Audition pour un poste d'enseignant-chercheur à l'EPITA Paris

Gaëtan Staquet

Centre Inria de l'Université de Rennes

23 mai 2025

Pour consulter ces diapositives :
`https://www.gaetanstaquet.com/MCF/AuditionEPITA.pdf`

Qui suis-je ?

Parcours académique :

2024 – Présent Postdoctorat au Centre Inria de l'Université de Rennes.
Collaboration : Nathalie Bertrand et Patricia Bouyer (LMF, ENS Paris-Saclay).

2020 – 2024 Doctorat à l'Université de Mons et l'Université d'Anvers.
Supervision : Véronique Bruyère et Guillermo A. Pérez.

Août 2023 – Oct. 2023 Séjour de recherche à l'Université Radboud (Nijmegen, Pays-Bas) chez Frits W. Vaandrager.

2015 – 2020 Bachelier et master en sciences informatiques à l'Université de Mons.

Thématique de recherche : Apprentissage de modèles formels.

Participation aux activités collectives :

- Membre de plusieurs Conseils et Comités.
- Comité d'évaluation des artefacts pour QEST+FORMATS 2025.
- Revues pour 2 journaux et 5 conférences.
- Activités de vulgarisation.

Liste de mes publications

Journaux internationaux :

1. Swen Jacobs *et al. The Reactive Synthesis Competition (SYNTCOMP) : 2018 – 2021*, International Journal on Software Tools for Technology Transfer (STTT), 2024.

Conférences internationales :

1. Véronique Bruyère, Guillermo A. Pérez, Gaëtan Staquet, Frits W. Vaandrager. *Automata with Timers*, FORMATS 2023. **Best paper award**.
2. Véronique Bruyère, Guillermo A. Pérez, Gaëtan Staquet. *Validating Streaming JSON Documents with Learned VPAs*, **TACAS** 2023.
3. Véronique Bruyère, Guillermo A. Pérez, Gaëtan Staquet. *Learning Realtime One-Counter Automata*, **TACAS** 2022.
4. Aziz Amezian El Khalfioui *et al. Optimization of Answer Set Programs for Consistent Query Answering by Means of First-Order Rewriting*, **CIKM** 2020.

Soumissions en cours :

1. Nathalie Bertrand, Patricia Bouyer, Gaëtan Staquet. *Antichains for Concurrent Parameterized Games*, soumis à MFCS 2025.
2. Véronique Bruyère, Guillermo A. Pérez, Bharat Garhewal, Gaëtan Staquet, Frits W. Vaandrager. *Active Learning of Mealy Machines with Timers*, soumis à QEST+FORMATS 2025.

Mes productions logicielles

Productions scientifiques :

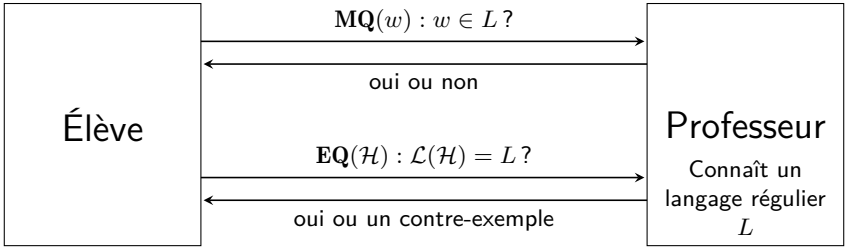
- Implémentation en **Python** pour [CIKM'20].
- Implémentations en **Java** pour [TACAS'22] et [TACAS'23].
  Seul développeur.
- Optimisation d'un code **Python** pour [SYNTCOMP].
- Implémentation en **C++** pour [Soumission MFCS].
  Seul développeur.
  Environ 5500 lignes de code en 1 mois.

Productions non-scientifiques :

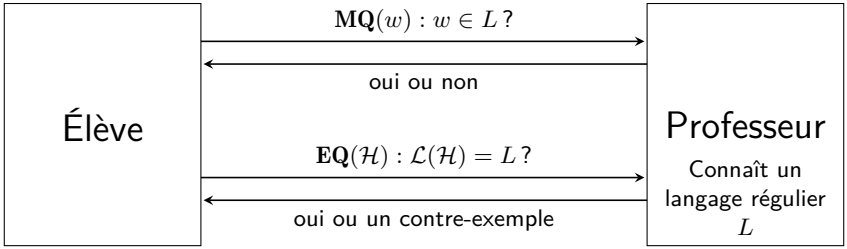
- Moteur de jeux vidéos en **C++**.
- Petits jeux vidéos en **C**.
- Mon site Internet en **Python** (nouvelle version en **Haskell** en cours d'écriture).
- Puzzles d'algorithmique, notamment avec **Haskell**.
- Concours de programmation en **Python** et **C++**.

Maîtrise de **plusieurs langages de programmation**, dans des **paradigmes différents**.

Thématique de recherche – Apprentissage actif d'automates



Thématique de recherche – Apprentissage actif d'automates



Cadre étendu à :

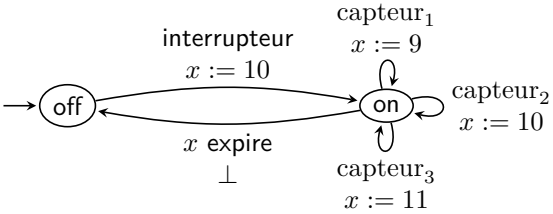
- Des automates à un compteur [TACAS 22].
- Des automates à minuteurs [Soumission QEST+FORMATS] (+ [FORMATS 23]).

Apprendre un automate à pile pour de la vérification de documents JSON [TACAS 23].

Exemple – automate **avec minuteurs**

Système d'extinction automatique d'une lampe :

- 10 minutes après appui sur l'interrupteur ;
- si présence détectée, à nouveau 10 minutes ;
- bugs : un capteur met la contrainte de temps à 9 minutes, et un autre à 11 minutes.



Horloges et minuteurs

Automates temporisés (horloges)

- Horloges de 0 à l'infini ;
- Accès aux valeurs des horloges ;
- Outils de model checking efficaces (UPPAAL, TChecker, IMITATOR) ;
- Plus expressifs ;
- Apprentissage coûteux (Waga (2023). Active Learning of Deterministic Timed Automata with Myhill-Nerode Style Characterization) ;
- Bien connus.

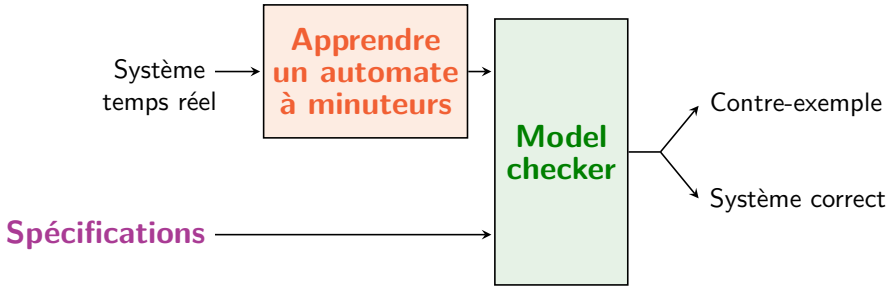
Automates à minuteurs

- Minuteurs d'une valeur donnée à 0 ;
- Seulement tests pour zéro ;
- Pas d'outils ;
- Plus restrictifs ;
- Apprentissage plus facile ([Soumission QEST+FORMATS]) ;
- Moins connus ([FORMATS'23]).

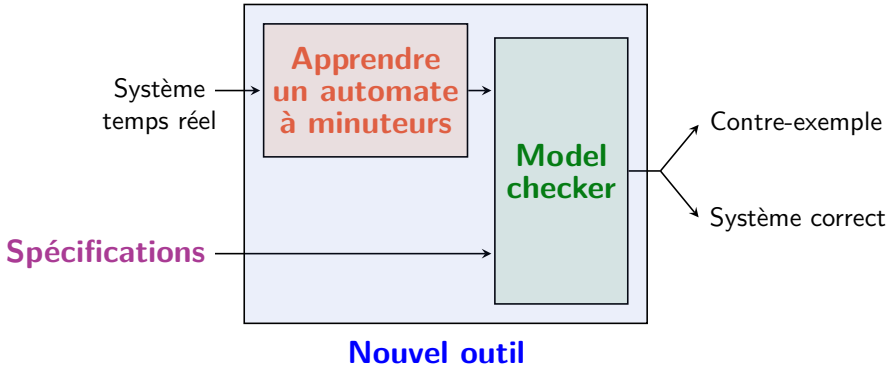
Problèmes ouverts

- Équivalence avec une sous-famille des automates temporisés ?
- Famille fermée par intersection, union, *etc.* ?
- Algorithmes spécifiques et efficaces en pratique.

Projet de recherche – Vue générale



Projet de recherche – Vue générale



Projet de recherche – Objectifs

Théorie des automates à minuteurs.

- Équivalence avec une sous-famille des automates temporisés ?
- Algorithmes spécifiques et plus efficaces en pratique.

Spécifications.

- Logiques existantes : TPTL, ECTL ?
- Nouvelle(s) logique(s).

Apprentissage.

- Modèles plus expressifs.
- Apprentissage robuste aux variations dans la mesure du temps.
- Implémentations.

Vérification.

- Étendre TChecker.

Model checking de systèmes temps réel : cas d'étude.

- Collection de benchmarks.
- Apprentissage de systèmes :

Protocoles réseaux ; Systèmes embarqués ; Systèmes d'exploitation temps-réel.

- Model checking.

Intégration au LRE – Équipe Automates et Applications

Équipe AA.

- Model checking ;
- Théorie des automates ;
- Développement d'outils.

Collaborations internes.

- Amazigh Amrane ;
- Adrien Pommellet ;
- Uli Fahrenberg ;

Collaborations externes.

- Nijmegen et Eindhoven (Pays-Bas) : Frits Vaandrager et Pieter Cuijpers.
- Mons et Anvers (Belgique) : Véronique Bruyère et Guillermo Pérez.
- Rennes et Paris-Saclay : Nathalie Bertrand et Patricia Bouyer.

Projets et pistes.

- Projet (i)Po(m)set.
- Automates à haute dimension.

Recherche – Synthèse

Parcours académique :

2024 – Présent Postdoctorat à Rennes

Août – Oct. 2023 Séjour à Nijmegen.

2020 – 2024 Doctorat à Mons et Anvers.

Activités collectives :

- Membre de plusieurs Conseils et Comités.
- Encadrement de 5 stagiaires.
- Comité évaluation artefact QEST+FORMATS 2025.
- Vulgarisation scientifique.

Recherche : Apprentissage et vérification de modèles pour des systèmes temps-réel.

Intégration Équipe Automates et Applications.

Collaborations Nijmegen, Eindhoven, Mons, Anvers, Paris-Saclay, Rennes.

Contributions :

- 4 conférences internationales, dont CIKM et TACAS (x2) ;
- 1 journal international : STTT ;
- 2 soumissions : MFCS, QEST+FORMATS ;
- Plusieurs réalisations logicielles.

Structures de Données (STD)

Buts du cours :

- Découvrir les structures de données principales.
- Être capable d'appréhender de nouvelles structures.
- Être capable d'implémenter de manière efficace.

Sources :

- Donald Knuth, *The Art of Computer Programming – Fundamental Algorithms*, Addison-Wesley, 1968.
- Robert Sedgewick et Kevin Wayne, *Algorithms, 4th Edition*, Addison-Wesley, 2011.
- Thibaut Balabonski *et al.*, *MP2I, MPI – Informatique, Cours et exercices corrigés*, ellipses, 2022.

Structures de Données (STD) – Structure et évaluations

Structure :

1. Introduction – 1h CM.
2. Structures linéaires (vectors, listes, files, piles) – 9h TD/TP.
3. Recherche (linéaire, dichotomique, arbre binaire de recherche) – 8h TD/TP.
4. Arbres (tries, B-trees) – 6h TD/TP.
5. Structures associatives et ensembles – 4h TD/TP.
6. Matrices – 2h TD/TP.

Évaluations :

- 50% TP et 50% examen (papier et machine).
- Types de questions :
  - Description d'un problème. Quelle structure de données ?
  - Un exemple de code. Quelle structure de données est implémentée ?
  - Un exemple de code. Trouvez l'erreur de logique.
  - Question ouverte : description d'une structure de données ou d'un algo à implémenter, *e.g.*, garbage collector via liste chaînée.

Structures de Données (STD) – Pédagogie

Pédagogie :

- Explications haut niveau au tableau.
- Exercices sur papier et machine.
  - Calcul complexité ;
  - Lister invariants ;
  - Implémentation efficace.
- Cadre de travail pour exécuter les tests et visualiser les résultats.
  - Correction ;
  - Temps d'exécution.

Structures de Données (STD) – Séquencement

1 - Introduction

- CM, 1h.
- Description structure :
  - Application programming interface (**API**) ; et
  - **Invariants**.
- En pratique :
  - **C++** avec **templates**.
  - API forcées par **concepts**.
  - TPs incrémentaux.

2 - Structures linéaires

- TD/TP, 9h.
- Complexité en temps et espace, récursivité, tableaux.
- Tableaux redimensionnables (**vector**) :
  - Ajout, insertion, suppression.
  - Combien de redimensionnements ? (Calcul papier et graphique machine)
  - Fonction pour la croissance de la capacité. $\rightsquigarrow$ Quelle fonction utiliser ?
- **Listes chaînées**.
- **Comparaisons** vectors et listes :
  - Insertions fin, début, aléatoire.
  - Quelle structure dans quel cas ? Pourquoi ?
- **Sacs** (bags), **files**, **piles**, **deques**. Vectors ou listes ?

Structures de Données (STD) – Séquencement

3 - Recherche

- TD/TP, 8h.
- Recherches **linéaire** et **dichotomique** sur tableaux et listes.
- **Arbres binaires de recherche**.
- Calculer distribution hauteur d'un arbre construit avec des valeurs aléatoires.
- Arbres **équilibrés** (rouge-noir).

5 - Structures associatives et ensembles

- TD/TP, 4h.
- Fonctions de hachage.
- **Structure associative** via :
  - Tables de hachage.
  - Arbre.
- **Ensemble**.

4 - Arbres

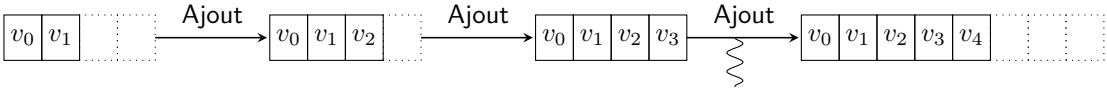
- TD/TP, 6h.
- Arbres avec k enfants.
- Arbres avec nombre d'enfants différents.
- Application : Arbres préfixes (**tries**) pour un lexique.
- **B-arbres**.

6 - Matrices

- TD/TP, 2h.
- Représentation matrices (tableaux/vectors vs listes).
- Matrices *sparse*.
- Application : résoudre un **système d'équations linéaires**.

Structures de Données (STD) – Vectors

- Tableaux **redimensionnables** :
  - Attention : **taille** (nombre d'éléments) et **capacité** (taille maximale).
  - **Copies** des valeurs quand capacité ne suffit plus.

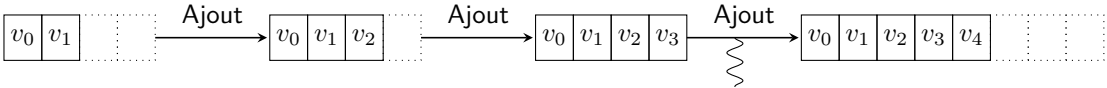


Capacité insuffisante !

1. Réserver nouvel espace mémoire.
2. Copier le tableau.
3. Libérer ancien espace mémoire.

Structures de Données (STD) – Vectors

- Tableaux **redimensionnables** :
  - Attention : **taille** (nombre d'éléments) et **capacité** (taille maximale).
  - **Copies** des valeurs quand capacité ne suffit plus.



1. Réserver nouvel espace mémoire.
2. Copier le tableau.
3. Libérer ancien espace mémoire.

Plusieurs possibilités pour la nouvelle capacité :

- ajouter 1 ;
- multiplier par 2 ;
- et bien d'autres.

- ajouter 2 ;
- multiplier par 10 ;

Structures de Données (STD) – Vectors – API

```
template <class T>
concept IsVector =
  // La fonction de croissance de la capacité est donnée à la construction
  std::is_constructible_v<T, std::function<int(int)>> and
  requires () {
    typename T::value_type;       // Type de valeurs stockées
    typename T::reference;        // value_type&
    typename T::const_reference;  // const value_type&
  } and requires(const T vector) { // Ces méthodes doivent être const
    { vector.size() } -> std::same_as<std::size_t>; // Nombre de valeurs stockées
    { vector.capacity() } -> std::same_as<std::size_t>; // Taille maximale
  } and requires (T vector, std::size_t i) {
    { vector.reserve(i) } -> std::same_as<void>; // Si capacité < i, alors capacité <- i
    { vector.at(i) } -> std::same_as<typename T::reference>; // Référence vers valeur à position i
    { vector[i] } -> std::same_as<typename T::reference>; // Référence vers valeur à position i
  } and requires (const T vector, std::size_t i) { // Ces méthodes doivent être const
    { vector.at(i) } -> std::same_as<typename T::const_reference>; // Const ref vers valeur à position
    { vector[i] } -> std::same_as<typename T::const_reference>; // Const ref vers valeur à position i
  } and requires(T vector, const typename T::value_type &value) {
    { vector.push_back(value) } -> std::same_as<void>; // Ajoute un élément à la fin
  }
;
```

Dans les fichiers : commentaires plus détaillés et des exemples.

Structures de Données (STD) – Vectors – Exemple

```
template <typename T>
class vector {
public:
  using value_type = T;
  using reference = value_type&;
  using const_reference = const value_type&;
  vector(std::function<int(int)>
  ↪  growthFunction);
  std::size_t size() const;
  std::size_t capacity() const;
  void reserve(std::size_t size);
  reference at(std::size_t i);
  reference operator[](std::size_t i);
  const_reference at(std::size_t i) const;
  const_reference operator[](std::size_t i)
  ↪  const;
  void push_back(const value_type &value);
};
static_assert(IsVector<vector<int>>);
```

Fonctions disponibles :

- `test_results vectors::test<V>()` pour tester la correction ;
- `benchmark_results vectors::benchmark<V>()` pour évaluer performances ;
- `plot_data new_plot()`, `plot_data::add(benchmark_results)` et `plot_data::save(std::filesystem::path)` pour avoir des graphiques des performances.

Structures de Données (STD) – Vectors – Questions

1. Sur papier : calculez le nombre de redimensionnements lorsqu'on ajoute N valeurs et que la nouvelle capacité est
   - 1.1 la capacité actuelle $+1$;
   - 1.2 la capacité actuelle $+2$;
   - 1.3 le double de la capacité actuelle.
2. Implémentez `vector<T>`.
3. Créez un graphique du nombre de redimensionnements. Comparez avec vos calculs.

Mon expérience d'enseignement – **251h** équivalent TD

Programmation et Algorithmique :

- **Typologie des langages**, L3 (EPITA), 8h CM et 4h TD, 2024 – 2025.
- **Algorithmique et Complexité Expérimentale**, L1 (ISTIC), 28,5h TP, 2024 – 2025.
- **Programmation Logique** (Prolog) et **Programmation Fonctionnelle** (Scheme), L3 (UMONS), 42h et 48h TD, 2021 – 2022 à 2023 – 2024.
- **Programmation et Algorithmique I** (Python) et **Programmation et Algorithmique II** (Java), L1 (UMONS), 80h et 80h TP, 2018 – 2019 et 2019 – 2020.

Intelligence artificielle :

- **Intelligence artificielle et Machine Learning**, L3 (EPITA), CM et TD, 2024 – 2025.

Implication personnelle :

- Préparer et corriger des examens.
- Co-encadrer des mémoires de master.
- Évaluer et assister aux soutenances de 11 mémoires de master.

Intégration à l'EPITA – Enseignements

Enseigner sur les trois années :

- Informatique théorique ;
- Programmation et algorithmique ;
- Intelligence artificielle ;
- Enseignements en anglais (bachelors).

Me former pour élargir mes possibilités d'enseignement :

- Compilation ;
- Systèmes d'exploitation.

Approche pédagogique :

- Outils de visualisation et de manipulation pour les enseignements théoriques.
- Développement d'un outil **générique** pour visualisation des calculs d'automates pour **Théorie des Langages**.
  Dès le premier semestre de la **deuxième année**.
- Implémentations modulaires, prévues pour être facilement adaptées.

Intégration à l'EPITA – Responsabilités

Prendre des responsabilités :

- Encadrement d'élèves.
- Suivis de stage.
- Implication dans la vie de l'EPITA.

Féminisation et inclusion :

- Mettre en évidence des femmes avec un parcours inspirant.
- Rendre les études plus visibles
- Environnement bienveillant (réunions en **non-mixité** ou **mixité choisie**).

Enseignement – Synthèse

Enseignement :

- 251h équivalent TD.
- Capacité d'enseigner dès la rentrée 2025 :
  - Informatique théorique ;
  - Programmation et algorithmique ;
  - Intelligence artificielle.
- Responsabilités envisagées :
  - Encadrement d'élèves ;
  - Suivis de stage.
  - Implication dans la vie de l'EPITA.
  - Féminisation et inclusion.
- Pédagogie via des outils visuels.