights correctly reflect the current requests.

$\bigwedge_i \mathbf{G}\,(\,C_i \Rightarrow (LC_i \vee servicing_i\,))$: turned on when necessary

with: $servicing_i = at_i \wedge D_i$

$\bigwedge_i \mathbf{G}\,(\,LC_i \Rightarrow (\,LC_i\ \mathbf{W}\ servicing_i\,)\,)$: stay lit when necessary

$\bigwedge_i \mathbf{G}\,(\,servicing_i \Rightarrow (\neg\, CL_i \wedge \neg\, CS_i))$: turned off when necessary

$\bigwedge_i \mathbf{G}\,(\,(\neg\, LC_i\,) \Rightarrow (\,(\neg LC_i)\ \mathbf{W}\ C_i)\,)$:only turned on when necessary

or $\bigwedge_i \mathbf{G}\,(\,LC_i \Rightarrow (\,LC_i\ \mathbf{S}\ C_i)\,)$

(and the same for S_i and SL_i)

Specification of a lift

P2. Indicator lights:
The indicator lights correctly reflect the current requests.

An *alternative* is:

$$\bigwedge_i \mathbf{G}\,(\,LC_i \Leftrightarrow (\,(\neg servicing_i)\;\mathbf{S}\;(C_i \wedge \neg\, servicing_i))\,)$$

(and the same for S_i and SL_i)

Specification of a lift

P3. Services:
All requests are eventually satisfied.

$$\bigwedge_i \mathbf{G}\,(\,request_i \Rightarrow \mathbf{F}\; servicing_i\,))$$

with: $request_i = C_i \vee S_i$

P4. Smart service:
The cabin only services the requested floors and does not move when there is no request.

$$\bigwedge_i \mathbf{G}\,(\;servicing_i \Rightarrow [\;servicing_i\;\mathbf{S}\;(CL_i \vee SL_i)\;]\,)$$

$$\bigwedge_i \mathbf{G}\,(\;at_i \Rightarrow (\,at_i\;\mathbf{W}\;(\;\bigvee_{j\neq i}(CL_j \vee SL_j)\;)\;)\,)$$

Specification of a lift

P5. Diligent service:
The cabin does not pass by a floor for which it has a request without servicing it.

$$\bigwedge_i \mathbf{G}\,([(\,LC_i \vee LS_i) \wedge at_i\,] \Rightarrow (at_i\;\mathbf{U}\;servicing_i\,))$$

Specification of a lift

P6. Direct movements:
The cabin always moves directly from previous to next serviced floor.

$$\bigwedge_{i<j} \mathbf{G}\,(\,From\_i\_to\_j \Rightarrow (at_i \vee betw\_floors)\;\mathbf{U}\;(at_{i+1} \wedge (at_{i+1} \vee betw\_floors)\;\mathbf{U}\;(at_{i+2}\ldots(at_{j-1} \wedge (at_{j-1} \vee betw\_floors)\;\mathbf{U}\;at_j))))$$

$\neg at_0 \wedge \ldots \wedge \neg at_n$

$servicing_i \wedge [\,(servicing_i \vee \neg\, service)\;\mathbf{U}\;servicing_j\,]$

$service = servicing_0 \vee servicing_1 \vee \ldots \vee servicing_k$

Specification of a lift

P7. Priorities:
The cabin services in priority requests that do not imply a change of direction (upward or downward).

$$Up = \bigvee_{i=1..k} [\ (at_i \vee betw\_floors)\ \mathbf{S}\ at_{i-1} \ \wedge\ (at_i \vee betw\_floors)\ \mathbf{U}\ at_i\ ]$$

$$Down = \bigvee_{i=0..k-1} [\ (at_i \vee betw\_floors)\ \mathbf{S}\ at_{i+1} \ \wedge\ (at_i \vee betw\_floors)\ \mathbf{U}\ at_i\ ]$$

$$\mathbf{G} \bigwedge_{i=0..k-1} [(servicing_i \wedge Down \wedge \bigvee_{j<i} (CL_j \vee SL_j)) \Rightarrow \bigvee_{n<i} From\_i\_to\_n]$$

$$\mathbf{G} \bigwedge_{i=1..k} [(servicing_i \wedge Up \wedge \bigvee_{j>i} (CL_j \vee SL_j)) \Rightarrow \bigvee_{n>i} From\_i\_to\_n]$$

LTL sans opérateurs du passé

Pour cette partie, on va considérer une variante de LTL où il n'y a ni Since, ni F^{-1}, ni X^{-1}.

Pourquoi ? pour simplifier un peu la suite…
sans rien perdre sur le fond:

- les mêmes techniques marchent pour LTL avec passé…
- les opérateurs du passé sont pratiques pour exprimer des propriétés mais pas indispensables: on peut toujours se débrouiller avec Until et X.

Débrouiller ? Toute formule de LTL avec passé est équivalente à une formule sans passé lorsqu'on les interprète au début d'une exécution.

Procédures de décision

LTL sans opérateurs du passé

Simplifier ?

Syntaxe:

$$\phi,\psi ::= P \mid \neg\phi \mid \phi\vee\psi \mid \phi\wedge\psi \mid \mathbf{X}\ \phi \mid \psi\mathbf{U}\phi$$

On interprète désormais les formules de LTL sur une exécution ρ d'un STE… (sans position !)

$\rho \models \mathbf{X}\ \phi$ ssi $\rho^1 \models \phi$

$\rho \models \psi\mathbf{U}\phi$ ssi ($\exists\ i\geq 0$. ($\rho^i \models \phi$ et $\forall\ 0\leq j<i$ on a $\rho^j \models \psi$)

ρ^i est le i-ème suffixe: $\rho(i)\rho(i+1)$…

Une histoire de mots !

$\mathbf{S}$ = (Q, Act, →, q_0, AP, L)

Lorsqu'on travaille avec LTL, $\mathbf{S}$ est vu comme un ensemble d'exécutions étiquetées:

$\rho : q_0 \rightarrow q_1 \rightarrow \ldots$ + $L: Q \rightarrow 2^{AP}$

$\in Q^\omega$

En fait, on n'utilise les noms des états que pour utiliser la fonction d'étiquetage L et obtenir les propositions atomiques associées à chaque état de ρ.

ρ + L peuvent être vues comme une « trace », un mot infini sur l'alphabet 2^{AP}.

Ex:
ρ: $q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_1 \rightarrow \ldots$ et $L(q_0)=\{a\}$, $L(q_1)=\{b,c\}$
La trace est le mot {a}{b,c}{a}{b,c}{a}{b,c}{a}{b,c}{a}{b,c}{a}{b,c}…

Et les formules de LTL s'interprète presque de la même manière:
$\rho,i \models P \Leftrightarrow P \in \rho(i)$ remplace $\rho,i \models P \Leftrightarrow P \in L(\rho(i))$

Une histoire de mots !

Une formule ϕ de LTL décrit une propriété le long d'un mot infini sur l'alphabet 2^{AP}.

Les modèles de ϕ de LTL sont l'ensemble des mots où ϕ est vraie.

On note mod(ϕ) les modèles de ϕ de LTL.

Donc mod(ϕ) = les mots infinis sur l'alphabet 2^{AP} vérifiant ϕ.

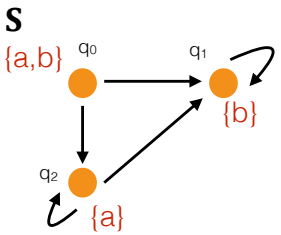
mod(ϕ) est donc aussi un **langage** !

Une histoire de mots !

$\mathbf{S}$ = (Q, Act, →, q_0, AP, L)

Donc $\mathbf{S}$ est vu un ensemble de mots.

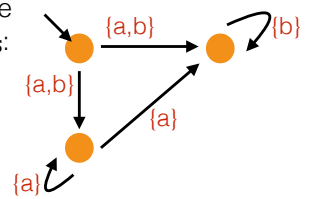
Donc $\mathbf{S}$ est vu comme un **langage** → Traces($\mathbf{S}$)



Exec: $q_0.q_1^\omega \cup q_0.q_2^\omega \cup q_0.q_2^+.q_1^\omega$

Langage des traces:
$\{a,b\}.\{b\}^\omega \cup \{a,b\}.\{a\}^\omega \cup \{a,b\}.\{a\}^+.\{b\}^\omega$

Un automate de mots infinis $\mathcal{A}_S$:



Traces($\mathbf{S}$) = $\mathcal{L}(\mathcal{A}_S)$

Problèmes de vérification

Model-checking:
input: un modèle (STE) $\mathbf{S}$ et une formule φ
output: oui ssi $\mathbf{S} \models \varphi$.

Satisfaisabilité:
input: une formule φ
output: oui ssi il existe un modèle $\mathbf{S}$ t.q. $\mathbf{S} \models \varphi$.
(+ $\mathbf{S}$ si il existe !))

Synthèse de contrôleur:
input: un modèle partiel $\mathbf{S}$ une formule φ
output: un « controleur » $\boldsymbol{C}$ t.q. $\mathbf{S} \times \boldsymbol{C} \models \varphi$.

Problèmes de vérification pour LTL

Quel lien entre Traces($\mathbf{S}$) et mod(ϕ) ?

1) Traces($\mathbf{S}$) $\subseteq$ mod(ϕ) $\quad\quad \mathbf{S} \models \phi$

2) Traces($\mathbf{S}$) $\cap$ mod(ϕ) = $\varnothing$ $\quad\quad \mathbf{S} \models \neg\, \phi$

3) sinon $\quad\quad \mathbf{S} \not\models \phi$, $\mathbf{S} \not\models \neg\phi$

$\text{mod}(\neg\phi) = (2^{AP})^{\omega} \setminus \text{mod}(\phi)$

$(2^{AP})^{\omega}$ = ens. de tous les mots infinis sur l'alphabet 2^{AP}.

Rappel: $\mathbf{S} \models \phi \iff (\ \rho \models \phi \ \ \forall\, \rho \in \text{Exec}(\mathbf{S})\ )$

Satisfaisabilité de LTL

Comment tester si il existe un modèle pour ϕ ?

→ Tester mod(ϕ) est non vide.

C'est-à-dire tester si $\mathcal{L}(\mathcal{A}_\phi) \neq \varnothing$?

Construire les modèles de ϕ

Etant donnée ϕ, on sait construire un automate $\mathcal{A}_\phi$ qui reconnait les modèles de ϕ !

mod(ϕ) = $\mathcal{L}(\mathcal{A}_\phi)$

Pourquoi chercher des automates ? Parce que nous disposons de nombreux outils pour les manipuler (union, intersection, complément, inclusion, *etc.*) !

Model-checking de LTL

Comment tester si $\mathbf{S} \models \phi$?

Tester Traces($\mathbf{S}$) $\subseteq$ mod(ϕ) ?

C'est-à-dire tester si $\mathcal{L}(\mathcal{A}_S) \subseteq \mathcal{L}(\mathcal{A}_\phi)$?

Ou plutôt tester Traces($\mathbf{S}$) $\cap$ mod($\neg\phi$) = $\varnothing$
(donc tester si $\mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_{\neg\phi}) = \varnothing$)
car tester le vide est plus simple que tester l'inclusion de deux langages, et faire l'intersection est facile.

Satisfaisabilité et Model-checking de LTL

Les deux problèmes se ramènent donc aux deux questions suivantes:

- $\mathcal{L}(\mathcal{A}_\phi) \neq \varnothing$
- $\mathcal{L}(\mathcal{A}_S) \cap \mathcal{L}(\mathcal{A}_{\neg\phi}) = \varnothing$?

Tout repose sur les deux automates $\mathcal{A}_S$ et $\mathcal{A}_\phi$ (ou $\mathcal{A}_{\neg\phi}$). $\mathcal{A}_S$ ne pose pas de problème: il est facile à construire à partir de $\mathbf{S}$.
Et $\mathcal{A}_\phi$?

Construction de $\mathcal{A}_\phi$ - 1

Etant donnée une formule de LTL ϕ, on veut construire un automate $\mathcal{A}_\phi$ qui reconnait le langage mod(ϕ).

$\mathcal{A}_\phi = (Q, q_0, \rightarrow, \boldsymbol{\mathcal{F}})$ sera un automate de Büchi généralisé.
kesako ?

Un automate de Büchi reconnait des mots infinis: un mot w est accepté si il existe un chemin dans l'automate dont l'étiquetage correspond à w et si le chemin passe infiniment souvent par un des états acceptants (un sous-ensemble de Q).

Dans un automate de Büchi généralisé, les états acceptants sont donnés par un ensemble de sou-ensemble $\boldsymbol{\mathcal{F}}=\{\boldsymbol{\mathcal{F}}_1,\ldots,\boldsymbol{\mathcal{F}}_k\}$: un « bon » chemin doit passer infiniment souvent par un des états de chaque $\boldsymbol{\mathcal{F}}_i$…

Construction de $\mathcal{A}_\phi$ - 2

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \boldsymbol{\mathcal{F}})$

Chaque état de Q sera associé (défini) par un sous-ensemble de sous-formules de ϕ.
idée de la construction: depuis un état associé à l'ensemble de sous-formules $\{\psi_1,\ldots,\psi_n\}$, on reconnait des mots vérifiants chacune de ses sous-formules.

Comme $\mathcal{A}_\phi$ doit reconnaître les modèles de ϕ, l'ensemble des états initiaux Q_0 contiendra tous les états de Q contenant la sous-formule ϕ…

Construction de $\mathcal{A}_\phi$ - 3

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \boldsymbol{\mathcal{F}})$

Comment définir les états Q ? Quels ensembles de sous-formules de ϕ choisir ?

On va choisir des sous-ensembles cohérents (logiquement), maximaux et conforme à la sémantique de LTL.

Soit S_ϕ l'ensemble des sous-formules de ϕ et leur négation.

Exemple:
ϕ = a $\mathbf{U}$ ($\mathbf{X}$ b)
S_ϕ = {a, ¬a, b, ¬b, $\mathbf{X}$ b, ¬$\mathbf{X}$ b, a $\mathbf{U}$ ($\mathbf{X}$ b), ¬(a $\mathbf{U}$ ($\mathbf{X}$ b)) }

Construction de $\mathcal{A}_\phi$ - 4

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$

Comment définir les états Q ?

Les états q sont des sous-ensembles cohérents, maximaux et conforme à la sémantique de LTL…

‣ Cohérents:
Si $\psi_1 \wedge \psi_2 \in q$, alors ψ_1 et ψ_2 sont dans q, si $\psi_1 \vee \psi_2 \in q$, alors ψ_1 ou ψ_2 sont dans q, si $\psi \in q$, alors $\neg\psi \notin q$.

‣ Maximaux:
Dans tout état, pour chaque sous-formule ψ, on met soit ψ, soit $\neg\psi$.

‣ Conforme à la sémantique de LTL:
Dans tout état q, si la sous-formule $\psi_1 \mathbf{U} \psi_2$ est présente, alors on a soit ψ_1, soit ψ_2 dans l'état q.
Si $\psi_1 \mathbf{U} \psi_2 \in S\phi$, alors si ψ_2 est dans un état q, $\psi_1 \mathbf{U} \psi_2 \in q$

Et les états initiaux sont ceux contenant ϕ.

Construction de $\mathcal{A}_\phi$ - 5

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$

Comment définir les états Q ?

Ce sont des sous-ensembles cohérents, maximaux et conforme à la sémantique de LTL…

Exemple:
$\phi = a \mathbf{U} (b \wedge c)$
$S_\phi = \{a, \neg a, b, \neg b, c, \neg c, b \wedge c, \neg(b \wedge c), a \mathbf{U} (b \wedge c), \neg(a \mathbf{U} (b \wedge c)) \}$

$q = \{a, \neg b, c, \neg(b \wedge c), a \mathbf{U} (b \wedge c)\}$ ou
$q' = \{a, \neg b, c, \neg(b \wedge c), \neg (a \mathbf{U} (b \wedge c))\}$ sont ok !

Mais $r = \{a, b, \neg b, c, \neg(b \wedge c), a \mathbf{U} (b \wedge c)\}$,
$r' = \{a, b, c, \neg(b \wedge c), a \mathbf{U} (b \wedge c)\}$ ou
$r'' = \{a, b, c, (b \wedge c), \neg(a \mathbf{U} (b \wedge c))\}$ ne sont pas bien formés !

Construction de $\mathcal{A}_\phi$ - 6

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$

Comment définir les transitions de l'automate $\mathcal{A}_\phi$?

On met une transition $(q, \sigma, q') \in Q \times 2^{AP} \times Q$ si et seulement si:

- $\sigma = q \cap AP$ (*ie* les prop. atomiques de q)
- $\forall \mathbf{X}\psi \in S_\phi, \quad \mathbf{X}\psi \in q \iff \psi \in q'$
- $\forall \psi_1 \mathbf{U} \psi_2 \in S_\phi, \quad \psi_1 \mathbf{U} \psi_2 \in q \Leftrightarrow (\psi_2 \in q \vee (\psi_1 \in q \wedge \psi_1 \mathbf{U} \psi_2 \in q'))$

Exemple:
$\phi = a \mathbf{U} (b \wedge c)$
$S_\phi = \{a, \neg a, b, \neg b, c, \neg c, b \wedge c, \neg(b \wedge c), a \mathbf{U} (b \wedge c), \neg(a \mathbf{U} (b \wedge c)) \}$
$q = \{a, \neg b, c, \neg(b \wedge c), a \mathbf{U} (b \wedge c)\}$
$q' = \{\neg a, b, c, (b \wedge c), (a \mathbf{U} (b \wedge c))\}$

On a : $q \text{ —- } \{a,c\} \text{ ——> } q'$

Construction de $\mathcal{A}_\phi$ - 7

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$

Comment définir les conditions d'acceptation $\mathcal{F}$?

Pour chaque sous-formule $\psi_1 \mathbf{U} \psi_2$, on a un ensemble $\mathcal{F}_{\psi_1 \mathbf{U} \psi_2}$ défini par:

$$\mathcal{F}_{\psi_1 \mathbf{U} \psi_2} = \{ q \in Q \mid \psi_1 \mathbf{U} \psi_2 \notin q \vee \psi_2 \in q \}$$

idée: un état contenant $\psi_1 \mathbf{U} \psi_2$ doit reconnaître les modèles de $\psi_1 \mathbf{U} \psi_2$ et donc visiter un jour un état contenant ψ_2.

Pour en être sûr, on impose de visiter infiniment souvent des états contenant ψ_2… ou infiniment souvent des états contenant $\neg\, \psi_1 \mathbf{U} \psi_2$…

Dans les deux cas, on est sûr de ne pas attendre indéfiniment la satisfaction de ψ_2.

$\mathcal{A}_\phi$ et les modèles de ϕ

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$

Prenons un chemin dans $\mathcal{A}_\phi$ $q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4 \rightarrow \ldots$
étiqueté par le mot $\sigma_1 \sigma_2 \sigma_3 \sigma_4 \ldots$ de $(2^{AP})^\omega$

Alors on a:

$$\forall\, \psi \in q_i, \quad \sigma_i \sigma_{i+1} \sigma_{i+2} \ldots \models \psi$$

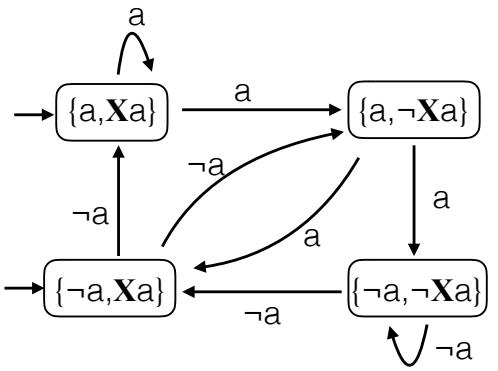
Construction de $\mathcal{A}_\phi$ - exemple

$\phi = \mathbf{X}\, a$

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$

$S_\phi = \{a, \neg a, \mathbf{X}a, \neg \mathbf{X}a\}$

$q_1 = \{a, \mathbf{X}a\}$,
$q_2 = \{a, \neg \mathbf{X}a\}$
$q_3 = \{\neg a, \mathbf{X}a\}$
$q_4 = \{\neg a, \neg \mathbf{X}a\}$



{a}{a}{¬a}… {¬a}{a}{a}… {a}{a}{a}…

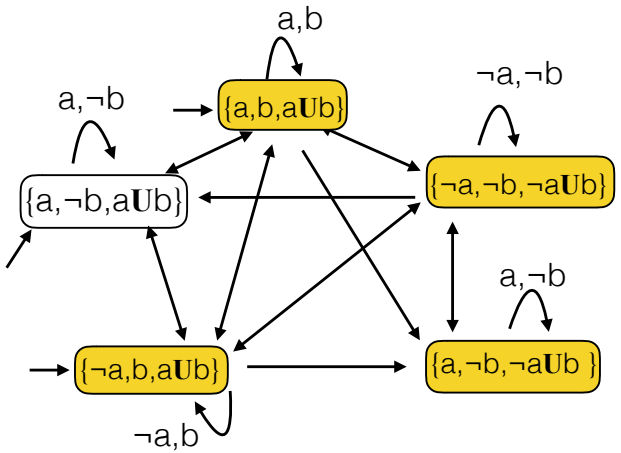
Construction de $\mathcal{A}_\phi$ - exemple

$\phi = a\ \mathbf{U}\ b$

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$

$S_\phi = \{a, \neg a, b, \neg b, a\mathbf{U}b, \neg a\mathbf{U}b\}$

$q_1 = \{a, b, a\mathbf{U}b\}$,
$q_2 = \{\neg a, b, a\mathbf{U}b\}$
$q_3 = \{a, \neg b, a\mathbf{U}b\}$
$q_4 = \{\neg a, \neg b, \neg a\mathbf{U}b\}$
$q_5 = \{a, \neg b, \neg a\mathbf{U}b\}$



état acceptant

{a,¬b}{a,¬b}{a,b}…

Construction de $\mathcal{A}_\phi$ - exemple

$\phi = \mathbf{G}\,(a \Rightarrow \mathbf{F}\ b)$

$\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$

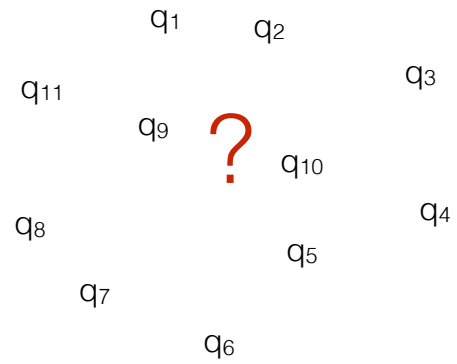
$S_\phi = \{a, \neg a, b, \neg b, \mathbf{F}b, \neg \mathbf{F}b, \neg a \vee \mathbf{F}b, a \wedge \neg \mathbf{F}b, \phi, \neg\phi\}$

if $q_1 = \{a, b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,
f $q_2 = \{a, b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \neg\phi\}$,
i $q_3 = \{a, \neg b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,
$q_4 = \{a, \neg b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \neg\phi\}$,
f $q_5 = \{a, \neg b, \neg \mathbf{F}b, \neg(a \Rightarrow \mathbf{F}b), \neg\phi\}$,
if $q_6 = \{\neg a, b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,
f $q_7 = \{\neg a, b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \neg\phi\}$,
i $q_8 = \{\neg a, \neg b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,
$q_9 = \{\neg a, \neg b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \neg\phi\}$,
if $q_{10} = \{\neg a, \neg b, \neg \mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,
f $q_{11} = \{\neg a, \neg b, \neg \mathbf{F}b, a \Rightarrow \mathbf{F}b, \neg\phi\}$.

i : états initiaux
f : états acceptant

Construction de $\mathcal{A}_\phi$ - exemple

$\phi = \mathbf{G}\,(a \Rightarrow \mathbf{F}\,b)$ $\quad\quad$ $\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$



On remarque ici que on ne peut jamais aller d'un état contenant ϕ à un état contenant ¬ϕ…

Construction de $\mathcal{A}_\phi$ - exemple

{a,b}{a,b}{a,¬b}{a,¬b}{a,¬b}{¬a,¬b}{¬a,b}…

$\phi = \mathbf{G}\,(a \Rightarrow \mathbf{F}\,b)$ $\quad\quad$ $\mathcal{A}_\phi = (Q, Q_0, \rightarrow, \mathcal{F})$

$q_1 = \{a, b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,

$q_3 = \{a, \neg b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,

$q_6 = \{\neg a, b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,

$q_8 = \{\neg a, \neg b, \mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,

$q_{10} = \{\neg a, \neg b, \neg\mathbf{F}b, a \Rightarrow \mathbf{F}b, \phi\}$,