M1 MPRI : Automates et Applications

Cours 4
Automates de mots infinis et logiques temporelles

kn@lri.fr

31 mars 2021

Plan

université PARIS-SACLAY

Revenons aux mots

université PARIS-SACLAY

Nous avons considérés les mots finis et les arbres, permettant de leur rajouter de la structure.

Certains systèmes sont cependant « infinis » par nature. Par exemple un protocole d'échange de messages, un programme de type « serveur » qui boucle à l'infini, . . . Pour de tels systèmes, on veut pouvoir s'assurer qu'une propriété est vraie « infiniement souvent », ou qu'une autre propriété devient vrai dès qu'une autre cesse de l'être.

Mots infinis

université PARIS-SACLAY

Définition

Étant donné un alphabet Σ, un mot infini sur Σ est une fonction totale de $\mathbb{N}_1 \to \Sigma$.

Rappel : $\mathbb{N}_1$ est l'ensemble des entiers naturels non nuls. Dans notre définition précédente, les mots finis étaient des suites de lettre donc des fonctions de $\{1, \ldots, n\} \to \Sigma$, pour un n fixé.

Ensemble de mots infinis

université PARIS-SACLAY

Définition

On note Σ^ω l'ensemble des mots infinis sur Σ. Une notation pour parler des mots finis ou infinis est Σ^∞ (on a donc $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$).

Attention ne pas confondre langage fini de mot infini et langage infini de mots finis !

Exemples

Si on considère $\Sigma = \{a, b\}$:

- $\{a, b, ab\}$ est un ensemble fini de mots finis
- $a^*b = \{b, ab, aab, aaab, \ldots\}$ est un ensemble infini de mots finis
- $\left\{ \begin{array}{l} \mathbb{N}_1 \to \Sigma \\ n \mapsto a \end{array} , \begin{array}{l} \mathbb{N}_1 \to \Sigma \\ n \mapsto b \end{array} \right\}$ est un ensemble fini de mots infinis
- $\left\{ \begin{array}{l} \mathbb{N}_1 \to \Sigma \\ n \mapsto a, n < k \\ n \mapsto b, n \geq k \end{array} \middle| \, k \in \mathbb{N}_1 \right\}$ est un ensemble infini de mots infinis

Aparté

université PARIS-SACLAY

Le symbole ω représente le plus petit ordinal transfini. Les ordinaux sont une généralisation des entiers naturels. Si on prend une définition ensembliste des entiers :

- $0 \equiv \varnothing$
- $1 \equiv \{0\} \equiv \{\varnothing\}$
- $2 \equiv \{0, 1\} \equiv \{\varnothing, \{\varnothing\}\}$
- $3 \equiv \{0, 1, 2\} \equiv \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}$
- $4 \equiv \{0, 1, 2, 3\} \equiv \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}, \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}\}$
- $\ldots$
- $\omega \equiv \mathbb{N} \equiv \{\varnothing, \ldots\}$
- $\omega + 1 \equiv \{0, 1, \ldots, \omega\}$

Aparté (2)

université PARIS-SACLAY

- $\omega + 2 \equiv \{0, 1, \ldots, \omega, \omega + 1\}$
- $\ldots$
- $\omega + \omega \equiv \omega 2$
- $\ldots$
- ω^2
- $\ldots$
- ω^ω
- $\ldots$
- $\left.\omega^{\omega^{\omega^{\cdots^{\omega}}}}\right\}\omega$ fois $\equiv \epsilon_0$

Aparté (3)

université PARIS-SACLAY

- on peut définir de l'arithmétique sur ces objets là
- on peut dépasser ϵ_0 et arriver aux ordinaux indénombrables
- il y a sur les ordinaux un principe d'induction transfinie, qui permet de faire des récurrences même sur les ordinaux (dénombrables)
- si besoin, on utilisera ω, mais pas plus, et le plus souvent comme une notation
- proche (mais distinct) de la notion de cardinal (le fait que « la taille de $\mathbb{R}$ est un infini plus grand que la taille de $\mathbb{N}$»)
- Georg Cantor (1845-1918) est devenu fou en découvrant ça

Plan

université PARIS-SACLAY

Puissance ω

université PARIS-SACLAY

Définition (Puissance ω d'un langage)

Soit $L \subseteq \Sigma^*$ un langage de mots finis. On définit le langage L^ω par :

$$L^\omega = \{\sigma = u_1 u_2 \dots u_n \dots \mid \forall i \geq 1, u_i \in L \setminus \{\epsilon\}\}$$

Autrement dit, c'est l'ensemble de tous les mots infinis que l'on peut construire en concaténant une infinité de mots *non-vides* choisis dans L.

Langages ω-réguliers

université PARIS-SACLAY

On peut généraliser la notion de langage régulier aux langages de mots infinis :

Définition (Langage ω-régulier)

Un langage $L \subseteq \Sigma^\omega$ est ω-régulier si et seulement si :

- *il est de la forme A^ω avec A un langage réguliers (de mots finis)*
- *ou il est de la forme $A \cdot B$ avec A un langage réguliers et B un langage ω-régulier*
- *ou il est de la forme $A \cup B$ où A et B sont deux langages ω-réguliers*

Langages ω-réguliers (2)

université PARIS-SACLAY

La seule façon de construire un langage ω-régulier est de prendre un langage régulier et de prendre sa puissance ω. Une fois qu'on a fait ça, on peut lui rajouter des préfixes finis (décrits par un langage régulier) ou prendre la réunion de deux langages ω-réguliers.

Langages reconnaissables de mots infinis

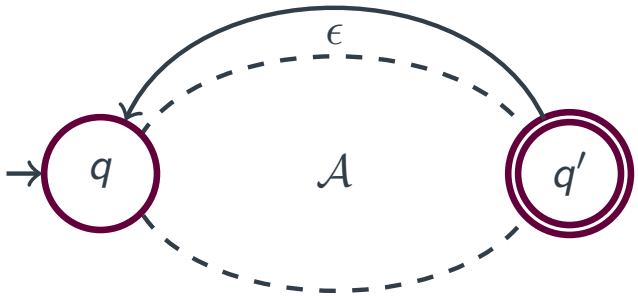
université PARIS-SACLAY

Puisque les langages ω-réguliers sont formés à partir de langages réguliers (donc reconnaissables) en les concaténant infiniment, peut on utiliser des automates ?

Langages reconnaissables de mots infinis

université PARIS-SACLAY

Puisque les langages ω-réguliers sont formés à partir de langages réguliers (donc reconnaissables) en les concaténant infiniment, peut on utiliser des automates ?



Si l'automate « boucle infiniment » en passant par q' il va reconnaître un mot de $L^{\omega}_{\mathcal{A}}$.

Plan

Automates (rappel)

université PARIS-SACLAY

Définition (Automate non-déterministe)

Un automate non-déterministe (Non-deterministic Finite Automaton, NFA) est un 5-uplet $(Q, \Sigma, \delta, I, F)$ où :

- *Q est un ensemble fini d'éléments appelés états*
- *Σ est un alphabet*
- *$\delta : Q \times \Sigma \to \mathcal{P}(Q)$ est une fonction de transition*
- *$I \subseteq Q$ est un ensemble d'états initiaux*
- *$F \subseteq Q$ est un ensemble d'états acceptant (ou finaux)*

(c'est la bonne vieilles définition des automates vue au premier cours)

Définition d'un run

On étend la définition de *run* aux mots infinis :

Définition

Soit $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ un automate et u un mot de longueur $\underline{n} \leq \omega$. On appelle run de $\mathcal{A}$ pour u une fonction $r : \underline{n} \to Q$ telle que :

- $r(0) \in I$
- $\forall i \in \underline{n}, r(i+1) \in \delta(r(i), u(i+1))$

Rappel : les ordinaux (et entre autre les entiers) sont à la fois des ensembles et des « nombres ». On note $\underline{n}$ pour dire qu'on considère n (l'entier ou ω) comme l'ensemble qui contient tous ses prédécesseurs.

États atteints par un run

université PARIS-SACLAY

Définition (États atteints par un run)

On considère un run r de l'automate $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ pour le mot u :

si $|u| = \underline{n} < \omega$: l'ensemble des états atteints par r est $\{r(n)\}$

si $|u| = \omega$: l'ensemble des états atteints par r est

$$\{q \mid \#\{i \mid r(i) = q\} \notin \mathbb{N}\}$$

On note $\lim(r)$ l'ensemble des états atteints.

Ci-dessus, $\#S$ dénote le cardinal de l'ensemble S.

Dans le cas d'un mot infini, l'ensemble des états atteint est l'ensemble des états qui apparaissent infiniment souvent dans le run.

Condition de Büchi

université PARIS-SACLAY

Définition (Run acceptant)

Étant donné un automate $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, un mot $u \in \Sigma^\infty$ et un run r de $\mathcal{A}$ pour u, on dit que r est acceptant si et seulement si $\lim(r) \cap F \neq \varnothing$.

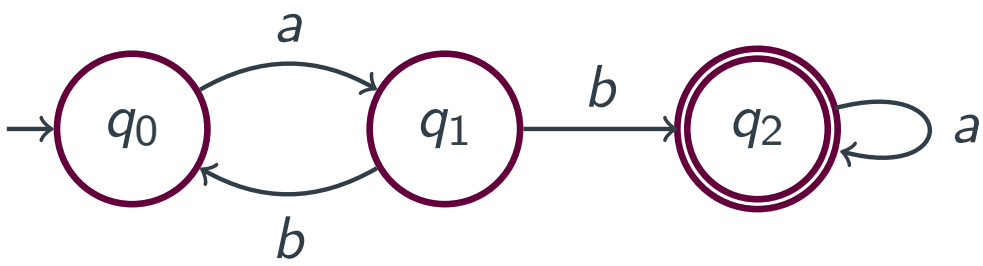
On note $L_\mathcal{A}$ l'ensemble des mots finis reconnus par $\mathcal{A}$ et $L^\omega_\mathcal{A}$ l'ensemble des mots infinis reconnus par $\mathcal{A}$.

La condition qu'un run « passe une infinité de fois par un état acceptant » est appelée condition de Büchi. Un langage de mots infinis reconnu ainsi est dit Büchi-reconnaissable.

Julius Richard Büchi, 1924-1984, mathématicien Suisse.

Exemple

Le langage L des mots infinis commençant par une suite finie de ab et finissant par une suite infinie de a est Büchi-reconnaissable :



Remarque : l'automate n'a rien de spécial. C'est celui qui reconnaît $(ab)^+a^*$ si on ne considère que les mots finis.

Déterminisation ?

université PARIS-SACLAY

Comme on a pas changé nos automates mais la condition d'acceptation, la notion de déterministe reste la même :

Un automate est déterministe si :

- I est un singleton
- $\forall q \in Q, x \in \Sigma, \quad \delta(q, x)$ est un singleton.

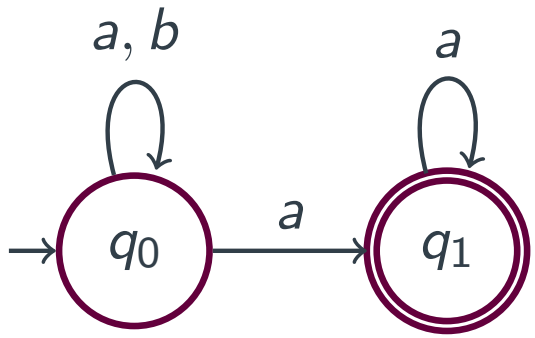
Déterminisation ? (2)

université PARIS-SACLAY

Théorème

Les automates de Büchi non-deterministes reconnaissent strictement plus de langages que les automates de Büchi déterministes.

Preuve : on considère le langage $(a|b)^* a^\omega$. Ce langage est reconnu par l'automate :



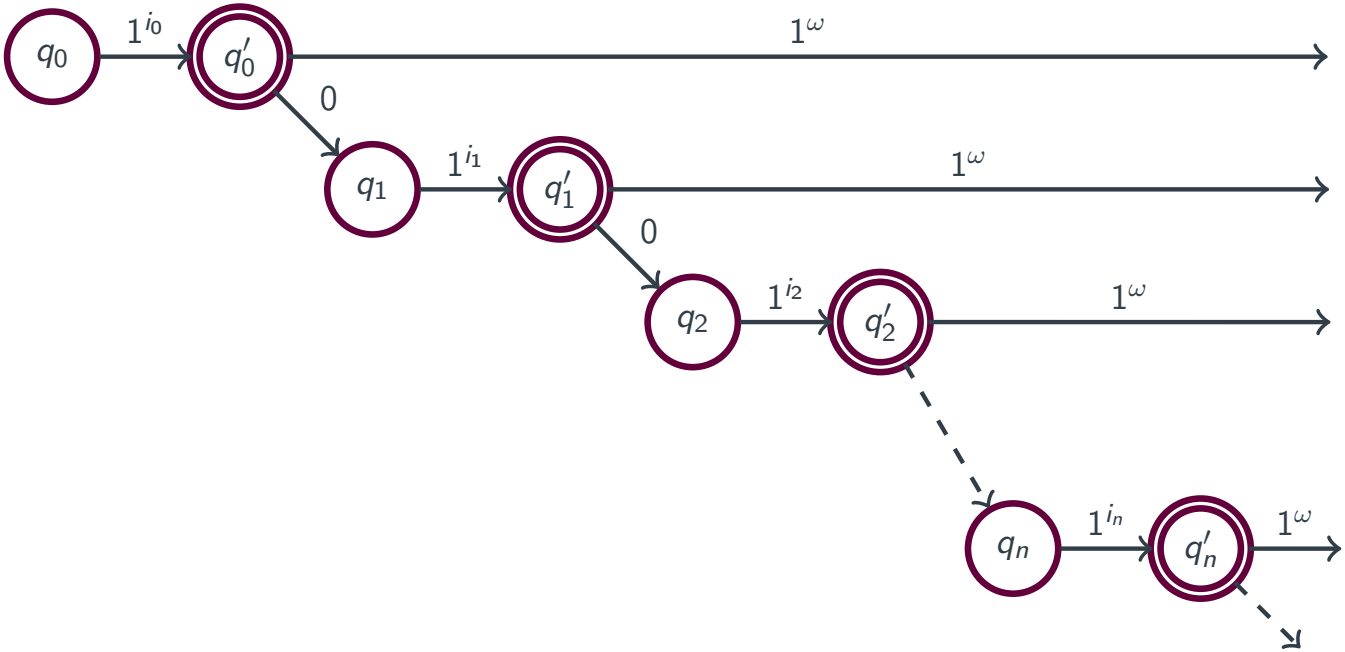
Déterminisation ? (3)

université PARIS-SACLAY

Supposons que ce langage est reconnu par un automate déterministe $\mathcal{A} = (Q, \{a, b\}, \delta, q_0, F)$. Appelons $n = |Q|$. Considérons le mot $m_0 = 1^\omega$. Ce mot fait partie du langage. Il y a donc un run qui passe une infinité de fois par un état de F. Appelons q'_0 le premier état de F atteint dans ce run, après avoir lu i_0 lettres.

Considérons maintenant le mot $m_1 = 1^{i_0}01^\omega$. Comme l'automate est déterministe, Le run de m_1 et celui de m_0 commencent de la même façon jusqu'à arriver en q'_0. Comme m_1 est un mot du langage, q'_0 possède une transition sortante vers un état q_1 puis de ce dernier on passe infiniment souvent dans un état $q'_1 \in F$.

Déterminisation ? (4)



Le mot $1^{i_0}01^{i_1}01^{i_2}0\ldots1^{i_{n+1}}01^\omega$, passe par $n+1$ états acceptants (les q'_i), hors il n'y a que n états dans l'automate. C'est donc qu'il y a une boucle impliquant l'un de ces états acceptants. Cet état à (par construction) une transition sortante pour 0. L'automate peut donc reconnaître des mots qui continennent une infinité de 0, ce qui n'est pas le cas de notre langage de départ.

Problèmes de décision

université PARIS-SACLAY

Théorème

Le vide du langage d'un automate de Büchi est décidable en temps linéaire.

Preuve : il suffit de trouver un état acceptant q_f :

- accessible depuis un état initial q_0
- accessible depuis lui même

Si on trouve un tel état q_f le langage n'est pas vide. On peut se ramener au problème d'accessibilité dans un graphe.

Problèmes de décision (2)

université PARIS-SACLAY

Théorème

Le plein du langage d'un automate de Büchi est décidable en temps exponentiel.

Note : on peut prendre le complémentaire et décider le vide. La construction du complémentaire d'un automate de Büchi non déterministe est exponentiel en taille.

Propriétés de clôtures

université PARIS-SACLAY

- Les automates de Büchi sont clos par union (comme pour les mots)
- Les automates de Büchi sont clos par complémentaire (preuve et construction compliquée)
- Les automates de Büchi sont clos par intersection

Produit d'automates naïf

On doit adapter la construction produit car celle utilisée dans le cas des mots ne fonctionne pas. En effet :

- $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, I_1, F_1)$.
- $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, I_2, F_2)$.

Supposons que $\mathcal{A}_1$ reconnaisse les mots $0(10)^\omega$ et que $\mathcal{A}_2$ reconnaisse les mots $(01)^\omega$. Un run pour $\mathcal{A}_1$ est de la forme $q_0 q_1 q_2 q_1 q_2 \ldots$ avec q_2 acceptant. Un run pour $\mathcal{A}_2$ est de la forme $p_0 p_1 p_0 p_1 \ldots$, avec p_1 acceptant. Avec le produit « naïf » on aurait donc un run : $(q_0, p_0)(q_1, p_1)(q_2, p_0)(q_1, p_1), \ldots$ avec *aucun état acceptant*. Hors, les deux langages sont identiques (suites infinies de 01 commençant par 0).

Produit d'automates amélioré

université PARIS-SACLAY

$\mathcal{A}_{1\wedge 2} = (Q_1 \times Q_2 \times \{1,2\}, \Sigma, \delta, I_1 \times I_2 \times \{1\}, F)$ avec :

- $\delta((q_1, q_2, i), a) = (q_1', q_2', f(q_1, q_2, i)),\ \ q_1' \in \delta_1(q_1, a), q_2' \in \delta_2(q_2, a)$
- $f(q_1, q_2, 1) = 2$ si $q_1 \in F_1$
- $f(q_1, q_2, 2) = 1$ si $q_2 \in F_2$
- $f(q_1, q_2, i) = i$ sinon
- $F = Q_1 \times F_2 \times \{2\}$

L'entier 2 signifie que le dernier état acceptant vu venait de F_1 (et donc, qu'on cherche un état acceptant de F_2). De même l'entier 1 signifie que le dernier état acceptant vu vient de F_2 (et donc qu'on cherche un de F_1). Un état final $(q, q_f, 2)$ signifie donc que l'on est dans un état final pour $\mathcal{A}_2$, mais qu'on est passé avant par un état final de $\mathcal{A}_1$.

Plan

Système de transitions

université PARIS-SACLAY

Si on considère un système de transitions (un programme, un protocole, un ensemble de processus, . . .) modélisable par un automate, les mots finis permettent d'exprimer des propriété de sûreté. On peut par exemple faire en sorte que les états acceptants soient les états « interdits » du système et demander si l'automate reconnait le vide. Si ce n'est pas le cas, on peut alors exhiber un mot, c'est à dire une suite de transitions qui fait aller d'un état initial à un état interdit (et ainsi détecter un bug).

Système de transitions (2)

université PARIS-SACLAY

D'autre propriétés sont intéressantes. Par exemple les propriétés de vivacité :

- Est-ce que tous les processus du système ont la main a un moment donné
- Est-ce que mon server se remet toujours en attente de connexion
- ...

Les mots infinis permettent de caractériser ces propriétés.

Définition (LTL)

Soit AP un ensemble de symboles appelés propositions atomiques, une formule de la logique temporelle linéaire (LTL) est une production finie de la grammaire :

$$
\begin{array}{rcll}
\phi & ::= & p & p \in AP \quad \textit{(proposition)} \\
 & | & \phi_1 \vee \phi_2 & \textit{(disjonction)} \\
 & | & \neg\phi & \textit{(negation)} \\
 & | & \mathcal{X}\phi & \textit{(next)} \\
 & | & \phi_1 \mathcal{U} \phi_2 & \textit{(until)}
\end{array}
$$

LTL (2)

université PARIS-SACLAY

Les formules de LTL sont interprétée sur des séquences infinies d'ensembles de propositions. Par exemple si on considère $AP = \{\text{wait}, \text{log}, \text{recv}, \text{send}\}$. On peut considérer la trace infinie :

$$\begin{matrix}\text{wait}\\ \text{log}\end{matrix} \quad \begin{matrix}\text{wait}\\ \text{log}\end{matrix} \quad \text{recv} \quad \text{send} \quad \begin{matrix}\text{wait}\\ \text{log}\end{matrix} \quad \text{recv} \quad \text{send} \quad \ldots$$

On veut se poser des questions comme :

- est-ce que je me remets toujours à logger ?
- est-ce que j'envoie toujours quelque chose après avoir reçu ?
- est-ce que si je suis en attente, je reçois toujours quelque chose après ?

Semantique de LTL

université PARIS-SACLAY

Définition

Soit w une séquéquence infinie de $\mathcal{P}(AP)^\omega$, la valeur de vérité d'une formule LTL est donnée par :

$$
\begin{array}{lcl}
w \vdash p & \Leftrightarrow & p \in w(0) \\
w \vdash \phi_1 \vee \phi_2 & \Leftrightarrow & w \vdash \phi_1 \text{ ou } w \vdash \phi_2 \\
w \vdash \neg\phi & \Leftrightarrow & w \nvdash \phi \\
w \vdash \mathcal{X}\phi & \Leftrightarrow & w^1 \vdash \phi \\
w \vdash \phi_1 \mathcal{U} \phi_2 & \Leftrightarrow & \exists i \geq 0 \text{ tel que } w^i \vdash \phi_2 \text{ et } \forall k < i, \quad w^k \vdash \phi_1
\end{array}
$$

Ici $w^i = w(i)w(i+1)\ldots$ est le suffixe infini de w commençant en position i.

Notations

université PARIS-SACLAY

Pour réduire la taille des formules, on pourra utiliser les notations suivante :

$\phi_1 \wedge \phi_2$	$\equiv$	$\neg(\neg\phi_1 \vee \neg\phi_2)$	conjonction
$\phi_1 \Rightarrow \phi_2$	$\equiv$	$\phi_1 \vee \neg\phi_2$	implication
$\phi_1 \Leftrightarrow \phi_2$	$\equiv$	$(\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1)$	équivalence
$\top$	$\equiv$	$p \vee \neg p$	vrai, pour $p \in AP$
$\bot$	$\equiv$	$\neg\top$	faux
$\mathcal{F}\phi$	$\equiv$	$\top\mathcal{U}\phi$	futur
$\mathcal{G}\phi$	$\equiv$	$\neg\mathcal{F}\neg\phi$	global

Exemples

université PARIS-SACLAY

$AP = \{\text{wait}, \text{log}, \text{recv}, \text{send}\}$.

- $\mathcal{G}(\text{recv} \Rightarrow \mathcal{X}\text{send})$

Exemples

université PARIS-SACLAY

$AP = \{\text{wait}, \text{log}, \text{recv}, \text{send}\}$.

- $\mathcal{G}(\text{recv} \Rightarrow \mathcal{X}\text{send})$: « il est toujours vrai que j'envoie juste après avoir reçu »
- $\mathcal{F}\neg\text{log}$

Exemples

université PARIS-SACLAY

$AP = \{\text{wait}, \text{log}, \text{recv}, \text{send}\}$.

- $\mathcal{G}(\text{recv} \Rightarrow \mathcal{X}\text{send})$: « il est toujours vrai que j'envoie juste après avoir reçu »
- $\mathcal{F}\neg\text{log}$: « il y aura un moment où je m'arrête de logguer (pendant un instant)»
- $\mathcal{G}\neg(\text{send} \wedge \text{recv})$

Exemples

université PARIS-SACLAY

$AP = \{\text{wait}, \text{log}, \text{recv}, \text{send}\}$.

- $\mathcal{G}(\text{recv} \Rightarrow \mathcal{X}\text{send})$: « il est toujours vrai que j'envoie juste après avoir reçu »
- $\mathcal{F}\neg\text{log}$: « il y aura un moment où je m'arrête de logguer (pendant un instant)»
- $\mathcal{G}\neg(\text{send} \wedge \text{recv})$: « je n'envoie jamais au même moment où je reçois »

Revisitons nos propriétés

université PARIS-SACLAY

Sûreté : quelque chose de mauvais n'arrive jamais. $\mathcal{G}\neg\phi$

Vivacité : quelque chose de bon se reproduit sans cesse : $\mathcal{G}(\phi_1 \Rightarrow \mathcal{X}\phi_2)$.

Équité forte : Un évènement *a* se produite infiniment souvent, et quand il se produit, un évènement *b* se produit

Revisitons nos propriétés

université PARIS-SACLAY

Sûreté : quelque chose de mauvais n'arrive jamais. $\mathcal{G}\neg\phi$

Vivacité : quelque chose de bon se reproduit sans cesse : $\mathcal{G}(\phi_1 \Rightarrow \mathcal{X}\phi_2)$.

Équité forte : Un évènement a se produite infiniment souvent, et quand il se produit, un évènement b se produit : $\mathcal{G}\mathcal{F}\phi_a \Rightarrow \mathcal{G}\mathcal{F}\phi_b$

Comment décider ces propriétés ?

LTL vers Automates de Büchi

université PARIS-SACLAY

Définition

Pour toute formule LTL, on peut construire un automate de Büchi qui reconnaît le même langage que la formule.

Remarque : le langage d'une formule ϕ est l'ensemble des mots de $w \in \mathcal{P}(AP)^\omega$ tels que $w \vdash \phi$. La complexité est EXPTIME, des algorithmes optimisés existent pour ne pas exploser dans tous les cas.