INF 302 : Langages & Automates

Introduction du cours - Informations générales, motivations

Yliès Falcone

ylies.falcone@univ-grenoble-alpes.fr — www.ylies.fr

Univ. Grenoble-Alpes, Inria
-
Laboratoire d'Informatique de Grenoble - www.liglab.fr
Équipe de recherche LIG-Inria, Corse - team.inria.fr/corse/

Année Académique 2020 - 2021

Chap. 0 - Introduction - Infos générales et motivations

Chap. 0 - Introduction - Infos générales et motivations

Présentation

Nom : Yliès Falcone

Affiliation : Univ. Grenoble Alpes, Inria
et LIG (Laboratoire d'Informatique de Grenoble)

Email : Ylies.Falcone@univ-grenoble-alpes.fr (sans accents)
Insérer svp. ***[INF 302]*** *en début de sujet de mail.*

Page Web : www.ylies.fr

Téléphone : +33 4 38 78 29 51 (fixe).

Bureaux :
- Bâtiment IMAG
  700 avenue Centrale
  Domaine Universitaire
- Antenne Inria Giant
  DRT/LETI/DACLE/ – Bâtiment 50C
  Minatec Campus, 17 rue des Martyrs, 38054 Grenoble.

Équipe pédagogique

Resp. UE : Yliès Falcone

Cours FR : Yliès Falcone

Cours EN : Saddek Bensalem (MIN_INT)

TD FR : Nicolas Basset, Cristian Ene, David Monniaux, Anne Rasse, David Rouquet, Alexandra Steinhilber, Vincent Tavernier, Paul Youssef

- MAT_S3_01 : ?
- MAT_S3_02 : ?
- MIN_S3_01 : ?
- MIN_S3_02 : ?
- MIN_S3_03 : ?
- INF_S3_01 : ?
- INF_S3_02 : ?
- MAT_S3_02 : ?
- INF_S3_03 : ?
- INF_S3_04 : ?

TD EN : MIN_INT : Akshay Mambakam

Contacts : nicolas.basset1@univ-grenoble-alpes.fr, david.rouquet@tetras-libre.fr, alexandra.steinhilber@grenoble-inp.org, Prenom.Nom@univ-grenoble-alpes.fr.

Sujet du cours et plan

Langages (réguliers) et Automates

Plan (approximatif) du cours

1. Introduction : informations générales, objectifs, motivations
2. Automates à États Finis Déterministes
   - Définition (syntaxe et sémantique)
   - Opérations de composition
   - Algorithmes et problèmes de décision
   - Minimisation et équivalence
3. Automates à États Finis Non-Déterministes
4. Automates à États Finis Non-Déterministes avec ϵ-transitions
5. Expressions régulières
6. Grammaires régulières
7. Langages non-réguliers et lemme de l'itération

Évaluation

Examen Final (EF)

- coefficient : 1,2
- durée : 2 heures
- dates : entre le 6 et le 10 janvier (après interruption pédagogique de Noël)

Examen à Mi-parcours (EM)

- coefficient : 0,4
- durée : 2 heures
- programme : Automates
- dates : avant ou après l'interruption pédagogique de la Toussaint

Évaluation Continue (EC)

- coefficient : 0,4
- dates : à n'importe quel TD
- peut prendre n'importe quelle forme

$$\text{Note Finale} = \frac{1.2 * \text{EF} + 0.4 * \text{EM} + 0.4 * \text{EC}}{2}$$

Attention aux ABI !

Ressources pédagogiques et bibliographie

Ressources pédagogiques

- Toutes les ressources pédagogiques se trouveront sur le **Moodle** du cours :

  https://imag-moodle.e.ujf-grenoble.fr

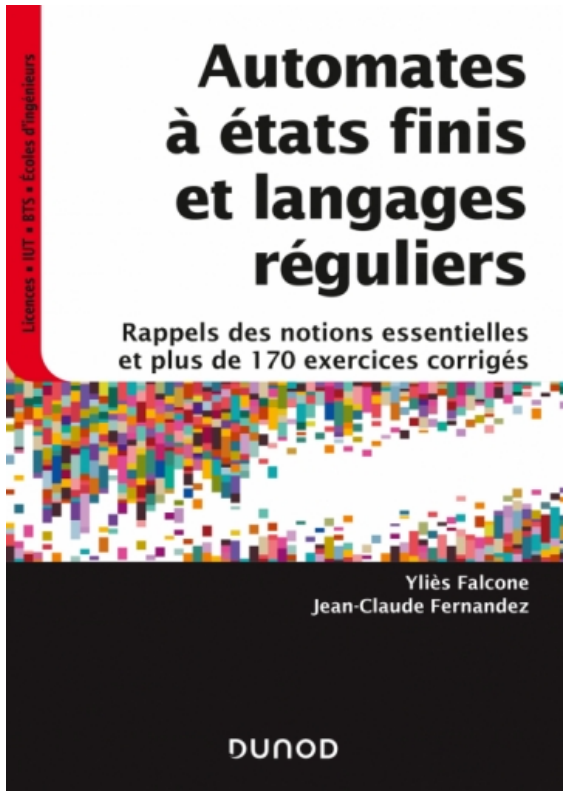
  - Transparents de cours.
  - Poly avec les sujets de TD.
- **Aude**, un outil pour travailler seul avec les automates :

  http://aude.imag.fr

  (implémenté par des anciens étudiants d'INF 302 durant les stages d'excellence)
  (contact & rapport de bug : aude-contact@univ-grenoble-alpes.fr)

Communication via le Moodle. **Inscrivez vous !**

Ressources pédagogiques et bibliographie



Quelques références supplémentaires

- J. Hopcroft, R. Motwani, J. Ullman, *Introduction to Automata Theory, Languages and Computation*, 2nd edition, Addison-Wesley, 2001
- P. Wolper. *Introduction à la calculabilité* - Paris : InterEditions, 1991.
- Cl. Benzaken, *Systèmes Formels*, Masson, 1991.

Chap. 0 - Introduction - Infos générales et motivations

Les objectifs de ce cours

Méthode et approche scientifique

Objectif : **Résolution des problèmes (informatiques)**

- La modélisation mathématiques de problèmes informatiques.
- La recherche de solutions *efficaces et correctes*.
- L'analyse des solutions.
- La présentation des solutions.

Connaissances spécifiques

- Différents formalismes pour la définition des langages (formels) : automates, expressions régulières (et grammaires).
- Leur étude d'un point de vue algorithmique : problèmes de décision, composition etc.
- L'étude de leur pouvoir expressif.
- Définition d'algorithmes "classiques" en informatique.

Chap. 0 - Introduction - Infos générales et motivations

Chap. 0 - Introduction - Infos générales et motivations

Automates : leur histoire

D'hier

- Premiers automates conçus plusieurs siècles avant notre ère [a].
- Premiers automates pour le calcul définis dans les années 40 et 50.
- La motivation au départ était l'étude du cerveau humain.

a. fr.wikipedia.org/wiki/Automate_mécanique

À aujourd'hui

Utilisés dans les domaines suivants (entre autres) :

- Conception de matériels informatiques (ex : CPU, GPU, . . .).
- Reconnaissance de texte (ex : formulaire Web, moteur de recherche).
- Conception et vérification des protocoles (p.ex. TCP/IP).
- Programmation de robots, plan de production industriel.
- Vérification des programmes et de matériels informatiques.
- Synthèse de programmes.
- Compilation des langages.
- Biologie (génomique).
- Compression de données.

Chap. 0 - Introduction - Infos générales et motivations

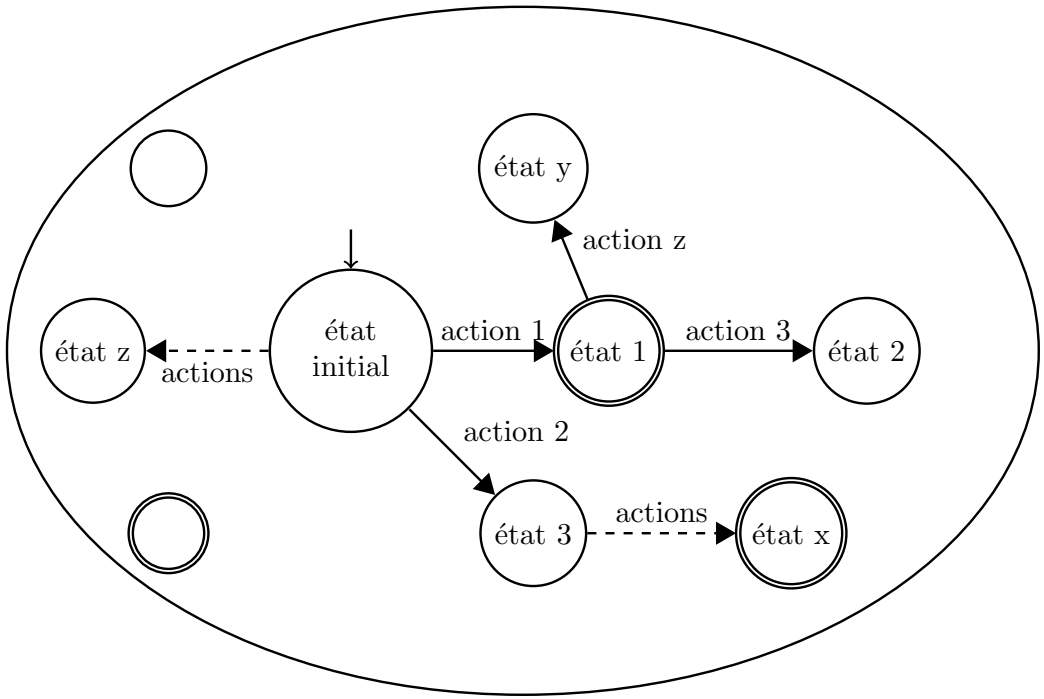
Automates : une première description informelle

Un automate est défini par :

- un ensemble d'**états**,
- des entrées (actions ou événements),
- des sorties (optionnelles), et
- des règles décrivant les **transitions** entre les états.

Représentation la plus simple sous forme de **graphe** :

- *noeuds* : états (certains sont spéciaux),
- *arcs* : transitions,
- *étiquettes* des arcs : événements ou actions déclenchant les transitions.

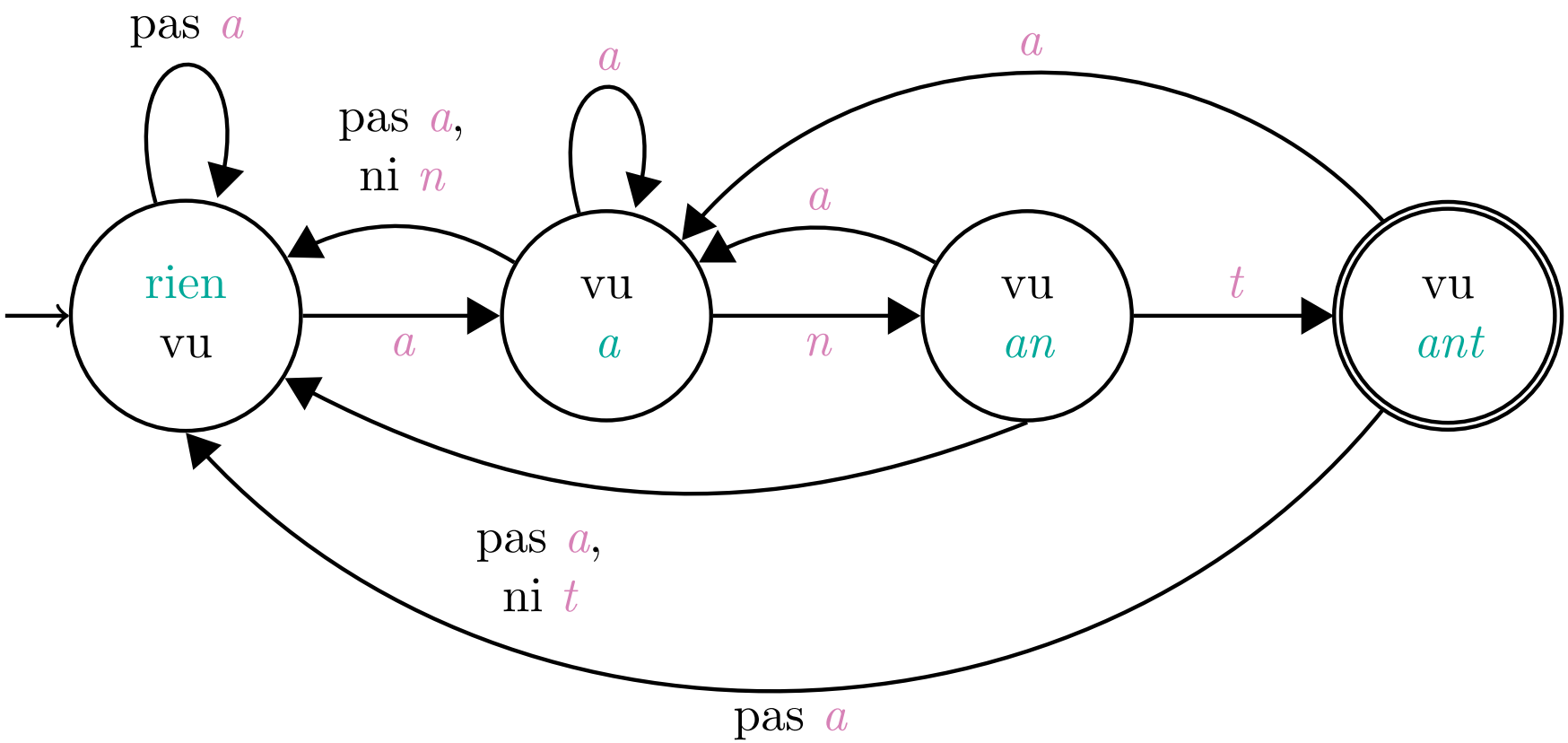


Nous allons voir des exemples simples d'automates utilisés en modélisation et vérification.

Exemple : reconnaissance de (motifs de) texte

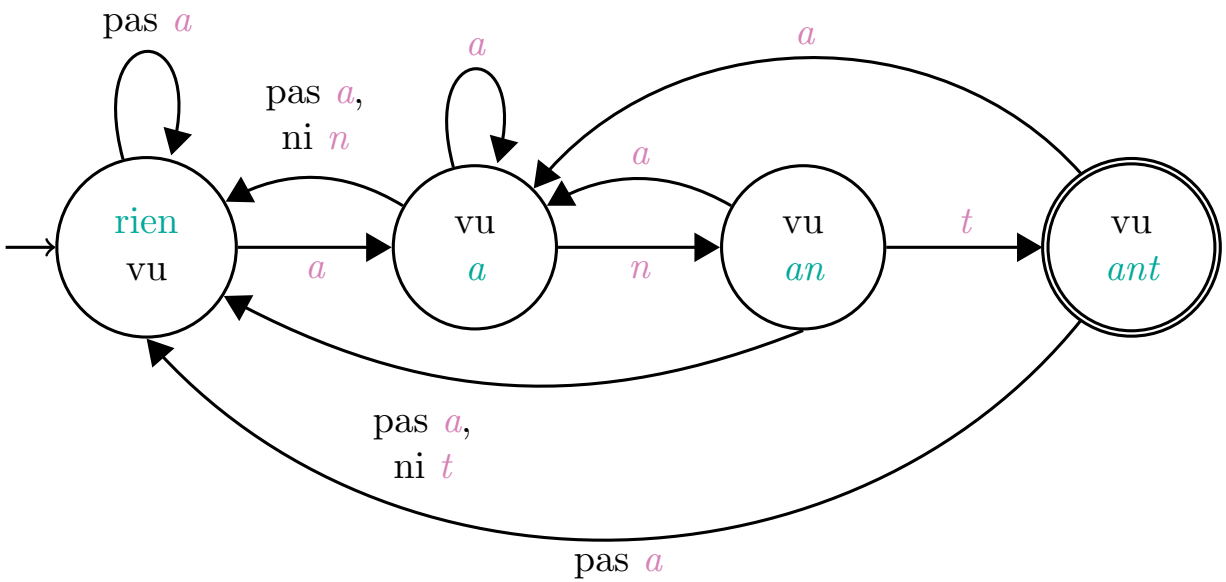
L'automate modélise un programme lisant caractère par caractère une chaine de caractères, par exemple depuis une mémoire tampon (« buffer »).

Reconnaissance des chaines de caractères qui *terminent par* ***ant***.



Exemple : reconnaissance de (motifs de) texte

Implémentation



Un algorithme qui réalise l'automate :

1. Lire l'entrée suivante.
2. Décider du prochain état en fonction de l'état courant et de l'entrée.
3. Changer d'état.

```
1   c = getNextInput();
2   switch (etat_courant)
3   {
4   ...
5   case 2: /* etat "vu a" */
6           if (c=='n') etat_courant = 3
7           else if (c=='a') break;
8           else etat_courant = 1
9   break;
10  ...
```

Exemple : reconnaissance de (motifs de) texte

Il existe des générateurs de code qui à partir d'automates ou de descriptions plus concises (que nous verrons dans le cours) permettent de générer le code "reconnaisseur".

Exemple (Reconnaissance de motifs)

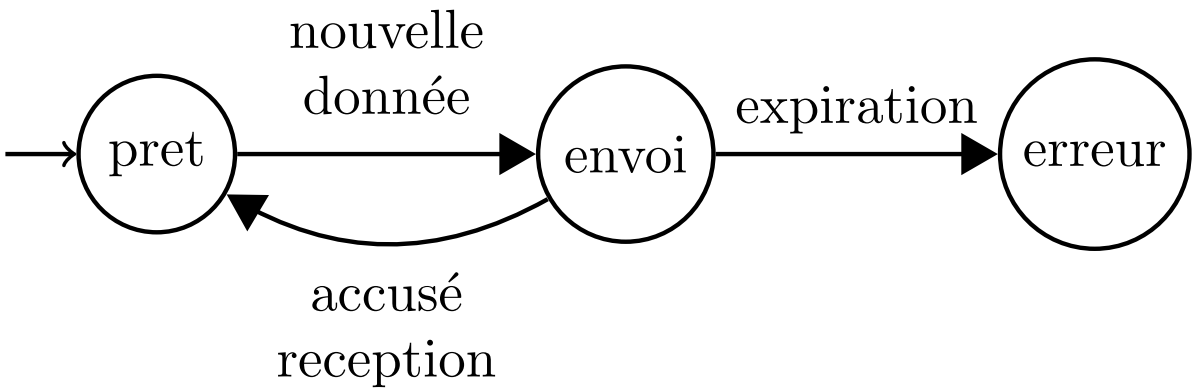
- En compilation de langages (outil LeX) pour reconnaitre les mots clés du langage.
- En UNIX :
  - Lister (`ls`) tous les fichiers (`*`)

```
1  ls *
```

  - Lister (`ls`) tous les fichiers terminant par .txt (`*.txt`)

```
1  ls *.txt
```

Exemple : protocole pour l'envoi de données



Exemple : modélisation d'une transaction électronique

Contexte

Nous voulons modéliser une *transaction électronique*.

Participants :

- un client,
- un marchand,
- une banque.

Le client veut acheter une marchandise chez le marchand et la paye de manière électronique (avec son numéro de CB)

Exemple : modélisation d'une transaction électronique

Les actions des participants

Actions des participants :

- Le client peut payer sa marchandise en envoyant au marchand l'argent sous la forme d'un message électronique (**paie**)
- Il peut aussi abandonner la transaction et récupérer son argent (**abd**)
- Le marchand peut envoyer la marchandise au client (**env**)
- Le marchand peut solder le paiement (**sol**)
- La banque peut transférer l'argent au marchand (**tra**)

Hypothèses sur le comportement des participants :

- La banque se comporte de manière honnête.
- Le marchand doit faire attention à ne pas livrer la marchandise sans être payé.
- Le client va essayer de recevoir la marchandise tout en récupérant son argent.

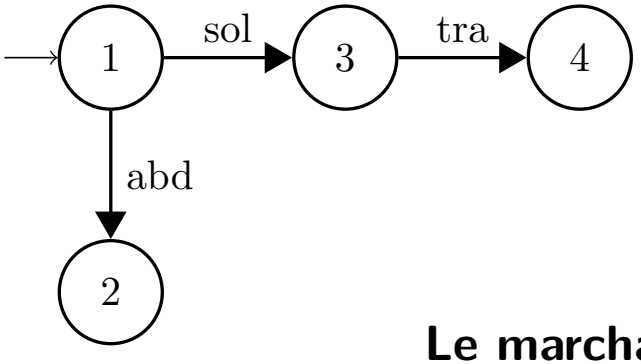
Objectifs

1. Modéliser le comportement des trois participants.
2. Voir s'il y a un moyen pour le client de recevoir la marchandise sans payer

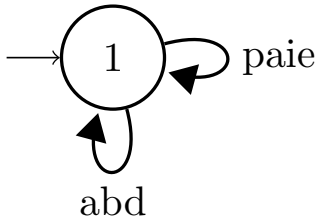
Exemple : modélisation d'une transaction électronique

Modélisation des participants et d'une transaction

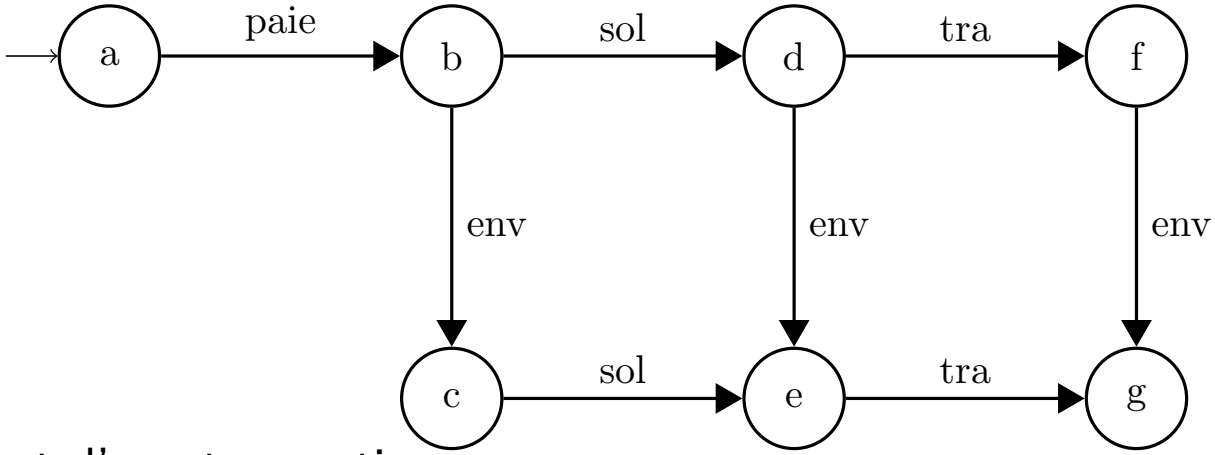
La banque



Le client



Le marchand



Pour le déroulement d'une transaction :

- chacun des participants peut agir librement ; mais
- "synchronisation" sur les actions communes.

Exemple : modélisation d'une transaction électronique

Vérification du modèle

Approche :

- On construit un *automate* qui modélise les comportements des trois participants en composant leurs comportements respectifs.
- On utilise un *algorithme* pour savoir si certains comportements indésirables apparaissent dans l'automate « composé ».
  Par exemple, si des états indésirables sont accessibles.

Exemple de problèmes (exercice de TD)

Donner un automate qui modélise la composition des trois participants... et étudier les propriétés attendues du protocole sur cet automate.

Étude des automates

Pour faire ceci nous avons besoin d'étudier les automates pour savoir :

- ce qu'on peut et ce qu'on ne peut pas décrire avec un automate.
- définir des opérations pour composer des comportements.
- développer des algorithmes sur les automates.

Exemple : évolution des espèces sur une étrange planète

Contexte

Utilisation des automates pour modéliser la dynamique d'un système.

Nous prenons l'exemple d'une planète lointaine où cohabitent trois espèces que nous appellerons les espèces rouge, bleue et orange [1].

Nous souhaitons modéliser l'évolution de la population de ces espèces selon leur mode de reproduction, qui suit les règles suivantes :

- 2 individus de deux *espèces différentes* peuvent s'unir ;
- la reproduction tue les deux individus ;
- la reproduction génère 2 individus de la troisième espèce.

Par exemple : 1 rouge et 1 bleu $\rightarrow$ 2 orange.

Nous supposons également que :

- des individus peuvent s'unir juste après avoir été générés (pas de distinction adulte/enfant) ;
- les individus ne peuvent mourir que lors de la reproduction.

1. Voir la notion d'automate cellulaire et le jeu de la vie.

Exemple : évolution des espèces sur une étrange planète

Observations et questions

Observons, avec ces règles, que :

- le nombre d'individus n'évolue pas,
- la planète n'évolue plus si tous les individus sont de la même espèce.

La configuration de la planète est déterminée par le nombre d'individus de chaque espèce.

Quelques (exemples de) questions :

- Est-ce qu'il y a plusieurs évolutions possibles ?
- Est-ce que la configuration initiale détermine complètement l'évolution de la population ?
- Est-ce certaines configurations mènent inévitablement à un blocage de l'évolution ?
- . . .

Exemple : évolution des espèces sur une étrange planète

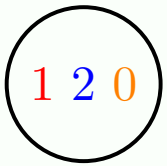
Modélisation

Modélisation de l'évolution de la population par un automate

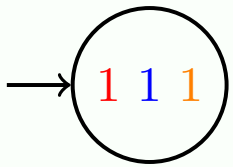
- État courant représente la configuration de la planète par 3 entiers (effectifs espèces)
- État initial représente les effectifs initiaux (équilibrés si possible)
- État terminal où espèces ne peuvent plus évoluer (une espèce a dominé les autres) avec deux espèces avec effectif 0
- Une transition indique une naissance d'individus suite à une reproduction
  - L'événement rouge (r) représente la naissance de deux individus rouge et la mort d'un individu bleu et d'un individu orange
  - Les événements bleu (b) et orange (o) se définissent de manière analogue

État 1 2 0

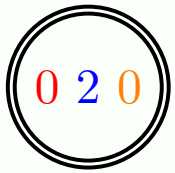
- Il y a 1 individu de l'espèce rouge
- Il y a 2 individus de l'espèce bleu
- Il y a 0 individu de l'espèce orange



État initial

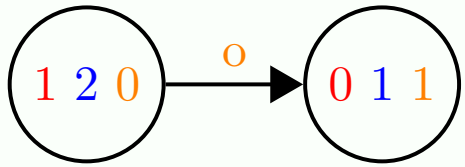


État terminal



Transition

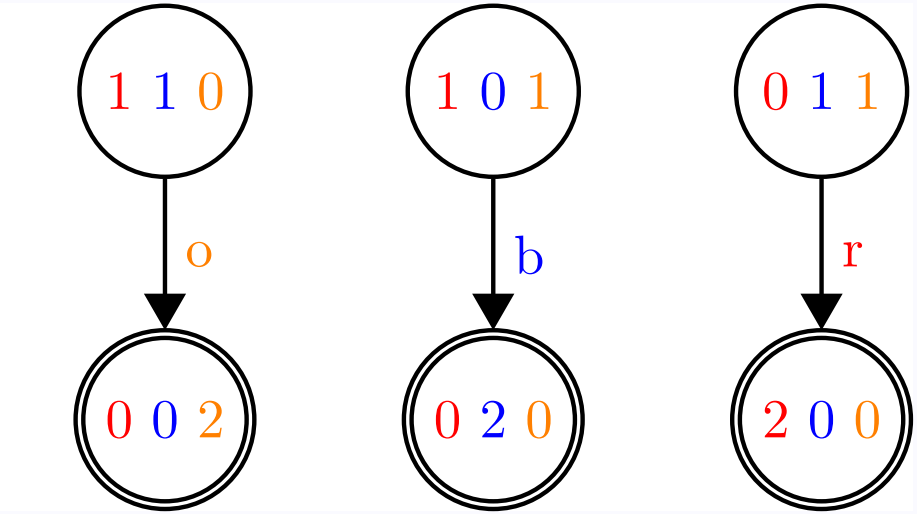
Naissance individu orange à partir de l'état 1 2 0 :



Exemple : évolution des espèces sur une étrange planète

Évolution à deux individus

Évolutions de la planète avec **deux** individus



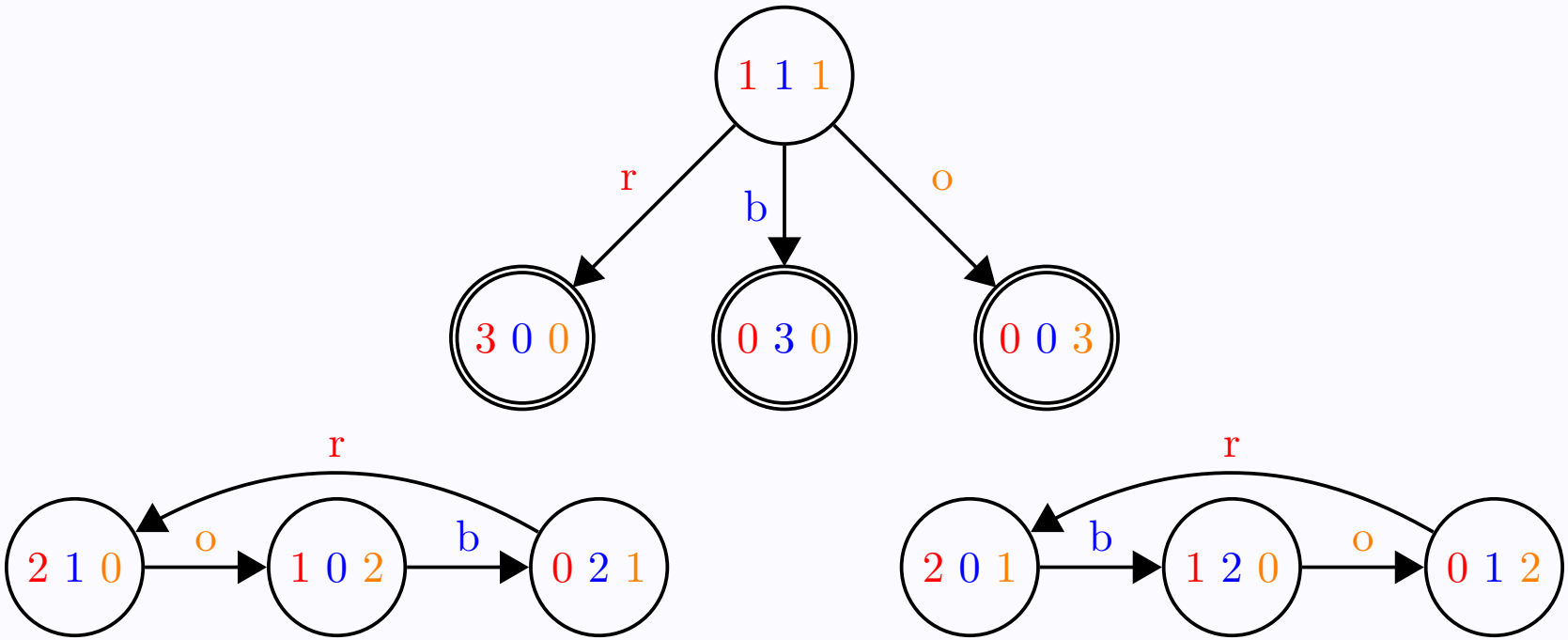
- Tous les états mènent inévitablement à l'arrêt de l'évolution.

(Nous avons fait le choix (arbitraire) d'avoir comme états initiaux possibles les états où les deux individus ne sont pas de la même espèce.)

Exemple : évolution des espèces sur une étrange planète

Évolution à trois individus

Évolutions de la planète avec **trois** individus

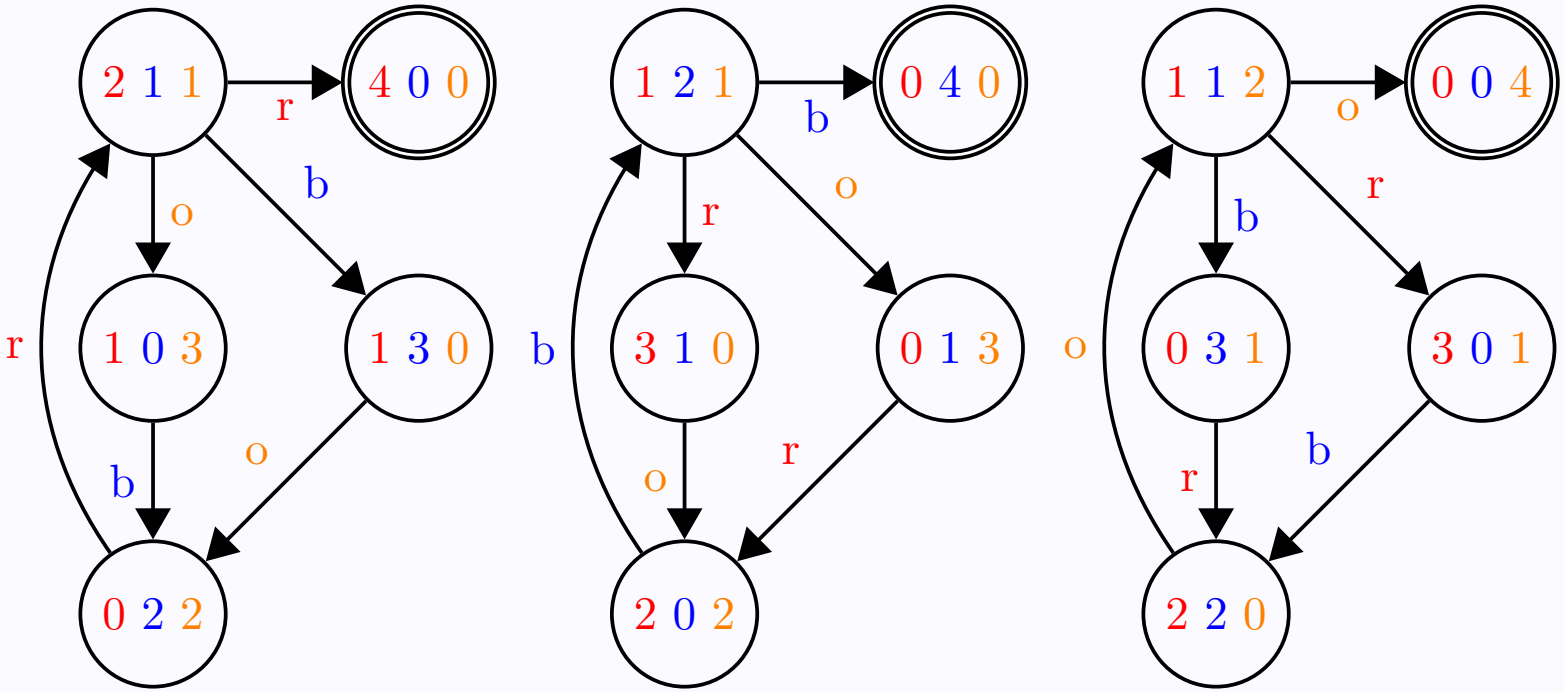


- Certains états mènent inévitablement à l'arrêt de l'évolution : 1 1 1, 3 0 0, 0 3 0, 0 0 3.
- Certains états ne peuvent pas mener à l'arrêt de l'évolution : 2 1 0, 1 0 2, 0 2 1, 2 0 1, 1 2 0, 0 1 2.

Exemple : évolution des espèces sur une étrange planète

Évolution à quatre individus

Évolutions de la planète avec **quatre** individus



- Certains états mènent inévitablement à l'arrêt de l'évolution : 4 0 0, 0 4 0, 0 0 4.
- Les autres états mènent possiblement à l'arrêt de l'évolution.

En TD, nous continuerons cette modélisation pour trouver une représentation qui passe mieux à l'échelle (càd où le nombre d'états n'explose pas avec le nombre de participants).

Chap. 0 - Introduction - Infos générales et motivations

À propos de l'informatique théorique

Une question fondamentale en informatique est la suivante :

Quels problèmes peut-on résoudre à l'aide d'une « machine à calculer » ?

Autrement dit :

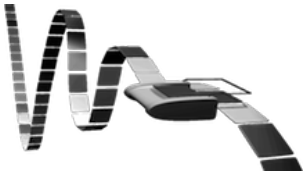
- Existe-t-il une limite à ce qu'on peut programmer ?
- Si oui, quelle est cette limite ?

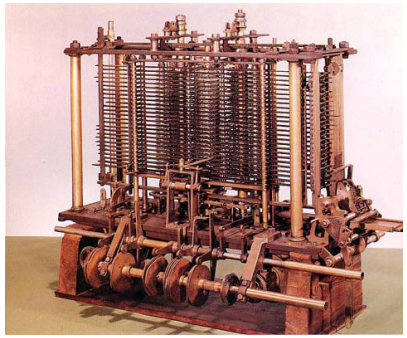
Quelques éléments de réponses

Alan Turing a étudié cette question en 1930 avant même que le premier ordinateur existe.

Turing a conçu et étudié une machine abstraite, un **automate** appelé

Machine de Turing.





La machine de Turing est un modèle d'ordinateur qui est « compatible » avec *tous* les modèles d'ordinateurs.
Les réponses à ces questions obtenues sur la machine de Turing s'appliquent donc à tous les ordinateurs.

Quelques éléments de réponses (suite)

Existe-t-il une limite à ce qu'on peut programmer ?

Oui, il y a des problèmes qu'on ne peut pas résoudre à l'aide d'un ordinateur, même si on suppose une mémoire non-bornée (« infinie ») et un processeur aussi rapide que l'on veut.

On parle de fonctions *non calculables* et de problèmes *non décidables*.

Exemple (Problèmes indécidables)

- Arrêt d'un programme.
- β-équivalence de deux termes en λ-calcul ($\sim$ équivalence de deux programmes fonctionnels).
- Toute question non triviale dont la réponse dépend uniquement du résultat d'un programme.
- ...

Quelques éléments de réponses (suite)

Quelle est la limite de ce qu'on peut programmer ?

Les réponses sont :

- On a plusieurs modèles pour représenter des problèmes ou leurs solutions, et différents modèles ont différentes limites. Certains modèles sont équivalents.
- Caractériser ces limites peut se faire de plusieurs manières équivalentes : automates, classes de fonctions mathématiques, formules de logique.

À propos des automates comme modèle de calcul

- Les automates à états finis (étudiés dans ce cours) sont un modèle restreint de machine de Turing (avec mémoire finie).
- La plupart des problèmes (intéressants) sur les automates sont décidables.

Complexité des problèmes et algorithmes

Stephen Cook a re-considéré en 1969 les mêmes questions que A. Turing mais en s'intéressant à ce qui peut être calculé de manière efficace. [2]

Les notions « d'efficacité » ainsi que les réponses à ces questions ont été développées à l'aide d'automates : on parle de problèmes Polynomiaux et de problèmes Non-déterministe Polynomiaux.

- Problèmes P : test de primalité, évaluation d'un circuit.
- Problèmes NP : SAT (trouver les valeurs de vérité qui rendent une proposition logique vraie), coloriage de graphe.

Une question qui reste aujourd'hui ouverte est :

$$\mathrm{P} \stackrel{?}{=} \mathrm{NP}$$

(une des questions à 1 M \$ – *Millenium Prize Problem* [a])

a. http://fr.wikipedia.org/wiki/Problèmes_du_prix_du_millénaire

Remarque En M1, vous aurez un cours dédié de calculabilité et complexité. □

2. Complexity Zoo : https://complexityzoo.uwaterloo.ca/Complexity_Zoo

Chap. 0 - Introduction - Infos générales et motivations

Résumé - Chap. 0 - Introduction - Infos générales et motivations

Résumé

- Connaissances :
  - formalismes pour définir et représenter les langages et leur pouvoir expressifs,
  - algorithmes (liés aux problèmes de décision).
- Automates = outils pour modéliser et vérifier (la correction).
- Automates = outils de l'informatique du quotidien.
- Automates = outils de l'informatique théorique.

Pour le prochain cours

- S'inscrire sur le Moodle.
- Donner la séquence d'actions permettant au client de récupérer la marchandise sans payer.
- Modifier les automates des participants de manière à ce qu'un client ne puisse plus récupérer la marchandise sans payer.
- Consulter les liens sur les applications des automates (cf. Moodle).