Haut Conseil de l'Évaluation de la Recherche et de l'Enseignement Supérieur

DOCUMENT D'AUTOÉVALUATION

Équipe APR

Campagne d'évaluation 2023-2024 — Vague D

Table des matières

1 INFORMATIONS GÉNÉRALES SUR L'ÉQUIPE APR

Nom de l'équipe : Algorithmes, Programmes et Résolution (APR)

Responsable de l'équipe : Antoine Miné (depuis le 01/01/2019)
Emmanuel Chailloux (jusqu'au 31/12/2018)

	2017	2018	2019	2020	2021	2022
PR	3	3	3	3	4	4
MCF HDR	0	1	1	1	1	1
MCF	5	5	5	5	5	4
DR	1	1	1	1	1	2
CR HDR	0	0	1	1	1	0
CR	1	1	1	1	1	1
Total permanents	10	11	12	12	13	12
Émérites	2	2	2	2	2	3
Doctorants	11	11	11	8	11	11
Ingénieurs CDD ou hors tutelles	1	0	1	0	2	0
Post-doc, ATER, etc.	1	2	2	4	4	5
Stagiaires	3	7	8	7	5	4
Total non permanents	16	20	22	19	22	20
Total avec émérites	**28**	**33**	**36**	**33**	**37**	**35**
Equivalent temps plein recherche	6.0	6.5	7.5	7.5	8.0	7.5

TABLE 1 – Personnels APR sur la période 2017-2022 (au 1er juillet de chaque année)

1.1 Les thématiques scientifiques et leurs enjeux

Les thématiques de l'équipe APR se développent autour de l'étude des algorithmes, des structures combinatoires, des langages de programmation et des logiciels, du calcul formel et de l'analyse topologique de données. L'équipe trouve une cohérence thématique dans l'emploi d'approches formelles basées sur des fondations mathématiques solides, que ce soit en combinatoire, en topologie ou en sémantique. Ces approches sont déclinées en résultats théoriques et appliqués dans des domaines variés.

En programmation, les thèmes principaux concernent la compilation et la vérification par typage, par interprétation abstraite, par preuve de théorèmes, par raffinement.

En algorithmique, l'équipe est spécialiste de combinatoire analytique, d'algorithmes de graphes, de topologie computationnelle. Un aspect fédérateur est le développement de méthodes ayant pour but l'amélioration de la qualité, la performance et la fiabilité des logiciels, avec des applications depuis les logiciels embarqués jusqu'aux applications web, en passant par les logiciels système, la programmation concurrente et parallèle, les langages dynamiques, etc. Ces buts sont atteints par des réalisations logicielles matures, visant des objectifs concrets.

Nos travaux sont présentés dans des conférences spécialisées, mais aussi dans nos séminaires et journées d'équipe, où les différentes thématiques peuvent se rejoindre et s'enrichir.

La variété des travaux et notre capacité à communiquer a permis de développer, par des collaborations au sein de l'équipe, des thèmes novateurs à la frontière des expertises de plusieurs chercheurs, comme les applications de la combinatoire (algorithmique) au test de programme (vérification).

Systèmes concurrents et combinatoire

Dans le cadre d'un projet de recherche à long terme initié en 2011 et d'une collaboration à l'international, notre équipe étudie les systèmes concurrents à travers le prisme de la combinatoire (analytique, énumérative et bijective).

Études quantitatives et algorithmes de génération aléatoire. Nos objectifs sont multiples. Dans un premier temps, il s'agit de fournir une interprétation combinatoire des phénomènes concurrents : parallélisme, non-déterminisme, modèles de synchronisation, etc. Dans un second temps, nous étudions les aspects quantitatifs de ces modèles, en nous focalisant notamment sur l'étude du phénomène bien connu de l'"explosion combinatoire" du nombre d'états dans les systèmes concurrents. Enfin, nous concevons des algorithmes basés sur nos études théoriques, en particulier dans le domaine de la génération aléatoire uniforme. Nous nous intéressons d'un côté à la génération de programmes concurrents (permettant de tester les compilateurs et environnements d'exécution), et de l'autre côté à la génération de *trajectoires* (exécutions possibles) permettant de développer des

analyses statistiques sur les programmes concurrents. Ces analyses sont effectuées sans construction explicite de l'espace d'état des programmes, ce qui rejoint donc les travaux de l'équipe en analyse statique de programmes. Dans ce cadre, nous avons par exemple mis en lumière dans [Dien et al., 2022] une bijection entre le modèle de synchronisation *async/await* et la combinatoire des *ordres d'intervalle*. Nous avons développé des algorithmes de génération aléatoire uniforme de structures et de trajectoires pour cette classe de processus.

Combinatoire des structures croissantes. Nos études quantitatives nous ont permis d'établir des liens étroits entre les phénomènes de concurrence et la combinatoire des structures à étiquetage croissant. En nous basant sur cette interprétation, nous avons obtenu des résultats importants en combinatoire analytique, notamment en contribuant de nouveaux opérateurs pour composer les structures et leur étiquetage [Bodini et al., 2017b]. Nous avons prolongé ces travaux en étudiant des processus de croissance plus généraux, permettant d'encoder de nombreux types de structures avec contraintes de croissance. Ces classes d'objets ne sont souvent pas récursivement décomposables et nos travaux [Genitrini et al., 2020a, Bodini et al., 2020a] font appel à des concepts comme les équations aux dérivées partielles, qui sont mal connues et peu étudiées en combinatoire. Le phénomène de synchronisation, étudié en particulier dans la thèse de Matthieu Dien, correspond à l'étiquetage croissant de graphes dirigés acycliques (DAGs). La combinatoire des DAGs (et des ordres partiels, qui sont intimement liés) est complexe, et nous nous sommes principalement reposés sur la combinatoire énumérative et bijective pour réaliser cette partie de nos études.

Dans le cadre de la thèse soutenue récemment par Martin Pépin, nous avons pour la première fois pu réunir nos différentes interprétations combinatoires pour étudier un langage concurrent proprement dit, à la fois sous ses aspects combinatoires, mais également d'un point de vue pratique en développant un prototype d'analyseur statistique. La publication principale concernant cette étude [Genitrini et al., 2022] est décrite au portfolio 2.

Analyse topologique de données

En analyse topologique de données, l'équipe se focalise sur deux aspects principaux :

1. les algorithmes haute-performance pour le calcul de signatures topologiques de données ;
2. la définition d'un cadre computationnel pour l'analyse statistique de signatures topologiques.

Ces recherches sont menées dans le cadre du projet ERC TORI.[1] Elles ont donné lieu à plusieurs résultats significatifs. Les algorithmes développés dans cette recherche sont maintenant la référence pour les calculs d'arbres de jointure [Gueunet et al., 2019a], de graphes de Reeb [Gueunet et al., 2019b] et de diagrammes de persistance [Guillou et al., 2023]. L'équipe a notamment développé de nouveaux outils pour l'analyse statistique de signatures topologiques [Vidal et al., 2019, Pont et al., 2021], avec récemment le premier cadre computationnel pour l'analyse par géodésiques principales de signatures topologiques [Pont et al., 2022]. Les implantations de tous ces algorithmes sont disponibles au sein de la bibliothèque en logiciel libre TTK [Tierny et al., 2017]. Ces contributions ont été saluées par la communauté scientifique compétente, avec plusieurs paper awards [Tierny et al., 2017, Soler et al., 2018a, Soler et al., 2019, Vidal et al., 2019], notamment à IEEE VIS (la conférence internationale de référence en visualisation et analyse de données).

Ces contributions ont par ailleurs été appliquées dans des domaines applicatifs variés, en collaboration avec des experts de ces domaines, notamment en chimie théorique [Olejniczak and Tierny, 2023, Olejniczak et al., 2019], en physique des matériaux [Soler et al., 2019, Lukasczyk et al., 2017] et en mécanique des fluides [Bridel-Bertomeu et al., 2019, Nauleau et al., 2022]. Un effort de diffusion important sur TTK est réalisé, notamment grâce la réalisation de tutoriels et de vidéos (portfolio 1).

Algorithmique de graphe. En algorithmique de graphes, nos travaux portent sur deux sujets mutuellement inclusifs :

1. l'extension de l'algorithmique de graphes (statiques) aux graphes temporels ;
2. la restriction des graphes à ceux exploitant une certaine structure géométrique.

À la suite des résultats polynomiaux dans l'algorithmique des chemins temporels [3] (400 citations), une question naturelle était d'étendre cette polynomialité aux autres versions temporelles des cas populaires, tels que les composantes connexes, arbres couvrants ou flots maximums. Malheureusement, l'absence de résultats dans ce sens depuis 20 ans, couplée à de nombreux résultats de NP-difficulté des cas cités ci-dessus, ont transformé cette question naturelle en une forme de défi en complexité. Sur ce sujet, nous avons montré un schéma d'approximation en temps polynomial (PTAS) résolvant un problème d'ordonnancement appelé CouplageTemporel dans les graphes temporels imitant les *unit disk graphs* [Picavet et al., 2021]. Ce problème a été introduit par

1. https://erc-tori.github.io/

notre équipe en 2018 comme une généralisation naturelle du problème de couplage maximum. Il est malheureusement NP-difficile aussi bien dans le cas général que dans des cas extrêmement restreints de graphes [7]. C'est dans ce sens que notre résultat de PTAS est important pour une application dans des cas d'utilisation pratique venant de nos collègues industriels. En examinant les précalculs polynomiaux possibles, nous avons observé qu'une énumération en amont d'objets appelés modules temporaires aident à résoudre plus rapidement COUPLAGETEMPOREL. Nous avons introduit le problème MODULETEMPOREL ainsi que des algorithmes polynomiaux pour résoudre complètement le cas où ceux-ci sont de taille 2, et partiellement le cas général [Bui-Xuan et al., 2022]. Tous ces algorithmes sont implantés dans des prototypes en logiciel libre et confrontés à des jeux de tests publiquement accessibles.

Langages de haut niveau pour matériels à faibles ressources

L'équipe est spécialiste des modèles de concurrence, tant pour l'expressivité et la sûreté (extensions réactives pour la programmation Web [6], projet FUI UCF) que pour l'efficacité dans la programmation fiable pour les multi-cœurs et les GPU ([2], projet FUI OpenGPU). Elle s'intéresse maintenant aux matériels à faibles ressources (microcontrôleurs, FPGA) pour utiliser pleinement leur puissance tout en leur apportant sûreté et expressivité, sans perdre en efficacité. Elle développe des approches complémentaires : langages typés de haut niveau avec garanties de fiabilité, "machines virtuelles" portables, intégration du modèle de programmation synchrone et analyse statique.

Approche machine virtuelle. Après avoir développé une machine virtuelle OCaml en assembleur PIC [9], nous avons conçu une machinerie virtuelle plus portable, OMicroB ([Varoumas et al., 2018], portfolio 5), qui construit un programme C à partir du code-octet engendré par le compilateur et qui intègre la bibliothèque d'exécution. Cela permet de bénéficier de l'intégralité du langage OCaml, de son typage statique et de la gestion sûre et automatique de la mémoire, avec une empreinte mémoire faible (4Ko de tas et 32Ko de code).

Approche synchrone. Les applications sur microcontrôleurs sont en interaction continue avec leur environnement, réagissent aux signaux en entrée et produisent des signaux en sortie. Pour gérer cette interaction, nous avons intégré le modèle synchrone (à la Esterel et Lustre) à un cadre fonctionnel/impératif typé. Nous avons développé OCaLustre [Varoumas et al., 2020], une extension synchrone à flots de données d'OCaml, compilée et exécutée avec OMicroB. Ce modèle simple et économe pour la programmation des comportements concurrents d'un système embarqué permet de garantir le déterminisme de la partie concurrente interne de l'application.

Circuits reconfigurables (FPGA). Exploitant les travaux ci-dessus, nous avons proposé un environnement de programmation sûre, expressive et efficace pour les FPGA, vus comme accélérateurs de calculs mais aussi comme applications autonomes. Nous avons décliné l'approche machine virtuelle d'OMicroB en un système, O2B, sur un *softcore* (processeur décrit en FPGA), pouvant exécuter tout programme OCaml ainsi que des circuits conçus manuellement. Afin d'accélérer les calculs du langage de haut niveau, un sous-ensemble fonctionnel/impératif d'OCaml, appelé Macle [Sylvestre et al., 2022a], peut à la demande engendrer une description VHDL, ensuite synthétisée en circuit.

Analyse de programmes et interprétation abstraite. APR développe des méthodes d'analyse statique sémantique basées sur l'interprétation abstraite [5] pour la vérification formelle de la correction des programmes. L'interprétation abstraite a connu ces deux dernières décades des succès académiques et industriels importants (comme l'outil commercialisé Astrée [1], qui constitue l'état de l'art en analyse statique des erreurs à l'exécution des programmes embarqués critiques C). Des outils ciblant des programmes plus généraux ont été proposés plus récemment, comme l'analyseur industriel Infer [4] (Méta), mais ces extensions se font au détriment des garanties de sûreté. Par contraste, notre but est l'extension de l'interprétation abstraite appliquée aux programmes réalistes pour de nouvelles cibles (nouveaux langages, nouvelles propriétés, nouvelles architectures) *sans concession aux garanties formelles de sûreté*.

Vérification des programmes concurrents. Un thème fédérateur de l'équipe est la concurrence, que ce soit en compilation, en analyse d'algorithme ou en vérification. En 2017, l'équipe achevait une collaboration avec l'ENS et l'industrie pour l'extension d'Astrée aux logiciels embarqués parallèles en C [Kästner et al., 2017]. Dans des travaux plus académiques, nous avons développé des analyses sûres pour les mémoires à faible cohérence [Suzanne and Miné, 2018], qui sont la norme dans les processeurs multi-cœurs. Nous avons également développé l'analyse des programmes concurrents à base de processus communicants avec création/destruction dynamique de processus [Botbol et al., 2017], et avec des applications à la sûreté numérique des programmes distribués utilisant la bibliothèque MPI/C.

Extension des langages et propriétés analysés. L'ERC MOPSA (2016–2022) marque le début d'un travail sou-

tenu, et qui se poursuit, sur l'extension de la théorie et des méthodes d'implantation des interprètes abstraits. Un axe majeur concerne l'analyse de nouveaux langages. Nous ciblons les langages dynamiques, avec l'exemple de Python [Monat et al., 2020], dont l'analyse sûre est rendue difficile par une sémantique très riche et complexe, mais devient un sujet important par l'explosion du nombre d'applications employant ces langages. Nos solutions se basent sur des abstractions dédiées allant au-delà de l'état de l'art en terme de précision et d'expressivité (prise en compte du flot de contrôle, propriétés relationnelles). Nous avons proposé le premier interprète abstrait *multi-langages* capable d'analyser des programmes réels mélangeant code C et Python [Monat et al., 2021]. Il s'agit d'un enjeu important puisque les applications modernes sont souvent multi-langages. Nous avons également appliqué l'interprétation abstraite à l'analyse de *smart contracts* [Bau et al., 2022] sur la *blockchain* Tezos, qui propose un modèle d'exécution original et peu étudié sur le plan sémantique. Un deuxième axe concerne l'extension des propriétés vérifiées. Nous avons développé des analyses de non-régression et de portabilité [Delmas et al., 2021]. Sur le plan théorique, il s'agit de propriétés d'*hypersafety*, notoirement plus complexes à appréhender que les propriétés classiques de sûreté. Sur le plan pratique, l'analyse de portabilité a été appliquée à la vérification de codes avioniques industriels d'une taille de 1 MLOC. Ces résultats sont implantés dans une plateforme en logiciel libre, MOPSA ([Journault et al., 2019a], portfolio 3), que nous développons, et qui a pour but de présenter aux chercheurs et ingénieurs les résultats de notre recherche de manière directement applicable, mais aussi d'aider à l'enseignement fondamental et à la formation pratique en interprétation abstraite. La plateforme est accompagnée d'exemples d'analyse et nous avons également publié un tutoriel sur l'analyse numérique par interprétation abstraite [Miné, 2017].

Temps d'exécution et consommation mémoire dans des matériels à faibles ressources. L'analyse statique est utilisée pour borner statiquement le temps d'exécution et la consommation mémoire. Elle est appliquée à la machine virtuelle OMicroB [Varoumas and Crolard, 2019] et exploite les caractéristiques des microcontrôleurs ciblés (prédiction de branchement, pipe-line et caches simples) pour raisonner au niveau du code-octet. Pour les programmes OCaLustre, une borne mémoire des flots de données est déterminée à la compilation par l'état global (ensemble des nœuds déclarés par l'extension synchrone du programme). Dans le cadre général de programmes avec allocation dynamique, il est difficile de savoir quelles zones mémoires sont libérables statiquement. Un travail a été initié pour fournir un cadre générique pour l'analyse de la consommation mémoire dans les langages fonctionnels/impératifs par analyse amortie automatisée des ressources, en utilisant une approche par inférence de types sur une machine abstraite *Call-By-Push-Value* ainsi qu'un mécanisme de régions pour supporter les références mutables. Un prototype, AutoBill, est en cours de finition.

Combinaison de la programmation par contraintes et de l'interprétation abstraite. Les liens entre méthodes de résolution et analyse de programmes sont riches. Alors que beaucoup de travaux exploitent les solveurs SAT et SMT, l'équipe étudie un lien plus original : celui entre interprétation abstraite et programmation par contraintes, des domaines ayant des communautés bien distinctes et qui communiquent peu. Au sein du projet ANR Coverif (2015–2019), à travers une thèse co-encadrée et un échange de post-doctorant, nous avons pu combiner nos compétences en interprétation abstraite avec celles en programmation par contraintes du LINA (Université de Nantes) pour développer des méthodes de résolutions hybrides originales [Ziat et al., 2018] et participer à la réalisation d'un outil pratique (solveur AbSolute), exploitant les domaines numériques abstraits relationnels dont l'équipe est spécialiste ([Miné, 2017], octogones [8], polyèdres). Ce travail s'est poursuivi en considérant l'abstraction par zonotopes, en collaboration avec l'École polytechnique, qui développe cette abstraction [Kabi et al., 2020].

Langages de programmation et sémantique

Spécifications et preuves de programmes impératifs. Nous avons développé une approche de programmation certifiée correcte par raffinement, en exploitant le cadre formel de l'assistant de preuve Coq [Sall et al., 2019]. Partant d'un langage impératif simple, nous avons défini une sémantique relationnelle exprimée dans un cadre prédicatif et pouvant être plongée dans la théorie des types. Afin que l'articulation des étapes de raffinement reste aussi proche que possible de l'activité de programmation, nous introduisons un "langage de développement" pour décrire l'arborescence des étapes de raffinement, puis une logique pour raisonner sur ces développements et garantir leur correction. Cette théorie de la programmation par raffinements a été entièrement mécanisée en Coq, ce qui permet au programmeur de vérifier ses programmes en profitant des mécanismes d'automatisation de l'assistant de preuve.

Sémantique des langages réactifs et probabilistes. En sémantique formelle, nous avons exploré trois directions intimement liées :

1. les langages de programmation réactive, permettant de développer une théorie des types pour des pro-

grammes synchronisés à des rythmes différents ;

2. les langages de programmation probabiliste, donnant des outils formels permettant d'étudier les langages et les programmes représentant des modèles bayésiens ;
3. l'étude de la substitution linéaire et non linéaire, permettant d'étudier la consommation de différents types de ressources lors de l'exécution des programmes.

Ces recherches sont menées dans le cadre du projet Émergence de la ville de Paris ReaLiSe portant sur la sémantique des langages de programmation réactive et probabilisite et de l'ANR JCJC portant sur la sémantique et les espèces de structure. Si elles sont intéressantes indépendemment, ces trois directions sont étroitement liées. Par exemple, le typage des horloges dans le cadre réactif fait apparaître deux types de ressources : les variables usuelles, qui correspondent à la mémoire, et les variables d'horloge, qui permettent de synchroniser les programmes à des rythmes différents. Ainsi, les outils formels développés pour étudier la substitution linéaire et non-linéaire guident le développement du langage réactif étudié. De même, dans les langages probabilistes, l'indépendance probabiliste est étroitement liée à la linéarité, et des variables non-probabilistes cotoient des variables probabilistes que l'on ne peut échantilloner qu'une seule fois. Enfin, la sémantique probabiliste a permis d'étudier un algorithme d'inférence adapté au cadre réactif dans lequel il est nécessaire de séparer les paramètres constants des paramètres qui varient au cours du temps et de prouver la correction de la pré-complilation nécessaire à l'application de cet algorithme. Il s'agit d'une thématique récente dans l'équipe, avec une publication sous presse [Baudart et al., 2023].

Complexité causale de la programmation distribuée. La thématique de recherche d'APR concernant la théorie des systèmes concurrents a également ciblé, durant la période d'évaluation, la complexité causale dans les algèbres de processus [Demangeon and Yoshida, 2023] (portfolio 6). En rapprochant des techniques issues du domaine de la complexité implicite et des méthodes formelles classiques pour la programmation (systèmes de types et parcours du code), une notion de complexité *en nombre de messages* des protocoles distribués récursifs (par exemple des protocoles clients-serveurs dans lesquels les serveurs peuvent récursivement s'appeler eux-mêmes) peut être définie, et une analyse basée sur la mise en évidence de dépendances entre les primitives d'envoi et de réception de messages dans du code récursif garantit le comportement polynomial d'une application distribuée.

Calcul formel et applications. L'équipe s'attache au développement d'algorithmes en calcul formel, numérique et symbolique, avec une thématique importante autour des nombres flous (une thèse soutenue), à l'étude des plans d'expériences et des systèmes dynamiques discrets ayant des applications dans le monde de la santé.

Calcul formel en santé. En collaboration avec M. Koné, nous avons généralisé les plans d'expériences pour les erreurs auto-régressives à tout $m-$intervalle de temps. Les derniers résultats dans ce domaine dataient de 1993 (par Grondona et Cressi) et concernaient le cas $m = 2$. Le traitement du cas général se heurtait à des difficultés combinatoires qui ont du être surmontées. Une application en santé est la détermination des meilleurs plans pour administrer des traitements à des patients, mais des liens avec d'autres domaines existent (comme les codes correcteurs d'erreurs).

En collaboration avec l'Institut Pasteur de Tunis, nous avons étudié [Abdeljaoued-Tej et al., 2019] les systèmes dynamiques discrets pour la modélisation des voies cellulaires affectées dans le cancer et le ciblage des thérapies. Ces systèmes apparaissent également dans les réseaux neuronaux (celui des activations et celui des poids). Pour ces systèmes, nous nous intéressons à calculer des séparateurs polynomiaux dans des structures discrètes, dans le cas des petites dimensions.

Résolution. Nous avons établi un algorithme général et parallèle [Aubry et al., 2020] pour résoudre des systèmes polynomiaux dont les coefficients sont des nombres flous. Nous avons établi des résultats "subtils" pour éviter de nombreux calculs inutiles que nous avons détectés comme redondants, ou pouvant se déduire d'autres calculs avec un théorème portant sur les signes. Ce résultat significatif est détaillé au portfolio 4.

Les résolutions numériques sont plus efficaces que les résolutions algébriques, mais elles sont approximatives et doivent être répétées à chaque modification des données. Pouvoir certifier algébriquement un résultat permet alors des gains importants. Nous avons établi dans [Abdeljaoued-Tej et al., 2019] une certification basée sur la théorie de Galois pour certifier, avec un test à zéro "partiel", des calculs réalisés avec des approximations numériques des racines d'un polynôme d'une variable.

2 INTRODUCTION DU PORTFOLIO

Les six éléments du portfolio ont été choisis pour mettre en valeur la diversité des thèmes de recherche abordés par l'équipe en montrant quelques réalisations et résultats d'intérêt en langage, en programmation, en algorithmique, en méthodes formelles, en analyse topologique de données, sans toutefois couvrir tous les thèmes de l'équipe. Ces éléments ont également pour but de montrer la variété des activités que cette recherche engendre : développements théoriques publiés sous forme d'articles, implantations, diffusion auprès de la communauté. Chaque élément disposant d'une fiche explicative détaillée, nous listons simplement ici ces éléments :

- *Élément 1 (vidéo)* : cette vidéo de présentation de la bibliothèque en logiciel libre TTK (Topology ToolKit) est un exemple de l'important effort de diffusion logiciel ; elle illustre l'axe de recherche en analyse topologique de données et l'ERC TORI.
- *Élément 2 (article)* : *A quantitative study of fork-join processes with non-deterministic choice : application to the statistical exploration of the state-space*. Cet article, publié dans Theoretical Computer Science, est un exemple de résultat original de recherche combinant des aspects algorithmiques (combinatoire analytique) et programmation (vérification de programmes), grâce à la collaboration au sein de l'équipe d'experts respectifs dans chacun de ces domaines.
- *Élément 3 (logiciel ou bibliothèque logicielle)* : plateforme MOPSA pour l'analyse statique par interprétation abstraite. Cet élément présente l'important développement logiciel issu du projet ERC MOPSA. Il illustre la recherche de l'équipe en interprétation abstraite et le développement en logiciel libre d'outils effectifs de vérification formelle.
- *Élément 4 (article)* : *Computing real solutions of fuzzy polynomial systems*. Cet article, publié dans Fuzzy Sets and Systems, illustre la recherche de l'équipe en calcul formel aboutissant à des implantations au sein de logiciels reconnus (SageMath).
- *Élément 5 (logiciel ou bibliothèque logicielle)* : environnement de développement OMicroB pour la programmation de microcontrôleurs. Cet ensemble de logiciels illustre les implantations associées aux recherches de l'équipe dans le domaine de l'embarqué et le portage des langages de haut-niveau (OCaml) sur des plateformes à faibles ressources.
- *Élément 6 (article)* : *Causal Computational Complexity of Distributed Processes*. Cet article, publié dans Information and Computation, illustre l'axe de recherche de l'équipe en typage avec des applications à la détermination de la complexité des systèmes de processus. Ce travail présente également un échantillon des coopérations internationales de l'équipe (ici, avec le Royaume-Uni).

3 AUTOÉVALUATION DU BILAN

3.1 Autoévaluation de l'équipe

Domaine 2. Attractivité

Référence 1. L'unité est attractive par son rayonnement scientifique et s'insère dans l'espace européen de la recherche.

Visibilité par projet. Un témoin de notre visibilité à l'échelle européenne est l'obtention de *deux bourses ERC Consolidator Grant* : MOPSA en vérification des logiciels et TORI en analyse topologique de données. Nous sommes également sollicités pour participer à de grands projets européens : *ITEA 3* Assume, *H2020-FET* VESTEC, et français : *PEPR* Cybersécurité Secureval. Les projets sont détaillés à la référence 3.

Invitations et prix. Outre la publication dans des conférences et revues internationales de premier rang (IEEE TVCG, TCS, ESOP, . . .voir les détails au domaine 3), nous sommes régulièrement invités à des exposés, des keynotes, des tutoriels à ces workshops et conférences : SIGGRAPH, EuroVis VISGAP, JFIG, le GdR IM, les workshops SYCO, CIE, ICALP, etc., des séminaires *Dagstuhl* et *NII Shonan*. L'une d'entre nous était invitée à un *séminaire du Collège de France*. Plusieurs de nos articles sont primés par des *prix du meilleur article* : workshop SOAP [Monat et al., 2020], conférences IEEE VIS [Vidal et al., 2019] et LDAV [Soler et al., 2019]. L'équipe a *deux lauréats de l'IUF* comme membres Juniors.

Logiciels. Un autre indice de visibilité est la diffusion de logiciels dont l'équipe est développeur principal, comme *TTK* (portfolio 1) ou *MOPSA* (portfolio 3). Cet aspect est détaillé à la référence 4.

Animation scientifique. Les membres de l'équipe participent au *comité de programme* de conférences internationales de référence dans les domaines de la visualisation et l'analyse de données (IEEE VIS 2017–2019, 2021–2022, EuroVis 2015–2017, 2020–2022), de la vérification (CAV 2022, SAS 2019–2022), des langages de programmation (ESOP 2020, IPDPS 2018–2020), de la programmation orientée services (ACM SAC-SOAP 2017–2019). Ils sont *chairs* dans des conférences internationales : *FOSSACS 2021* (conférence majeure sur les fondements en science des logiciels), *MFPS 2022* (conférence spécialisée en sémantique des langages), *LDAV* (conférence IEEE spécialisée en *big data* et visualisation). Un membre d'APR est *éditeur associé à IEEE TVCG* (revue internationale de référence en visualisation et analyse de données). Ils sont sollicités régulièrement pour relire des articles de revue et rédiger des notices bibliographiques (Mathematical Reviews de l'AMS de 2019 à 2022).

Ils organisent des workshops internationaux plus spécialisés ou portant sur des thèmes émergeants : *LAFI* (langages de programmation et apprentissage) en 2022 et 2023, *TopoInVis* (analyse visualisation de données) en 2019 et 2022, *EGPGV* (graphismes parallèles) de 2017 à 2020, et des évènements internationaux ponctuels : la semaine *Logique of Probabilistic Programming* et le workshop *Combining Abstract Interpretation and Constraint Programming* au *CIRM*, un *workshop Dagstuhl* sur l'interprétation abstraite. Un membre a participé au *comité d'organisation de POPL 2017* (une des conférences les plus importantes en langages de programmation, avec plus de 700 inscrits).

Coopérations internationales. Nous collaborons avec des collègues à l'international : en Italie, au Royaume-Uni, en Espagne, au Portugal, en Autriche, en Pologne, en Norvège, en Tunisie, au Vietnam, en Uruguay, au Canada, à Taiwan. Ces collaborations se traduisent notamment par des échanges d'étudiants (recrutement de deux doctorants italiens), des articles rédigés en commun (articles dans les revues Random Structures and Algorithms [Bodini et al., 2022a], Combinatorial Theory Series A [Genitrini et al., 2020a], Journal of Algebra and Its Applications, conférence Euro'Comb 2021 [Bui-Xuan et al., 2021]), ainsi que des invitations dans des institutions académiques (Vietnam Institute of Mathematics, Bergen University, Polytechnic University Valencia, LIA Uruguay).

Pilotage de la recherche. Nous participons au pilotage national et local de la recherche : L'un de nous a été *responsable du GT Visualisation du GdR CNRS IGRV*; Une collègue est *élue au conseil scientifique de l'INS2I*; Un autre membre de l'équipe est *membre élu de la section 6 du CoNRS* (2021–2026). Une collègue était *membre élue de la section 27 du CNU* (2011–2019). Au sein du LIP6, le responsable de l'équipe est aussi est co-animateur de l'axe "Sûreté, Sécurité et Fiabilité" (2019–) et un autre de l'axe "Théorie et Outils Mathématiques" (2019–2021) ; l'une de nous est membre de la commission de soutien à la publication depuis 2014 ; Nous comptons deux collèges qui ont été membres du conseil de laboratoire jusqu'en 2019, puis à partir de 2019.

Référence 2. L'unité est attractive par la qualité de sa politique d'accompagnement des personnels.

Accueil des stagiaires, doctorants, post-doctorants, ingénieurs contractuels. Entre 2017 et 2022, *16 thèses ont été soutenues* et il y a actuellement *14 thèse en cours*.

L'équipe encourage et finance ses jeunes chercheurs dès le Master pour assister à des conférences (même s'ils n'y présentent pas d'article), des écoles et des journées : École des Jeunes Chercheurs en Programmation, École des Jeunes Chercheurs en Informatique Mathématique, Journées Francophones des Langages Applicatifs, Synchron, GT LVP et CLAP du GdR GPL. Ils participent au séminaire d'équipe et à l'animation scientifique locale qui leur est dédiée : séminaire des doctorants du laboratoire, journées de l'école doctorale. Ils sont encouragés à présenter leur sujet de Master ou de thèse et leurs premiers résultats à l'extérieur du laboratoire dès que possible (JFLA, Synchron) pour avoir une première expérience et un retour avant la soumission d'articles à des conférences internationales.

Les stagiaires, doctorants, post-doctorants et ingénieurs sont associés aux publications de l'équipe. Quand l'ordre alphabétique n'est pas choisi, le stagiaire ou le doctorant qui a rédigé en majorité l'article et l'a présenté en conférence apparaît en premier auteur. Les ingénieurs ayant participé à l'effort d'implantation et d'expérimentation sont coauteurs. L'équipe encourage la prise de responsabilité scientifique : les post-doctorants et ATER participent régulièrement à l'encadrement de stages de Master, et sont proposés pour les comités d'évaluation d'artefacts aux conférences.

Les doctorants participent aux instances de gouvernance du laboratoire : ils sont régulièrement membres du conseil des doctorants et du conseil de laboratoire (Hector Suzanne, Raphaël Monat).

Les doctorants et ingénieurs présents même à temps partiel dans l'équipe (e.g., thèses CIFRE) disposent, comme les membres à temps plein, d'un espace de travail et d'un ordinateur.

Devenir des doctorants. L'équipe accompagne les doctorants dans leur trajet professionnel : candidature aux postes d'enseignant-chercheur, de chercheur, recherche d'ATER et de post-doctorat. Nous aidons à la rédaction et relisons les dossiers, organisons des répétitions de soutenance, fournissons des recommandations. Les pistes de post-doctorat sont évoquées tôt durant la thèse et peuvent être préparées par une prise de contact et des visites organisées durant la thèse.

La grande majorité de nos docteurs, post-doctorants et ingénieurs durant la période évaluée a trouvé un poste permanent dans la recherche, l'enseignement ou l'industrie : 1 chercheur à INRIA, 1 MdC à Caen et 1 à l'EPITA, 1 Assistant Professor à Kyushu University, 1 Lecturer à l'Université de Plymouth, 9 ingénieurs de recherche dans des entreprises (Mathworks, Meta, BeeDeez, Scalian, Nomadic Labs, BeSport, Huawei UK), 1 enseignant en classes préparatoires ; 2 docteurs récents sont en post-doctorat.

Accueil des membres permanents. L'équipe soutient ses membres dans leur évolution de carrière. APR a ainsi bénéficié de *deux promotions internes* : un en DR CNRS et l'autre en PR. Elle continue d'accueillir ses anciens permanents (deux émérites, trois collaborateurs bénévoles) et ses membres ayant une délégation (une personne). Elle leur fournit un espace de travail et un accès aux ressources de l'équipe. Tous les membres ont accès au budget de l'équipe pour les missions et le matériel.

Accueil de chercheurs étrangers. L'équipe accueille régulièrement des professeurs invités étrangers pour entamer ou poursuivre des collaborations. Jan Midtgaard (Danemark) était invité en 2019 pour 2 semaines et Jiangchao Liu (Chine) pour 3 mois dans le cadre de l'ERC MOPSA qui a financé le séjour. Joshua Levine (USA) était invité en 2022, dans le cadre de l'ERC TORI. Alfredo Viola (Uruguay) était invité en 2018 et Simão Melo de Sousa (Portugal) en 2021 pour 1 mois chacun, avec un financement par le programme de *professeurs invités du LIP6* complété par le budget de l'équipe. Ces collaborations ont donné lieu à des articles communs (notamment [Tierny et al., 2017, Bodini et al., 2018]) et au projet PHC franco-portugais GASPAR sur les garanties d'exécution de programmes par analyse de ressources.

Convivialité. Les membres permanents et non-permanents, anciens et nouveaux, se retrouvent dans des moments de convivialité : déjeuners d'équipe, séminaires annuels hors les murs. Nous partageons une salle de convivialité avec l'équipe Complex Networks du LIP6.

Référence 3. L'unité est attractive par la reconnaissance de ses succès à des appels à projets compétitifs.

Projets internationaux et européens. L'équipe a reçu deux bourses ERC Consolidator : l'*ERC MOPSA* (2015–2022, 1,7M €) en vérification des logiciels, et l'*ERC TORI* (2019–2024, 2M €) en analyse topologique de données. Nous avons participé au projet *H2020-FET VESTEC Visual Exploration and Sampling Toolkit for Extreme Computing* (2018–2021, 10 partenaires, 3,2M € dont 255K € pour SU). Un collègue a participé au projet *ITEA 3*

ASSUME Affordable Safe & Secure Mobility Evolution (2015–2018, 38 partenaires, 21M € dont 176K € pour SU). Un membre de l'équipe a obtenu un projet *CNRS 80|PRIME*, promouvant l'interdisciplinarité et qui a permis de co-financer une thèse avec l'Université de l'Arizona (2022–2025).

Projets nationaux. L'équipe est régulièrement lauréate à des appels à projets ANR : participation dans des projets collaboratifs, comme le projet *ANR Coverif* (2015–2019) sur la combinaison entre interprétation abstraite et programmation par contraintes et le projet *ANR ELICA* (2015–2019) sur les méthodes logiques pour l'étude statique de la complexité, l'*ANR JCJC* S^3 sur la sémantique et les espèces de structure (2020–2024) et un projet *ANR Tremplin-ERC* (2019–2021).

L'équipe était porteuse de projets PEPS du CNRS : *PEPS APRES* (2016–2017) sur la programmation concurrente et *PEPS GraphGPU* (2016–2017) sur la programmation d'algorithmes de graphes sur GPU. Dans le cadre du PIA, nous participons au projet *FSN AVIDO* (2015–2018). L'équipe est maintenant impliquée dans le *PEPR Cybersécurité* avec le projet Secureval (2022–2028, 7 partenaires académiques, 7,65M € dont 710K € pour SU).

Projets régionaux. Nous portons un projet *Émergence de la ville de Paris* : *ReaLiSe* (2020–2025), sur la sémantique des langages réactifs et probabilistes, avec l'IRIF, l'ENS et le LIP6. L'équipe participe aux pôles de compétitivité à travers le projet *LCHIP du Fond Unique Interministériel* (2017–2020) sur la programmation sécurisée des plateformes à faible coût et le projet *UCF* (2014–2017) sur la programmation Web réactive. L'équipe était membre du projet *DIM Biatlon* (2018–2021).

Financements à Sorbonne Université. L'équipe répond aux appels internes à projets à SU. Il s'agit d'abord des bourses de thèse de l'EDITE (3 doctorants concernés). Au sein du LIP6, le financement des projets LIP6 a permis de financer un stage de Mater 2 en collaboration avec l'équipe ALSOC (projet PART en 2018).

Utilisation des budgets. Les budgets sont essentiellement consacrés au recrutement de doctorants et post-doctorants, d'ingénieurs contractuels et de stagiaires. Une petite partie est consacrée aux missions (conférences, accueil de visiteurs), à du petit matériel informatique ($<$3K €) et des serveurs de calcul modestes ($<$15K €). Par exemple, l'ERC MOPSA a financé 3 thèses et 2 post-doctorats ; l'ANR Coverif a financé une thèse en co-tutelle ; les projets FUI ont financé une thèse et un ingénieur de recherche ; le PEPR Secureval financera 2 thèses et un post-doctorant dans l'équipe. **Financements industriels**.

L'équipe encadre *7 doctorants directement financés par l'industrie*, dont 5 sur bourse CIFRE (avec Renaut, Total, Kitware, Thalès et Courtanet) et 2 doctorants ayant un poste permanent d'ingénieur (chez Airbus et Nomadic Labs). Outre le financement de thèse, ces contrats industriels sont associés à un versement par l'industriel de crédits, utilisés par l'équipe notamment pour des missions et du matériel. Le nombre de bourses de thèse financées par projets et par collaborations industrielles dépasse de loin celui des bourses de l'École doctorale (3 sur la période).

Référence 4. L'unité est attractive par la qualité de ses équipements et de ses compétences techniques.

Les compétences techniques exploitées en interne et visibles à l'extérieur concernent essentiellement le développement logiciel. Outre des prototypes de recherche pour l'expérimentation à des fins de publication (décrits au domaine 3), nous développons et maintenons des logiciels robustes, testés, documentés et diffusés sous licence libre.

Logiciels de recherche. Plusieurs logiciels sont développés pour le support de la recherche en interne et pour une diffusion à la communauté scientifique du domaine. La *bibliothèque TTK* pour la visualisation de données, initiée en 2014, est activement développée dans l'ERC TORI ; elle a une très bonne visibilité (contributions par 14 universités et 3 entreprises), elle est intégrée au logiciel ParaView de Kitware, et fait l'objet de tutoriels et démonstrations (portfolio 1). L'outil de vérification de logiciels *MOPSA* (portfolio 3), plus récent (2019), est diffusé avec sa documentation et ses benchmarks sur GitLab. Initié durant l'ERC MOPSA (2016–2022), ce logiciel est exploité au sein du projet PEPR Secureval qui démarre ; il est au cœur de collaborations industrielles en cours avec Airbus et Nomadic Labs et a été primé à la compétition SV-COMP 2023 (médaille de bronze en analyse de logiciels système). L'équipe a participé au développement du solveur de contraintes *AbSolute* durant le projet ANR Coverif. L'algorithme développé dans l'équipe pour la résolution de polynômes en nombres flous (portfolio 4) fait l'objet d'une implantation diffusée dans l'outil *SageMath*. *Arbogen* est un outil de génération aléatoire de grandes structures arborescentes, développé par l'équipe et utilisé par des chercheurs de la communauté Aléa en combinatoire (27 étoiles sur GitHub). L'environnement de programmation *OMicroB* (portfolio 5) est un laboratoire expérimental pour la programmation de haut niveau sur des microcontrôleurs à faibles ressources (115 étoiles et 25 forks sous GitHub). L'équipe développe également des logiciels qui, même s'ils tirent parti des résultats de recherche, visent l'enseignement. Ils sont décrits au domaine 4.

Développement et maintenance de logiciels libres. L'équipe maintient également des logiciels de recherche plus anciens. La *bibliothèque Apron* (2009–) reste une bibliothèque de référence pour l'interprétation abstraite numérique utilisée dans de nombreux analyseurs statiques (78 étoiles et 27 forks sous GitHub, 508 citations dans des articles scientifiques). La *bibliothèque SYM* est une partie intégrante du système de calcul formel MAXIMA pour la manipulation de fonctions symétriques. *SPOC* est un ensemble d'outils de programmation de cartes graphiques en OCaml (127 étoiles et 8 forks sous GitHub). *OBrowser* est un portage de la machine virtuelle OCaml en JavaScript. *Zamcov* est une machine virtuelle en OCaml pour la couverture structurelle.

L'équipe est très active dans la communauté du logiciel libre par le développement de bibliothèques et d'utilitaires variés : Zarith, une bibliothèque multi-précision pour OCaml (67 forks et 180 étoiles sous GitHub, très utilisée dans la communauté OCaml) ; cl-jupyter, un shell interactif pour Common Lisp (199 étoiles sous GitHub) ; TikZ-editor, un éditeur pour le package TikZ (165 étoiles sous GitHub).

Stratégie de développement. Le développement des outils de recherche est assuré par les chercheurs permanents et les doctorants. Certains projets ont un budget spécifique pour recruter des ingénieurs pendant 1 à 5 ans pour soutenir ces développements (ERC TORI, ERC MOPSA, FUI LCHIP, etc.). Des développements ponctuels simples font l'objet de stages étudiants (Master 1, stages IRILL). L'absence de soutien structurel par les tutelles est un *risque important* pour ces logiciels et les compétences techniques associées : notre capacité à les maintenir et les développer est assujettie à l'obtention de financements sur contrat, et au temps passé (souvent par les permanents) pour former les nouveaux recrutés et garantir la continuité.

Domaine 3. Production scientifique

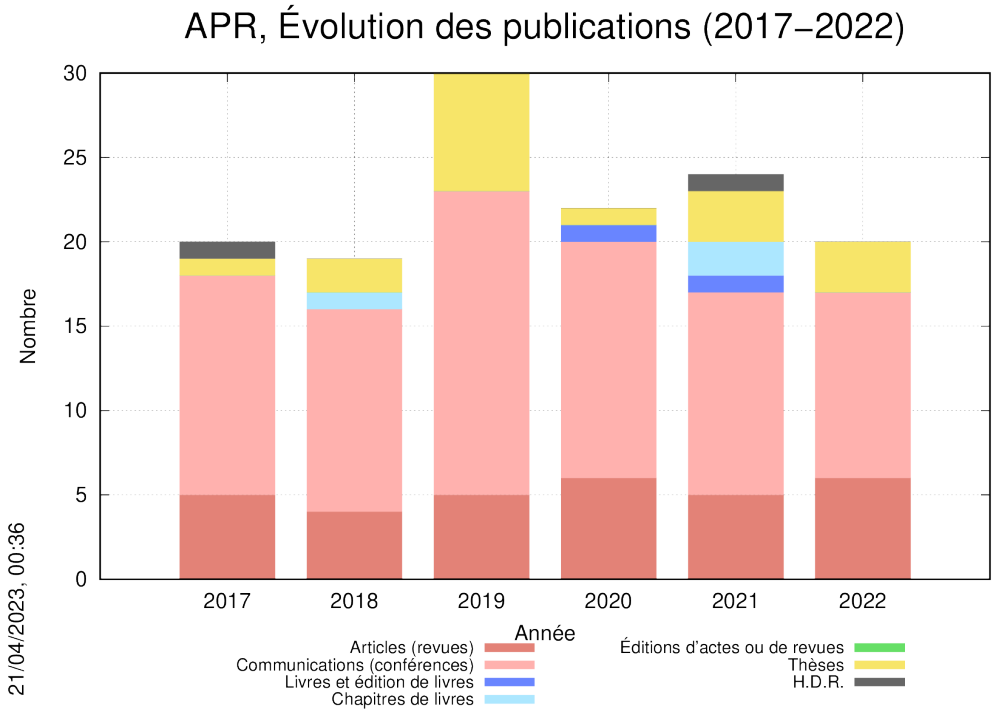


FIGURE 1 – Évolution des publications entre 2017 et 2022

	2017	2018	2019	2020	2021	2022
Articles (revues)	0.83	0.61	0.66	0.80	0.62	0.80
Communications (conférences)	2.16	1.84	2.40	1.86	1.50	1.46

TABLE 2 – Publications par ETPR par an entre 2017 et 2022

Référence 1. La production scientifique de l'unité satisfait à des critères de qualité.

Nos thèmes de recherche ont en commun de se baser sur des approches formelles. La validité des résultats théoriques est justifiée par des théorèmes, dont la preuve est publiée et vérifiable. Par exemple, nos résultats en analyse statique de programmes (portfolio 3, [Kabi et al., 2020, Bau et al., 2022, Monat et al., 2021, Delmas et al., 2021, Suzanne and Miné, 2018]) se basent sur le cadre formel de l'interprétation abstraite [5], appliqué à l'analyse de nouveaux langages et propriétés ; nos résultats en analyse de pire temps d'exécution [Varoumas and Crolard, 2019] suivent une approche basée sur la théorie du typage ; ceux sur la complexité de la programmation concurrente (portfolio 6, [Demangeon and Yoshida, 2023]) sont basés sur des algèbres de processus ; ceux en analyse combinatoire des systèmes concurrents (portfolio 2, [Genitrini et al., 2022]) prennent racine dans la combinatoire analytique de Flajolet, avec une application originale au phénomène de concurrence.

Outre les preuves théoriques, des résultats de l'équipe font l'objet d'une validation expérimentale reproductible. Par exemple, en analyse statique avec MOPSA, [Monat et al., 2021, Delmas et al., 2021] présentent des analyses de codes réels qui démontrent l'intérêt pratique des méthodes proposées.

Publications. L'équipe publie ses résultats dans des journaux et conférences internationales avec comité qui font référence dans leur domaine. Nous donnons ici quelques exemples en lien avec les éléments sélectionnés dans le portfolio. En topologie computationnelle (portfolio 1), [Tierny et al., 2017, Vidal et al., 2021, Lukasczyk et al., 2020, Guillou et al., 2023] sont publiés dans *IEEE TVCG*, la revue de référence en visualisation et analyse de données, avec des co-auteurs internationaux, tandis que [Gueunet et al., 2019a] sur les calculs des arbres de jointure est publié dans *IEEE TPDS*, revue internationale de référence en HPC. Les résultats en analyse combinatoire (portfolio 2) sont publiés dans la revue *Theoretical Computer Science*. En vérification de programmes (portfolio 3), les résultats sont publiés dans les actes de conférences spécialisées du domaine : *Static Analysis Symposium* [Monat et al., 2021, Delmas et al., 2021], *NASA Formal Methods*, *VMCAI* [Botbol et al., 2017], *ICFEM* [Sall et al., 2019], et des conférences plus généralistes en langages de programmation comme *European Symposium on Programming Languages*. Les résultats en complexité distribuée (portfolio 6) sont publiés dans la revue *Information and Computation* [Demangeon and Yoshida, 2023] dans le cadre d'une collaboration internationale avec Imperial College. En calcul formel, les résultats sont publiés dans *Fuzzy Sets and Systems* (portfolio 4). Ces publications de travaux mûrs sont complétées par des publications plus modestes (workshops, journées) de travaux en cours afin de leur donner une bonne visibilité et permettre un retour de la communauté (notamment pour les travaux des stagiaires et jeunes doctorants).

Production logicielle et prototypes de recherche. L'équipe développe de nombreux logiciels diffusés dans la communauté internationale, et qui sont donc une part importante de sa production scientifique. Ces logiciels sont développés selon des critères de qualité élevés, notamment par l'emploi de forges logicielles collaboratives (GitHub, GitLab), du développement de documentations, de tutoriaux, de présentations vidéos (portfolio 1), de tests avec intégration continue (portfolio 3), qui participent au succès de leur diffusion. Les articles sont généralement adossés à un développement logiciel. Outre les logiciels cités plus haut, nos doctorants développement des prototypes dans le cadre de leur recherche, par exemple : Pendulum (programmation Web synchrone), MPAI (analyse de programmes concurrents [Botbol et al., 2017]) et une bibliothèque Coq pour la programmation impérative par raffinement [Sall et al., 2019].

Référence 2. La production scientifique de l'unité est proportionnée à son potentiel de recherche et correctement répartie entre ses personnels.

La production scientifique est stable et nous semble bien proportionnée par rapport au potentiel de recherche de l'équipe, et ce malgré son fort investissement dans l'enseignement et la gestion de l'enseignement (cours de L1 à plus de 1100 étudiants, gestion d'un parcours du Master d'informatique, création d'une préparation à l'agrégation d'informatique) ainsi que la gestion de la recherche (participation au CoNRS, au CNU, aux GT de GdR). Elle couvre l'ensemble de ses thèmes de recherche : algorithmique, programmation et résolution. Sans s'arrêter au seul nombre d'articles publiés, il faut tenir compte, dans ces productions, des réalisations logicielles de qualité, qui sont un aspect important de l'équipe et contribuent à son rayonnement. Des chercheurs de l'équipe ont également publié ou participé à la publication de livres de cours (3 personnes).

La production est répartie entre les membres de l'équipe, avec des variations expliquées par la démographie (départs en retraite), les décharges et délégations partielles, ou encore l'arrivée récente des chercheurs. Les chercheurs en retraite depuis peu ont néanmoins une participation active à la vie scientifique de l'équipe, par exemple en assistant aux séminaires et journées scientifiques.Un de nos membres, par ailleurs directeur de l'ESIEE, était présent à 20% au LIP6, ce qui lui a permis de co-encadrer une thèse.L'arrivée d'une nouvelle professeure dans l'équipe en octobre 2020 correspond à un changement thématique (programmation probabiliste et réactive) et la

mise en place des projets associés, expliquant que ses travaux réalisés dans l'équipe commencent juste à être publiés. Un collègue a bénéficié d'une décharge de service pour activité syndicale : il était présent au LIP6 pour ses activités scientifique à 50 % jusqu'à 2019, puis à 20% depuis 2019. Néanmoins, il a repris une activité de recherche sur l'algèbre des nombres flous en collaboration avec une autre collègue, ce qui a produit plusieurs publications. Ce travail a fait suite au co-encadrement en 2016 d'un stage de Master 2 (qui a ensuite réalisé une thèse dans l'équipe ALMASTY). Il témoigne de l'efficacité de la *commission d'aide à la publication du LIP6* pour favoriser des reprises d'activité scientifique, puisque le stage en avait reçu le soutien financier.

Les doctorants participent pleinement à la production d'articles (ils sont souvent premier auteur, sauf choix d'un ordre alphabétique, et présentent leurs travaux en conférence) et les ingénieurs sont coauteurs des publications sur les logiciels qu'ils participent à développer.

Référence 3. La production scientifique de l'unité respecte les principes de l'intégrité scientifique, de l'éthique et de la science ouverte. Elle est conforme aux directives applicables dans ce domaine.

Pratique de la science ouverte. L'équipe APR s'efforce de mettre en pratique les principes d'intégrité scientifique en publiant de manière ouverte et transparente tous ses résultats de recherche. Les articles publiés sont déposés sur HAL en *texte intégral* ("version auteur"). Nous nous efforçons de mettre également en ligne des versions étendues des articles ou des pré-publications incluant les annexes non-publiés, car ces dernières fournissent des informations importantes pour vérifier la validité des résultats (preuves de théorèmes, etc.).

En matière de logiciel, nous plaçons nos réalisations originales sous une *licence libre* (GPL, LGPL, MIT, CeCILL), souvent dans des forges logicielles comme GitHub et GitLab, et nous favorisons l'intégration de nos travaux dans des plateformes existantes en logiciel libre (e.g., SageMath). Nous fournissons également en libre accès les bases de données d'exemples et les benchmarks servant à réaliser les expériences, ce qui permet de capitaliser les expérimentations clés de nos articles, par exemple pour TTK (portfolio 1) et MOPSA (portfolio 3). Plusieurs de nos publications en vérification de logiciels ([Monat et al., 2021, Delmas et al., 2021], etc.) ont reçu les *badges Validated, Extensible et Available* du comité d'évaluation des artefacts, et 5 publications en analyse topologique de données sont certifiées par le *Replicability Stamp Initiative* (initiative indépendante certifiant la reproductibilité des articles).

Promotion de la science ouverte. L'équipe est également fortement impliquée dans des instances académiques qui font la promotion de la science ouverte. Une collègue était membre (2018–2020) du *groupe de travail Libre Accès de Sorbonne Université* qui a défini un plan d'action de notre université sur les 4 années à venir et a élaboré la Charte pour le libre accès aux publications.[2] Un collègue est *directeur de l'IRILL* (Initiative de Recherche et d'Innovation sur le Logiciel Libre, initiative regroupant SU, INRIA, l'Université Paris Cité), hébergée à SU. Les activités liées à ces fonctions sont détaillées au domaine 4.

Domaine 4. Inscription des activités de recherche dans la société

Référence 1. L'unité se distingue par la qualité et la quantité de ses interactions avec le monde non-académique.

Projets avec industriels.

Plusieurs projets financés dans le cadre d'appels à projet compétitifs ont des partenaires industriels. Le projet *FSN AVIDO* (2015–2018), avec *EDF*, *Kitware*, *Total* et INRIA, portait sur l'analyse et la visualisation de données à grande échelle. Le projet *FUI LCHIP* (2017–2020), avec *Clearsy*, *OCamlPro* et la *SNCF*, portait sur la définition d'une plate-forme matérielle/logicielle pour développer des automatismes sécuritaires à bas coût. Le projet *FUI UCF* (2014–2017), avec *AlterWay*, *Xwiki* et *OCamlPro*, portait sur la gestion de contenus ubiquitaires pour le Web.

Thèses avec l'industrie. L'équipe entretient des collaborations avec des entreprises notamment par le co-encadrement de thèses CIFRE. Une collègue co-encadre avec *Renault Software Lab* une thèse sur la conception d'un DSL adapté à l'automobile pour l'architecture AUTOSAR et la certification de sécurité ISO 26262. Deux collègues co-encadrent avec *Thalès* une thèse (2016–2019) sur les codes correcteurs d'erreurs et nous travaillons depuis 2019 avec Courtanet dans le cadre d'une thèse sur la fouille de motifs dans les graphes temporels. Un membre d'APR a dirigé avec *Kitware* une thèse (2016–2019) sur le calcul haute performance pour l'analyse topologique de données et avec *Total* une thèse (2016–2019) sur la réduction et la comparaison de structures d'intérêt dans des jeux de données massifs par analyse topologique. Par ailleurs, deux thèses ont été financées sous forme

2. `https://www.sorbonne-universite.fr/bu/engagements-pour-la-science-ouverte`

de mise à disposition (partielle ou complète) pour la réalisation de la thèse d'un ingénieur en poste permanent dans une entreprise. Ainsi, David Delmas, ingénieur en CDI chez *Airbus* depuis plus de 20 ans, a réalisé une thèse sur la vérification de propriétés de portabilité (2018–2022 à 75%) et Guillaume Bau, ingénieur en CDI chez *Nomadic Labs*, réalise depuis 2020 une thèse sur la vérification de *smart contracts*. Ces collaborations sont encadrées par des conventions réalisées par la *Direction de la Recherche et la Valorisation* de SU, qui apporte par ailleurs une aide précieuse pour les mettre en place.

Ces collaborations ont un effet positif sur la recherche. Par exemple, David Delmas a pu valider ses résultats de thèse par des expériences sur du code propriétaire d'Airbus de grande taille [Delmas et al., 2021], ce qui serait difficile dans un cadre purement académique. Autre témoin de nos collaborations industrielles : de nombreuses publications sont co-signées par nos collaborateurs industriels, et nous faisons souvent appel à des personnalités du monde de l'industrie dans les jurys de thèse. De manière générale, les collaborations avec les entreprises s'inscrivent dans la durée, dont la réalisation d'une thèse CIFRE ou d'un projet n'est qu'un aspect ponctuel. Un témoin du succès des collaborations est le recrutement dans les entreprises des doctorants après leur thèse ainsi que l'adoption des technologies développées durant la thèse, par exemple l'entreprise Kitware avec la thèse de Charles Gueunet et le logiciel TTK. Un autre exemple est le recrutement chez Nomadic Labs d'un jeune docteur de l'équipe, Vincent Botbol, qui a permis par la suite le financement par l'entreprise de la thèse de Guillaume Bau, que Vincent co-encadre.

Formation et industrie. L'équipe interagit également avec l'industrie à travers la formation par la recherche au sein du Master STL (décrit plus en détail à la référence 2), qui propose à ses étudiants des *stages de fin d'étude* en recherche et développement dans l'industrie, ainsi qu'une filière en alternance par la voie de l'*apprentissage en entreprise*.

Interactions avec le monde de la santé. Les recherches en calcul formel de l'équipe ont une dimension pluri-disciplinaire, avec notamment des applications en santé. Il s'agit d'un côté d'applications aux plans d'expériences pour les études cliniques en médecine dans un travail réalisé avec un collègue du *LPSM* [Koné and Valibouze, 2021], et d'autre part d'applications à la modélisation en biologie dans un travail réalisé en collaboration avec l'*Institut Pasteur de Tunis* [Abdeljaoued-Tej et al., 2019].

Écosystème du logiciel libre. L'un de nous est depuis 2018 *directeur de l'IRILL* : l'Initiative de Recherche et d'Innovation sur les Logiciels Libres qui regroupe SU, INRIA, l'Université Paris Cité. L'IRILL, hébergée à SU, est un lieu d'échange entre chercheurs en informatique et communautés de développeurs de logiciels libres. L'IRILL s'intéresse aussi bien aux aspects scientifiques, qu'éducatifs et économiques des logiciels libres. Le séminaire régulier de l'IRILL invite des personnalités du monde académique et industriel. L'IRILL organise depuis 2022 des stages d'été de développement avec des partenaires académiques (SU, Université Paris Cité, EPITA) et des entreprises (Mozilla, Nomadic Labs).

Un collègue est membre du *comité de pilotage du HUB Open Source du pôle Systematic* depuis sa création en 2019 et, avant cela, du *Groupe de Travail Logiciel Libre* qui le précédait. Il a co-organisé les journées académiques–industriels "Language et Outils pour la Fiabilité Logicielle" en 2016 et 2017 et "IoT et sécurité" en 2018 et 2019 dans le cadre de l'*Open Source Innovation Spring*.

Référence 2. L'unité développe des produits à destination du monde culturel, économique et social.

Industrialisation des logiciels issus de la recherche.

Certains logiciels issus de la recherche, décrits au domaine 2, référence 4, ont vocation à être utilisables dans l'industrie. Notamment, le *logiciel TTK* (portfolio 1, [Tierny et al., 2017]) est intégré depuis 2022 dans le *logiciel ParaView*, une plateforme de visualisation en logiciel libre développée par l'entreprise Kitware. Nous avons par ailleurs pour projet le transfert du logiciel libre de vérification de programmes *MOPSA* (portfolio 3, [Journault et al., 2019a]) pour une utilisation industrielle chez Airbus, suite aux résultats obtenus dans la thèse récemment soutenue de David Delmas [Delmas et al., 2021].

Logiciels pour l'enseignement.

L'équipe développe des logiciels pour l'enseignement et la vulgarisation. *LaTTe* est un assistant de preuve en logiciel libre basé sur la théorie des types (9 forks, 235 étoiles sur GitHub, licence MIT). Il est utilisé par une petite communauté pour formaliser des théorèmes mathématiques dans un cadre de vulgarisation. Il est accompagné d'exemples, de vidéos de présentation et d'une intégration dans l'environnement NextJournal pour la recherche reproductible. *MrPython* est un environnement développé par Frédéric Peschanski pour l'enseignement de la programmation Python en L1 à SU ($\simeq$1100 étudiants par an), avec la particularité d'exploiter nos compétences techniques en analyse statique pour proposer un environnement plus pédagogique. Il accompagne un livre de cours de

programmation (2 collègues d'APR). La bibliothèque *EasyCheck*, présentée aux JFLA 2020, facilite la conception d'exercices à correction automatique pour OCaml dans l'environnement `learnocaml` en produisant des rapports de correction plus précis et en proposant de nouveaux schémas de correction pour les traits avancés du langage. Un de nos membre a développé une application pour gérer en temps réel les vœux des étudiants en Master d'informatique et construire des emplois du temps adéquats sous contraintes d'horaires et d'effectifs.

Formations universitaires.

L'équipe est très impliquée dans les activités de formation, notamment à SU, et ses membres ont de fortes responsabilités dans ces formations. En enseignement, ils interviennent à tous les niveaux de l'UFR d'ingénierie de SU, en Licence "sciences formelles" (L1) et d'informatique (L2–L3) et dans le parcours STL du Master d'Informatique (M1–M2, y compris dans les filières apprentissage CFA INSTA), avec des cours dans les domaines fondamentaux de l'informatique : algorithmique, programmation, sémantique, vérification, compilation, etc. Une collègue enseigne également à l'UFR de mathématiques à l'ISUP, et deux autres au MPRI. Deux collègues participent depuis 2009 à l'organisation des *Journées Franciliennes de Programmation*, un concours de programmation pour les étudiants en Licence regroupant les universités Paris Cité, Paris-Saclay et Sorbonne Université.

L'équipe a des responsabilités dans la gestion de l'enseignement en Licence, notamment avec la charge du *cours de L1 d'introduction à la programmation* (≃1100 étudiants chaque année). Une collègue est co-responsable depuis 2012 du parcours MIPI du L1 et participe à l'examen des dossiers Parcoursup de 2018 à 2021. Un membre de l'équipe est responsable depuis 2009 du *programme international de la Licence d'informatique*, permettant à des étudiants en Licence d'effectuer un semestre à l'Université de Montréal (5 à 10 étudiants par an). En Master, deux collègues sont depuis 2019 *co-responsables du parcours "Science et Technologie du Logiciel" (STL) du Master d'informatique* (≃110 étudiants par an). Deux autres l'étaient de 2014 à 2019. Le parcours STL a pour objectif de former des spécialistes en développement logiciel, avec une expertise pratique et théorique en programmation et en algorithmique. Il a des débouchés professionnels dans les grandes entreprise, les bureaux d'études, les petites entreprises innovantes et, en recherche, des thèses dans des laboratoires publics ou privés. En 2013, le M2 STL a ouvert une *filière en alternance* par la voie de l'apprentissage avec le CFA INSTA, mêlant périodes de travail en entreprise et de cours à l'université et au CFA. Cette expérimentation a été un succès par la diversité des contrats d'apprentissage (grands groupes, PME et startups), les retour très positifs des tuteurs et des étudiants, et un taux d'insertion professionnel équivalent aux filières classiques avec des salaires légèrement supérieurs.

Deux personnes d'APR sont membres élus du *conseil du département de Master* depuis 2011. Par ailleurs, l'un des deux est aussi élu de la *commission des enseignements de l'UFR* depuis 2020. Nous comptons parmi nous depuis 2022 le *directeur adjoint du Master d'informatique* (12 parcours, ≃800 étudiants par an). Une collègue est directrice des études informatiques et adjointe de l'*ISUP* jusqu'à 2020, puis directrice des études. Un de nos membres est depuis 2017 *directeur de l'école d'ingénieur ESIEE*, membre fondatrice de l'Université Gustave Eiffel.

L'introduction en 2021 de l'agrégation d'informatique, de la spécialité NSI et des classes préparatoires MP2I et MPI reconnaît l'informatique comme discipline scientifique. C'est un membre d'APR qui a *créé un parcours de Master préparant à l'agrégation*. Seules trois formations préparant à ce concours existent en France et celle-ci est la seule préparant à l'option "Informatique Pratique", formant ainsi des enseignants en poste, aussi bien que les étudiants désirant de se lancer dans cette voie. Cela permet d'introduire les enseignants en poste aux thématiques avancées de la recherche en informatique, mais aussi de former les futurs enseignants à l'apprentissage de la programmation comme discipline et pas seulement comme vecteur pour utiliser l'outil informatique pour résoudre des problèmes. L'agrégation d'informatique comporte une large part d'apprentissage de la programmation, et de méthodes et outils pour la correction des programmes (typage, tests, documentation, etc.). L'expertise de l'équipe en méthodes formelles a permis de concevoir un cursus avec une exigence particulière dans la conception de programmes sûrs et maintenables.

Ces forts investissements en enseignement ont des interactions positives avec l'activité de recherche. D'une part, la formation par la recherche nous permet de donner aux étudiants une vision précise des thématiques et résultats en recherche récente et, pour certains, l'idée de réaliser un doctorat. D'autre part, les recherches de l'équipe ayant vocation à améliorer la qualité des logiciels grâce à des méthodes bien fondées formellement, l'enseignement de cours de base dès le L1 permet de sensibiliser les étudiants scientifiques aux problématiques et aux solutions existantes (e.g., le typage statique) liées à la qualité de conception et de réalisation des programmes informatiques.

Colloquium. Le Colloquium d'informatique de Sorbonne Université accueille des orateurs internationaux prestigieux avec un spectre thématique large pour donner des exposés à un public de chercheurs et d'étudiant en informatique, mais aussi animer des master-class pour doctorants. Il contribue au rayonnement de SU en France. Deux de nos membres participent au *comité d'organisation du Colloquium* (organisation directe de 4 séances depuis 2017, participation à l'organisation d'autres séances, animation et gestion du comité par le responsable de l'équipe).

Référence 3. L'unité partage ses connaissances avec le grand public et intervient dans des débats de société.

Manifestations grand public. Nous participons à la *fête de la science*. Elle a donné un colloque sur la programmation probabiliste au *Collège de France* dans le cadre du cours de Xavier Leroy. Un membre d'APR a donné un exposé de vulgarisation invité à *Timeworld* 2022 sur la démonstration du théorème des 4 couleurs. Une collègue est co-rédactrice en chef de l'édition "Au cœur de la recherche" du *Club de Mediapart*. Deux membres de l'équipe ont co-écrit avec Michel Mauny un article de vulgarisation dans la *revue Techniques de l'ingénieur* sur le thème "Typage des langages de programmation".

Parité. Une de nos collègues est *membre élue au CSI de l'INS2I* et a organisé un séminaire thématique intitulé "Place des femmes dans l'institut". Elle a participé à une table ronde aux conférences des 75 du LIP6 sur le thème : "Femmes dans le numérique : disparition réversible ?". Nous comptons parmi nous le *référent parité INS2I pour le LIP6* qui intervient régulièrement dans des établissements du secondaire sur le sujet "Femmes et science".

Science et société. Une collègue a organisé un séminaire thématique au CSI de l'INS2I : "Numérique, enseignement et société". Une collègue a été membre du *groupe de travail Libre Accès à SU* et a participé à des manifestations liées à cette activité : la Journée Académie des Sciences (2/04/2019), la Journée Recherche SIF–SMAI (2/10/2018), les Journées Nationales de la Science Ouverte (4–6/12/2018). Un collègue est fortement impliqué dans les débats internes à l'enseignement supérieur et la recherche, aux missions et conditions de travail de ses personnels, comme à ses enjeux pour la société en tant qu'*élu au CSA de l'université* (jusqu'en 2018), au *CSA ministériel* (depuis 2016) et *secrétaire général adjoint du SNESUP-FSU* (depuis 2019).

4 RÉFÉRENCES BIBLIOGRAPHIQUES EXTERNES

[1] J. Bertrane, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, and X. Rival. Static analysis and verification of aerospace software by abstract interpretation. In *AIAA Infotech@Aerospace*, number 2010-3385, pages 1–38. AIAA, Apr. 2010.

[2] Mathias Bourgoin, Emmanuel Chailloux, and Jean-Luc Lamotte. Efficient Abstractions for GPGPU Programming. *International Journal of Parallel Programming*, 42(4) :583–600, August 2014.

[3] B.-M. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2) :267–285, 2003.

[4] C. Calcagno, D. Distefano, J. Dubreil, D. Gabi, P. Hooimeijer, M. Luca, P. O'Hearn, I. Papakonstantinou, J. Purbrick, and D. Rodriguez. Moving fast with software verification. In *NFM*, pages 3–11. Springer, 2015.

[5] P. Cousot and R. Cousot. Abstract interpretation : A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. of POPL'77*, pages 238–252. ACM, Jan. 1977.

[6] Rémy El Sibaïe and Emmanuel Chailloux. Synchronous Web Programming,. In *International Workshop on Reactive and Event-Based Languages and Systems (REBLS)*, Amsterdam, Netherlands, October 2016.

[7] G.B. Mertzios, H. Molter, R. Niedermeier, V. Zamaraev, and P. Zschoche. Computing Maximum Matchings in Temporal Graphs. In *37th International Symposium on Theoretical Aspects of Computer Science*, volume 154 of *LIPIcs*, pages 27 :1–27 :14, 2020.

[8] A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation (HOSC)*, 19(1) :31–100, 2006.

[9] Benoît Vaugon, Philippe Wang, and Emmanuel Chailloux. Programming Microcontrollers in Ocaml : the OCaPIC Project. In *International Symposium on Practical Aspects of Declarative Languages (PADL 2015)*, volume 9131 of *Lecture Notes in Computer Science*, pages 132–148, Portland, OR, United States, June 2015. Springer Verlag.

5 RÉFÉRENCES BIBLIOGRAPHIQUES SIGNIFICATIVES DE APR

[Abdeljaoued-Tej et al., 2019] Abdeljaoued-Tej, I., Benkahla, A., Haddad, G., and Valibouze, A. (2019). Separators for Polynomial Dynamic Systems with Linear Complexity. In Bortolussi, L. and Sanguinetti, G., editors, *Computational Methods in Systems Biology 17th International Conference, CMSB 2019*, volume 11773 of *Lecture Notes in Computer Science*, pages 373–378, Trieste, Italy. Springer.

[Aubry et al., 2020] Aubry, P., Marrez, J., and Valibouze, A. (2020). Computing real solutions of fuzzy polynomial systems. *Fuzzy Sets and Systems*, 399 :55 – 76.

[Bau et al., 2022] Bau, G., Miné, A., Botbol, V., and Bouaziz, M. (2022). Abstract interpretation of Michelson smart-contracts. In *Proc. of the 11th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis (SOAP'22)*, pages 36–43. ACM.

[Baudart et al., 2023] Baudart, G., Bussone, G., Mandel, L., and Tasson, C. (2023). Filtrer sans s'appauvrir : inférer les paramètres constants des modèles réactifs probabilistes. In Bourke, T. and Demange, D., editors, *JFLA 2023 - 34èmes Journées Francophones des Langages Applicatifs*, pages 24–42, Praz-sur-Arly, France.

[Bodini et al., 2017b] Bodini, O., Dien, M., Genitrini, A., and Peschanski, F. (2017b). The Ordered and Colored Products in Analytic Combinatorics : Application to the Quantitative Study of Synchronizations in Concurrent Processes. In *14th SIAM Meeting on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 16–30.

[Bodini et al., 2018] Bodini, O., Dien, M., Genitrini, A., and Viola, A. (2018). Beyond series-parallel concurrent systems : The case of arch processes. In *29th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms, AofA 2018, June 25-29, 2018, Uppsala, Sweden*, pages 14 :1–14 :14.

[Bodini et al., 2022a] Bodini, O., Genitrini, A., Gittenberger, B., Larcher, I., and Naima, M. (2022a). Compaction for two models of logarithmic-depth trees : Analysis and experiments. *Random Structures & Algorithms*, 61(1) :31–61.

[Bodini et al., 2020a] Bodini, O., Genitrini, A., Gittenberger, B., and Wagner, S. (2020a). On the number of increasing trees with label repetitions. *Discrete Mathematics*, 343(8) :111722.

[Botbol et al., 2017] Botbol, V., Chailloux, E., and Le Gall, T. (2017). Static Analysis of Communicating Processes Using Symbolic Transducers. In Bouajjani, A. and Monniaux, D., editors, *International Conference on Verification, Model Checking, and Abstract Interpretation - VMCAI 2017*, volume 10145 of *Lecture Notes in Computer Science*, Paris, France. Springer International Publishing.

[Bridel-Bertomeu et al., 2019] Bridel-Bertomeu, T., Fovet, B., Tierny, J., and Vivodtzev, F. (2019). Topological Analysis of High Velocity Turbulent Flow. In *IEEE Symposium on Large Data Analysis and Visualization (posters)*.

[Bui-Xuan et al., 2022] Bui-Xuan, B., Hourcade, H., and Miachon, C. (2022). Computing small temporal modules in time logarithmic in history length. *Social Network Analysis and Mining*, 12(1) :19.

[Bui-Xuan et al., 2021] Bui-Xuan, H., Bui-Xuan, B., Le, V., and Mai, H. (2021). Low diameter algebraic graphs. In *European Conference on Combinatorics, Graph Theory and Applications*, volume 14 of *Trends in Mathematics and Research Perspectives CRM Barcelona*.

[Delmas et al., 2021] Delmas, D., Ouadjaout, A., and Miné, A. (2021). Static analysis of endian portability by abstract interpretation. In *Proc. of the 28th International Static Analysis Symposium (SAS'21)*, volume 12913 of *LNCS*, pages 102–123. Springer.

[Demangeon and Yoshida, 2023] Demangeon, R. and Yoshida, N. (2023). Causal computational complexity of distributed processes. In *Inf. Comput.*, page 104998.

[Dien et al., 2022] Dien, M., Genitrini, A., and Peschanski, F. (2022). A combinatorial study of async/await processes. In *Theoretical Aspects of Computing - ICTAC 2022 - 19th International Colloquium, Tbilisi, Georgia, September 27-29, 2022, Proceedings*, volume 13572 of *Lecture Notes in Computer Science*, pages 170–187. Springer.

[Genitrini et al., 2020a] Genitrini, A., Gittenberger, B., Kauers, M., and Wallner, M. (2020a). Asymptotic enumeration of compacted binary trees of bounded right height. *Journal of Combinatorial Theory, Series A*, 172 :105177.

[Genitrini et al., 2022] Genitrini, A., Pépin, M., and Peschanski, F. (2022). A quantitative study of fork-join processes with non-deterministic choice : application to the statistical exploration of the state-space. *Theor. Comput. Sci.*, 912 :1–36.

[Gueunet et al., 2019a] Gueunet, C., Fortin, P., Jomier, J., and Tierny, J. (2019a). Task-based Augmented Contour Trees with Fibonacci Heaps. *IEEE Transactions on Parallel and Distributed Systems*. Accepted.

[Gueunet et al., 2019b] Gueunet, C., Fortin, P., Jomier, J., and Tierny, J. (2019b). Task-based Augmented Reeb Graphs with Dynamic ST-Trees. In *Eurographics Symposium on Parallel Graphics and Visualization*.

[Guillou et al., 2023] Guillou, P., Vidal, J., and Tierny, J. (2023). Discrete Morse Sandwich : Fast Computation of Persistence Diagrams for Scalar Data – An Algorithm and A Benchmark. *IEEE Transactions on Visualization and Computer Graphics*.

[Journault et al., 2019a] Journault, M., Miné, A., Monat, R., and Ouadjaout, A. (2019a). Combinations of reusable abstract domains for a multilingual static analyzer. In *Proc. of the 11th Working Conference on Verified Software : Theories, Tools, and Experiments (VSTTE19)*, volume 12031 of *LNCS*, pages 1–18. Springer.

[Kabi et al., 2020] Kabi, B., Goubault, É., Miné, A., and Putot, S. (2020). Combining zonotope abstraction and constraint programming for synthesizing inductive invariants. In *Proc. of the 13th International Workshop on Numerical Software Verification (NSV'20)*, volume 12549 of *LNCS*, pages 221–238. Springer.

[Kästner et al., 2017] Kästner, D., Miné, A., Schmidt, A., Hille, H., Mauborgne, L., Wilhelm, S., Rival, X., Feret, J., Cousot, P., and Ferdinand, C. (2017). Finding all potential run-time errors and data races in automotive software. In *Proc. of Automotive Software, SAE world Congress (SAE'17)*, SAE Technical Paper, page 9. SAE International.

[Koné and Valibouze, 2021] Koné, M. and Valibouze, A. (2021). Nearest neighbor balanced block designs for autoregressive errors. *Metrika*, 84(3) :281–312.

[Lukasczyk et al., 2017] Lukasczyk, J., Aldrich, G., Steptoe, M., Favelier, G., Gueunet, C., Tierny, J., Maciejewski, R., Hamann, B., and Leitte, H. (2017). Viscous fingering : A topological visual analytic approach. *Applied Mechanics and Materials*.

[Lukasczyk et al., 2020] Lukasczyk, J., Garth, C., Maciejewski, R., and Tierny, J. (2020). Localized Topological Simplification of Scalar Data. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*.

[Miné, 2017] Miné, A. (2017). Tutorial on static inference of numeric invariants by abstract interpretation. *Foundations and Trends in Programming Languages (FnTPL)*, 4(3–4) :120–372.

[Monat et al., 2020] Monat, R., Ouadjaout, A., and Miné, A. (2020). Value and allocation sensitivity in static Python analyses. In *Proc. of the 9th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis (SOAP'20)*, pages 8–13. ACM.

[Monat et al., 2021] Monat, R., Ouadjaout, A., and Miné, A. (2021). A multilanguage static analysis of Python programs with native C extensions. In *Proc. of the 28th International Static Analysis Symposium (SAS'21)*, volume 12913 of *LNCS*, pages 323–345. Springer.

[Nauleau et al., 2022] Nauleau, F., Vivodtzev, F., Bridel-Bertomeu, T., Beaugendre, H., and Tierny, J. (2022). Topological Analysis of Ensembles of Hydrodynamic Turbulent Flows – An Experimental Study. In *IEEE Symposium on Large Data Analysis and Visualization*.

[Olejniczak et al., 2019] Olejniczak, M., Gomes, A. S. P., and Tierny, J. (2019). A Topological Data Analysis Perspective on Non-Covalent Interactions in Relativistic Calculations. *International Journal of Quantum Chemistry*.

[Olejniczak and Tierny, 2023] Olejniczak, M. and Tierny, J. (2023). Topological Data Analysis of Vortices in the Magnetically-Induced Current Density in LiH Molecule. *Physical Chemistry Chemical Physics*.

[Picavet et al., 2021] Picavet, T., Nguyen, N., and Bui-Xuan, B. (2021). Temporal matching on geometric graph data. In *12th International Conference on Algorithms and Complexity*, volume 12701 of *LNCS and LNTCS*, pages 394–408.

[Pont et al., 2021] Pont, M., Vidal, J., Delon, J., and Tierny, J. (2021). Wasserstein Distances, Geodesics and Barycenters of Merge Trees. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*.

[Pont et al., 2022] Pont, M., Vidal, J., and Tierny, J. (2022). Principal Geodesic Analysis of Merge Trees (and Persistence Diagrams). *IEEE Transactions on Visualization and Computer Graphics*.

[Sall et al., 2019] Sall, B. D., Peschanski, F., and Chailloux, E. (2019). A Mechanized Theory of Program Refinement. In *ICFEM 2019 - 21st International Conference on Formal Engineering Methods*, volume 11852 of *Lecture Notes in Computer Science*, pages 305–321, Shenzhen, China. Springer.

[Soler et al., 2019] Soler, M., Petitfrere, M., Darche, G., Plainchault, M., Conche, B., and Tierny, J. (2019). Ranking Viscous Finger Simulations to an Acquired Ground Truth with Topology-Aware Matchings. In *IEEE Symposium on Large Data Analysis and Visualization*.

[Soler et al., 2018a] Soler, M., Plainchault, M., Conche, B., and Tierny, J. (2018a). Lifted Wasserstein matcher for fast and robust topology tracking. In *IEEE Symposium on Large Data Analysis and Visualization*.

[Suzanne and Miné, 2018] Suzanne, T. and Miné, A. (2018). Relational thread-modular abstract interpretation under relaxed memory models. In *Proc. of the 16th Asian Symposium on Programming Languages and Systems (APLAS'18)*, volume 11275 of *LNCS*, pages 109–128. Springer.

[Sylvestre et al., 2022a] Sylvestre, L., Chailloux, E., and Sérot, J. (2022a). Accelerating OCaml programs on FPGA. In *15th International Symposium on High-level Parallel Programming and Applications (HLPP 2022)*, Porto, Portugal.

[Tierny et al., 2017] Tierny, J., Favelier, G., Levine, J. A., Gueunet, C., and Michaux, M. (2017). The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*. `https://topology-tool-kit.github.io/`.

[Varoumas and Crolard, 2019] Varoumas, S. and Crolard, T. (2019). WCET of OCaml Bytecode on Microcontrollers : An Automated Method and Its Formalisation. In Altmeyer, S., editor, *19th International Workshop on Worst-Case Execution Time Analysis (WCET 2019)*, volume 72 of *OpenAccess Series in Informatics (OASIcs)*, pages 5 :1–5 :12, Stuttgart, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Varoumas et al., 2020] Varoumas, S., Pesin, B., Vaugon, B., and Chailloux, E. (2020). Programming microcontrollers through high-level abstractions. In *VMIL 2020 - 12th ACM SIGPLAN International Workshop on Virtual Machine and Intermediate Languages*, pages 5–14, Chicago / Virtual, United States. Association for Computing Machinery.

[Varoumas et al., 2018] Varoumas, S., Vaugon, B., and Chailloux, E. (2018). A Generic Virtual Machine Approach for Programming Microcontrollers : the OMicroB Project. In *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, Toulouse, France.

[Vidal et al., 2019] Vidal, J., Budin, J., and Tierny, J. (2019). Progressive Wasserstein Barycenters of Persistence Diagrams. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*.

[Vidal et al., 2021] Vidal, J., Guillou, P., and Tierny, J. (2021). A Progressive Approach to Scalar Field Topology. *IEEE Transactions on Visualization and Computer Graphics*.

[Ziat et al., 2018] Ziat, G., Pelleau, M., Truchet, C., and Miné, A. (2018). Finding solutions by finding inconsistencies. In *Proc. of the International Conference on Principles and Practice of Constraint Programming (CP'18)*, volume 11008 of *LNCS*, pages 420–435. Springer.

A ANNEXE — MEMBRES PERMANENTS AU 31/12/2022

La table ci dessous liste les membres permanents de l'équipe APR.

NOM	Prénom	Corps	Employeur
AUBRY	Philippe	MCF	Sorbonne Université
BUI-XUAN	Binh-Minh	CR	CNRS
CHAILLOUX	Emmanuel	PR	Sorbonne Université
DEMANGEON	Romain	MCF	Sorbonne Université
GENITRINI	Antoine	MCF (HDR)	Sorbonne Université
MAIRESSE	Jean	DR	CNRS
MINÉ	Antoine	PR	Sorbonne Université
PESCHANSKI	Frédéric	MCF	Sorbonne Université
TASSON	Christine	PR	Sorbonne Université
TIERNY	Julien	DR	CNRS
VALIBOUZE	Annick	PR	Sorbonne Université

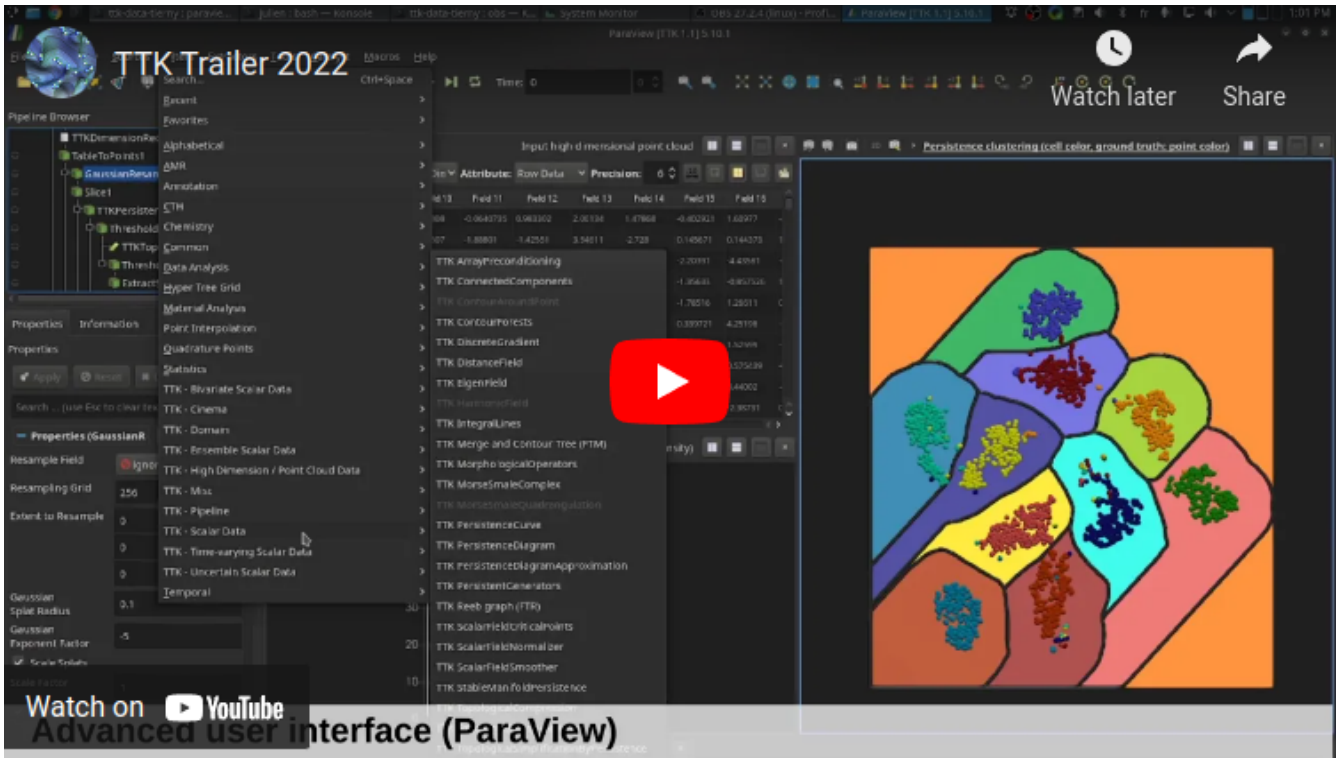
ÉLÉMENT DE PORTFOLIO 01

Vidéo

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : Vidéo de présentation de la bibliothèque logicielle *Topology ToolKit* (TTK)

URL de l'élément : https://youtu.be/8zg4seXlrss



2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

Cette vidéo fournit une vue d'ensemble concise des activités de l'équipe, dans la thématique « *Analyse Topologique de Données* », notamment dans le cadre du projet ERC TORI, https://erc-tori.github.io/.

3 PRÉSENTATION DE CET ÉLÉMENT

La vidéo fournit un bref panorama des fonctionnalités de la bibliothèque *Topology ToolKit* (TTK) [1, 6].

3.1 À propos de TTK

La bibliothèque TTK est écrite en C++ (environs 170,000 lignes). Elle a été initialement créée par Julien Tierny en 2014 et elle est ensuite devenue un réceptacle logiciel aux travaux de recherche de l'équipe en « *Analyse Topologique de Données* » et sert à ce titre de vitrine. TTK est distribuée en logiciel libre depuis 2017 sous une license permissive (BSD). Depuis, 17 institutions ont contribué à son code source (14 universités, 3 entreprises). Depuis 2021, TTK est officiellement intégrée au sein du logiciel libre *ParaView*, qui est un standard de-facto en visualisation et analyse de données.

3.2 À propos de la vidéo

La vidéo illustre, d'un point de vue utilisateur, l'usage de TTK aux travers d'exemples concrets (tirés de la base de données d'exemples de TTK [7]), provenant d'applications variées. En particulier, chaque séquence illustre un algorithme particulier, la plupart documentés dans des publications de l'équipe :

- Calcul rapide de diagrammes de persistance [3] :
  - Séquence $[0:23-0:27]$,
  - Séquence $[0:39-0:43]$,
  - Séquence $[1:01-1:10]$,
- Calcul rapide d'arbres de contour [2] :
  - Séquence $[0:50-0:54]$,
- Suivi de points critiques au cours du temps [5] :
  - Séquence $[1:11-1:16]$,
- Calcul rapide de barycentres de Wasserstein de diagrammes de persistance [8] :
  - Séquence $[1:23-1:27]$,
  - Séquence $[1:33-1:38]$,
  - Séquence $[1:44-1:48]$,
- Calcul rapide de barycentres de Wasserstein d'arbres de jointure [4] :
  - Séquence $[1:28-32]$,
  - Séquence $[1:39-1:43]$.

4 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Talha Bin Masood, Joseph Budin, Martin Falk, Guillaume Favelier, Christoph Garth, Charles Gueunet, Pierre Guillou, Lutz Hofmann, Petar Hristov, Adhitya Kamakshidasan, Christopher Kappe, Pavol Klacansky, Patrick Laurin, Joshua Levine, Jonas Lukasczyk, Daisuke Sakurai, Maxime Soler, Peter Steneteg, Julien Tierny, Will Usher, Jules Vidal, and Michal Wozniak. An Overview of the Topology ToolKit. In *TopoInVis*, 2019.

[2] Charles Gueunet, Pierre Fortin, Julien Jomier, and Julien Tierny. Task-Based Augmented Contour Trees with Fibonacci Heaps. *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[3] Pierre Guillou, Jules Vidal, and Julien Tierny. Discrete Morse Sandwich : Fast Computation of Persistence Diagrams for Scalar Data – An Algorithm and A Benchmark. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[4] Mathieu Pont, Jules Vidal, Julie Delon, and Julien Tierny. Wasserstein Distances, Geodesics and Barycenters of Merge Trees. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[5] Maxime Soler, Melanie Plainchault, Bruno Conche, and Julien Tierny. Lifted Wasserstein matcher for fast and robust topology tracking. In *IEEE LDAV*, 2018.

[6] Julien Tierny, Guillaume Favelier, Joshua A. Levine, Charles Gueunet, and Michael Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1) :832–842, 2017. `https://topology-tool-kit.github.io/`.

[7] TTK Contributors. The TTK Examples. `https://topology-tool-kit.github.io/examples/`, 2023.

[8] Jules Vidal, Joseph Budin, and Julien Tierny. Progressive Wasserstein Barycenters of Persistence Diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

ÉLÉMENT DE PORTFOLIO 02

Publication

1 DÉFINITION DE CET ÉLÉMENT

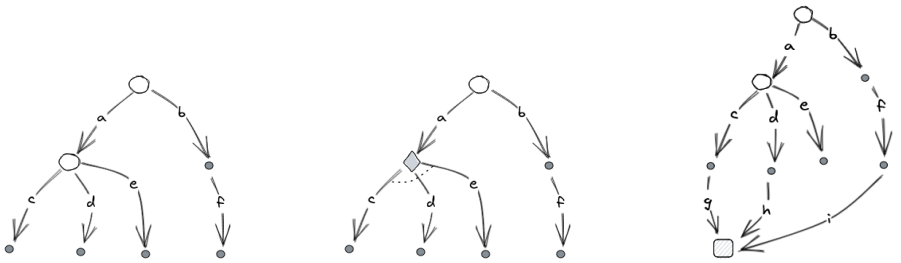
Titre de l'élément : *A quantitative study of fork-join processes with non-deterministic choice : application to the statistical exploration of the state-space.* Antoine Genitrini, Martin Pépin, Frédéric Peschanski. In Theoretical Computer Science, 2021, vol. 912.

URL de l'élément : `https://hal.science/hal-03201618/`

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

Nous avons choisi cet article car il représente un jalon important dans notre projet ParCo (Parallélisme & Combinatoire) dont l'objet concerne l'étude des systèmes concurrents sous l'angle de la combinatoire énumérative et/ou analytique. Pour la première fois, nous pouvons appliquer les outils quantitatifs et les algorithmes que nous avons développés pendant plusieurs années dans le cadre d'un langage concurrent assez expressif. L'article est d'ailleurs accompagné d'un artefact logiciel permettant de reproduire les expérimentations présentées. [1] Ce projet, en lui-même, est important pour l'équipe puisqu'il est un thème fédérateur entre des considérations liées à la programmation (concurrente) d'un côté, et aux études algorithmiques de l'autre. De plus, cet article est publié dans la revue TCS à fort impact, ce qui donne de la visibilité à notre projet. Cet article illustre également notre investissement dans la formation par la recherche, puisqu'il s'agit d'une des contributions importantes de la thèse de Martin Pépin, dirigée par Antoine Genitrini et co-encadrée par Frédéric Peschanski.

2.1 Contexte scientifique : le projet ParCo



(a) Parallélisme (b) Non-déterminisme (c) Synchronisation

FIGURE 1 – Les principaux phénomènes de concurrence.

Dans le projet ParCo, nous étudions les programmes parallèles vus comme des objets combinatoires, dans une démarche originale de rapprochement entre la sémantique des langages de programmation et la combinatoire. Nous nous basons sur des interprétations combinatoires des principaux phénomènes de concurrence : parallélisme, non-déterminisme et synchronisation (cf. Figure 1). D'un point de vue quantitatif, nous étudions en particulier le phénomène d'explosion combinatoire, au cœur de la problématique de concurrence. D'un point de vue pratique, nos travaux ouvrent de nouvelles voies dans le domaine de la génération aléatoire uniforme : génération de structures et génération de chemins d'exécutions. Sur ces fondements algorithmiques, nous développons des prototypes logiciels dans les domaines de la génération automatique de tests ainsi que de la vérification statistique de modèles. Ce projet a conduit à la publication de nombreux articles tant dans le domaine de la concurrence [2, 5, 6] que de la combinatoire et l'analyse d'algorithmes [1, 3, 4].

1. cf. `https://gitlab.com/ParComb/libnfj`

2.2 Présentation de l'article et de ses contributions

Le point de départ de l'article consiste en la présentation d'un modèle combinatoire permettant de combiner les interprétations que nous avons proposées dans nos précédents articles, en intégrant notamment :

- le parallélisme vu comme l'étiquetage croissant de structures arborescentes ;
- le non-déterminisme vu comme l'étiquetage croissant partiel de la structure ;
- la synchronisation de processus vue comme de l'étiquetage croissant sur-contraint.

En complément de cette interprétation combinatoire unifiée, l'article introduit également la possibilité de décrire des boucles dans les programmes, permettant les itérations. En terme d'expressivité, la proposition d'une interprétation combinatoire unique représente un grand pas en avant pour le projet ParCo, notamment pour ce qui concerne l'application en pratique de nos résultats.

Études quantitatives

Du point de vue analytique, la représentation de la sémantique de programmes concurrents sous forme de classes combinatoires nous permet d'étudier, en moyenne, des aspects quantitatifs concernant leur espace d'états. En guise d'exemple, l'article propose une étude quantitative précise du nombre moyen de chemins d'exécution induits par l'opérateur de choix non-déterministe. On peut noter, par exemple, que cet opérateur est bien moins « explosif » que l'opérateur de parallélisme, et qu'il peut donc être parfois intéressant de « déplier » les choix non-déterministes lorsque l'on désire effectuer des explorations statistiques de l'espace d'états. Un autre exemple d'étude quantitative proposée dans l'article concerne ce que l'on nomme « la forme typique » (*typical shape*) de l'espace d'états, étude basée sur l'estimation précise du nombre moyen de préfixes d'exécutions (plutôt que les exécutions complètes).

Algorithmique

D'un point de vue plus pratique, des algorithmes intéressants découlent naturellement de nos études quantitatives. Le premier algorithme présenté et expérimenté dans l'article concerne le comptage du nombre de préfixes d'exécution d'une longueur donnée. Bien que sans usage pratique immédiat, le comptage est la brique de base sur laquelle repose la plupart de nos autres algorithmes. Nous montrons dans [2] que si le modèle de synchronisation n'est pas contraint d'une façon ou d'une autre, le problème du comptage est difficile (techniquement, $\sharp P$-complet). Le nombre intrinsèquement exponentiel de choix non-déterministes « locaux » représente une seconde difficulté pour le comptage. Dans l'article, nous adoptons un modèle de synchronisation de type *fork-join* qui, d'une part, simplifie le problème du comptage, et d'autre part reste suffisamment intéressant du point de vue de l'expressivité. De plus, nous proposons un encodage de l'espace d'états sous la forme de fonctions génératrices, permettant de définir une notion de *choix global* ne nécessitant pas le dépliage des choix locaux.

Analyses de programme

Nous exploitons notre brique algorithmique de comptage pour développer deux analyses complémentaires pour les systèmes concurrents. Tout d'abord, nous développons un générateur aléatoire uniforme permettant de générer, pour un programme spécifique, des chemins d'exécution (ou ordonnancements) admis par ce programme. La contrainte d'uniformité permet de garantir la meilleure couverture possible en l'absence de connaissance préalable sur la forme de l'espace d'états. Cela représente donc une bonne stratégie par défaut pour l'exploration statistique de cet espace d'états. Nous proposons dans l'article une seconde approche complémentaire qui consiste à générer, de façon uniforme, des *préfixes* de chemins d'exécution plutôt que des chemins complets. Arrivés à un certain point de programme, par la voie uniforme, il est alors possible d'introduire un biais dans l'exploration en sélectionnant telle ou telle branche, avant de progresser – toujours de manière uniforme – dans l'espace d'états. Il est ainsi par exemple possible de choisir explicitement des branches « moins probables » durant l'exploration, et de fournir ainsi un moyen de contrôler finement cette exploration. Ces stratégies d'exploration contrôlable de l'espace d'états ouvrent des perspectives intéressantes pour le développement d'outils d'analyse de programmes concurrents dans le domaine du test automatisé et de la vérification statistique de modèle (*Monte-Carlo model checking*).

3 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Olivier Bodini, Matthieu Dien, Antoine Genitrini, and Frédéric Peschanski. The Ordered and Colored Products in Analytic Combinatorics : Application to the Quantitative Study of Synchronizations in Concurrent Processes. In *Analytic Algorithmics and Combinatorics (ANALCO17)*, pages 16 – 30, Barcelone, Spain, January 2017.

[2] Olivier Bodini, Matthieu Dien, Antoine Genitrini, and Frédéric Peschanski. The Combinatorics of Barrier Synchronization. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 386–405. Springer, 2019.

[3] Olivier Bodini, Matthieu Dien, Antoine Genitrini, and Frédéric Peschanski. Quantitative and Algorithmic aspects of Barrier Synchronization in Concurrency. *Discrete Mathematics and Theoretical Computer Science*, vol. 22 no. 3(3), 2021.

[4] Olivier Bodini, Antoine Genitrini, and Frédéric Peschanski. Enumeration and Random Generation of Concurrent Computations. In Nicolas Broutin and Luc Devroye, editors, *23rd International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'12)*, volume DMTCS Proceedings vol. AQ, 23rd Intern. Meeting on Probabilistic, Combinatorial, and Asymptotic Methods for the Analysis of Algorithms (AofA'12) of *DMTCS Proceedings*, pages 83–96, Montreal, Canada, June 2012. Discrete Mathematics and Theoretical Computer Science.

[5] Olivier Bodini, Antoine Genitrini, and Frédéric Peschanski. The Combinatorics of Non-determinism. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*, volume 24 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 425–436, Guwahati, India, 2013.

[6] Matthieu Dien, Antoine Genitrini, and Frédéric Peschanski. A Combinatorial Study of Async/Await Processes. In *The 19th International Colloquium on Theoretical Aspects of Computing*, volume 13572 of *Lecture Notes in Computer Science*, pages 170–187, Tbilisi, Georgia, September 2022. Springer.

ÉLÉMENT DE PORTFOLIO 03

Logiciel ou bibliothèque logicielle

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : La plateforme MOPSA pour l'analyse statique par interprétation abstraite

URL de l'élément : `https://gitlab.com/mopsa/mopsa-analyzer`

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

MOPSA (*Modular Open Platform for Static Analysis*) est une plateforme logicielle pour la vérification formelle de la correction des programmes par interprétation abstraite. Nous l'avons incluse dans ce portfolio car il s'agit d'un nouveau logiciel, initié en 2017 dans le cadre du projet ERC Consolidator MOPSA [1] (2016–2021), avec Antoine Miné d'APR comme *Principal Investigator*, et elle est un élément central de la recherche en interprétation abstraite dans l'équipe.

Cette plate-forme a une conception originale qui diffère des analyseurs statiques classiques (comme Astrée [2], Frama-C [3] ou Infer [4]) par son architecture très modulaire et extensible. Son rôle est d'aider la recherche, l'enseignement et la diffusion en analyse statique auprès des étudiants, des chercheurs et des industriels. Elle est diffusée publiquement en logiciel libre depuis 2020 sur GitLab. [2]

Au sein de l'équipe, les travaux de recherche ayant mené à la conception de MOPSA et les résultats de recherche originaux obtenus grâce à MOPSA ont fait l'objet de trois thèses soutenues, de trois thèse en cours, d'un travail d'ingénieur sur cinq ans (2016–2021) et de deux post-docs achevés. Une thèse soutenue et une thèse en cours sont réalisées en collaboration industrielle avec les entreprises Airbus et Nomadic Labs. MOPSA a un rôle structurant et agit comme un tremplin pour la recherche en vérification par interprétation abstraite dans APR. Elle continuera à alimenter sa recherche dans les années futures.

Cette réalisation montre que le travail théorique réalisé par l'équipe en interprétation abstraite (définition de sémantiques formelles, d'abstractions, preuves de correction, etc.) est complété par des réalisations pratiques (implantation, expérimentation) démontrant la réelle application des techniques développées sur des langages et des programmes réalistes. Elle illustre également le souhait d'aller au-delà de la réalisation de prototypes académiques pour diffuser dans la communauté des outils robustes, bien documentés et directement exploitables.

3 PRÉSENTATION DE CET ÉLÉMENT

MOPSA est un logiciel d'analyse statique par interprétation abstraite. Les analyseurs statiques sont des outils pour la vérification des logiciels permettant de détecter les erreurs dès la compilation. L'interprétation abstraite, proposée par Patrick et Radhia Cousot dans les années 70 [5], est un cadre formel pour concevoir des analyseurs sûrs, automatiques, basés sur une sémantique mathématique, couvrant intégralement l'espace des exécutions des programmes et sans faux négatif. Ce type d'outil entre donc dans la catégorie des méthodes formelles et permet de prouver des propriétés de correction sur les programmes avec des garanties mathématiques. L'interprétation abstraite résout les problèmes d'indécidabilité et de complexité inhérents en surapproximant les calculs sémantiques dans un univers abstrait, plus simple. Ces surapproximations sont sûres mais parfois imprécises et causent des fausses alarmes. Il est par ailleurs souvent difficile de définir les calculs abstraits pour les constructions des langages modernes, souvent très complexes (par exemple pour les langages dynamiques, comme Python). Le développement d'abstractions sûres, suffisamment précises et efficaces demeure l'enjeu premier de l'interprétation abstraite.

Pour aider dans cette recherche, MOPSA propose une architecture très modulaire et extensible. Comme les analyseurs de l'état de l'art (Astrée [2], Frama-C [3]), MOPSA réalise un calcul sémantique grâce à une multitude

1. `https://mopsa.lip6.fr`
2. `https://gitlab.com/mopsa/mopsa-analyzer`

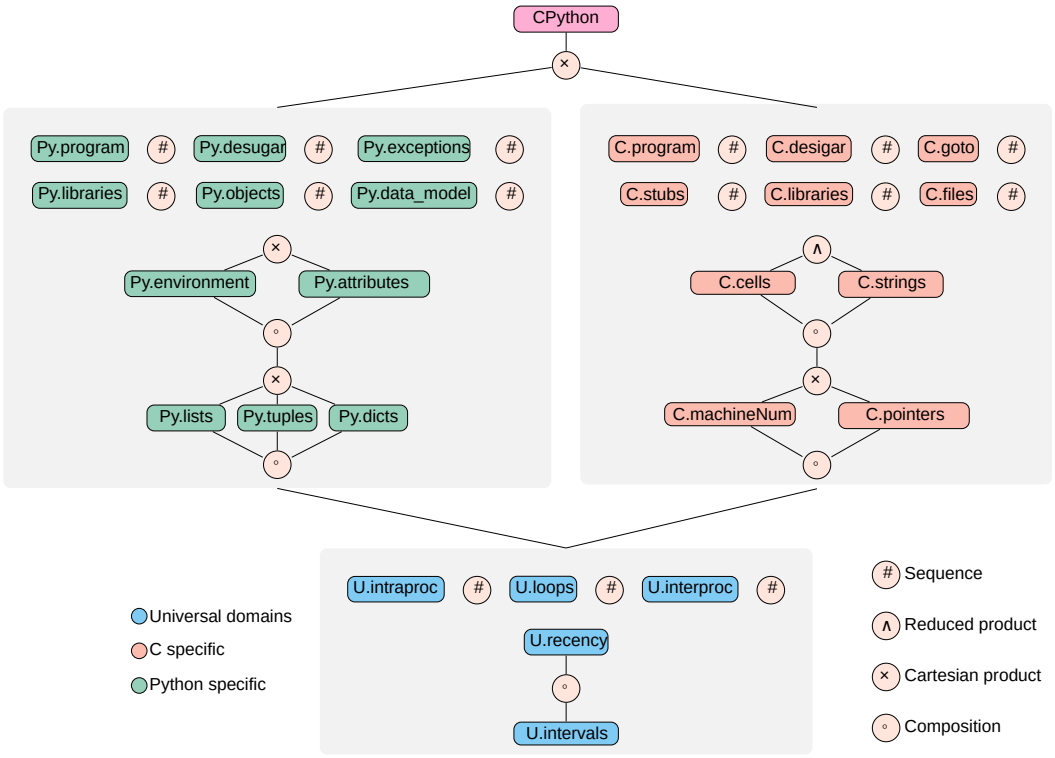


FIGURE 1 – Composition de domaines abstraits dans MOPSA pour l'analyse de valeurs et d'exceptions de programmes Python appelant des fonctions C natives et exploitant la bibliothèque C standard.

d'abstractions communiquantes, mais il possède une architecture beaucoup moins rigide que ces outils : il fait l'hypothèse que les abstractions de tous types de données (numériques, de pointeurs, d'objets, etc.) et les itérateurs sur les constructions syntaxiques du programme sont des abstractions obéissant à une signature commune, permettant d'exprimer des relations complexes entre valeurs et peuvent être combinées sans couplage. Il enrichit également les mécanismes existants de communication entre domaines avec une réécriture dynamique d'expressions.

MOPSA est écrit en OCaml. Son architecture, décrite dans [10], occupe 13 KLOC et est complétée par une bibliothèque d'abstractions génériques (abstraction du tas, intervalles, itérateurs de boucles, etc.) de 25 KLOC. Nous avons implanté dans MOPSA une première analyse, assez classique, des erreurs à l'exécution de programmes C (11 KLOC) [9] et défini un langage de modélisation de bibliothèques [14]. Cela nous a permis d'analyser des programmes C système de taille modeste, comme ceux de Coreutils. Afin de permettre la reproductibilité, ces analyses « benchmark » sont diffusées sous forme de projets compagnons sur Gitlab, et nous participons également à la compétition SV-COMP 2023. Nous avons également mis à contribution MOPSA pour soutenir la recherche en interprétation abstraite au-delà de l'état de l'art, vers des analyses de nouveaux langages et de nouvelles propriétés. Nous avons ainsi développé une analyse de type, de valeur et d'exception pour Python [11, 12] (13 KLOC). Nous avons également développé une analyse originale pour la vérification de programmes multi-langages, mêlant C et Python [13]. La figure 1 montre la combinaison des abstractions utilisées pour cette analyse : outre les domaines génériques (en bas) utilisés dans la plus part des analyses, nous avons inclus les domaines développés indépendemment et spécifiquement pour l'analyse du C (à droite) et de Python (à gauche). Un seul nouveau domaine (en haut), très court (2,5 KLOC), a dû être développé pour gérer les interactions entre les deux langages. Cela démonte la capacité de l'architecture à s'adapter à de nouveaux problèmes tout en réutilisant le travail déjà fourni. Des travaux en cours de finalisation ciblent l'analyse de *smart contracts* sur la *blockchain* Tezos [1], l'analyse de portabilité [6, 8] et de patchs en C [7]. Ils devraient être prochainement intégrés dans la branche principale de MOPSA.

4 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] G. Bau, A. Miné, V. Botbol, and M. Bouaziz. Abstract interpretation of Michelson smart-contracts. In *Proc. of the 11th ACM SIGPLAN Int. Workshop on the State Of the Art in Program Analysis (SOAP'22)*, pages 36–43. ACM, Jun. 2022.

[2] J. Bertrane, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, and X. Rival. Static analysis and verification of aerospace software by abstract interpretation. In *AIAA Infotech@Aerospace*, number 2010-3385, pages 1–38. AIAA, Apr. 2010.

[3] S, Blazy, D. Bühler, and B. Yakobowski. Structuring abstract interpreters through state and value abstractions. In *Verification, Model Checking, and Abstract Interpretation*, pages 112–130. Springer, 2017.

[4] C. Calcagno, D. Distefano, J. Dubreil, D. Gabi, P. Hooimeijer, M. Luca, P. O'Hearn, I. Papakonstantinou, J. Purbrick, and D. Rodriguez. Moving fast with software verification. In *NFM*, pages 3–11. Springer, 2015.

[5] P. Cousot and R. Cousot. Abstract interpretation : A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. of POPL'77*, pages 238–252. ACM, Jan. 1977.

[6] D. Delmas. *Static analysis of program portability by abstract interpretation*. PhD thesis, Sorbonne Université, Dec. 2022.

[7] D. Delmas and A. Miné. Analysis of software patches using numerical abstract interpretation. In *Proc. of the 26th Int. Static Analysis Symp. (SAS'19)*, volume 11822 of *LNCS*, pages 225–246. Springer, Oct. 2019.

[8] D. Delmas, A. Ouadjaout, and A. Miné. Static analysis of endian portability by abstract interpretation. In *Proc. of the 28th Int. Static Analysis Symp. (SAS'21)*, volume 12913 of *LNCS*, pages 102–123. Springer, Oct. 2021.

[9] M. Journault. *Precise and modular static analysis by abstract interpretation for the automatic proof of program soundness and contracts inference*. PhD thesis, Sorbonne Université, Nov. 2019.

[10] M. Journault, A. Miné, R. Monat, and A. Ouadjaout. Combinations of reusable abstract domains for a multilingual static analyzer. In *Proc. of the 11th Working Conf. on Verified Software : Theories, Tools, and Experiments (VSTTE'19)*, volume 12031 of *LNCS*, pages 1–18. Springer, Jul. 2019.

[11] R. Monat. *Static type and value analysis by abstract interpretation of Python programs with native C libraries*. PhD thesis, Sorbonne Université, Nov. 2021.

[12] R. Monat, A. Ouadjaout, and A. Miné. Static type analysis by abstract interpretation of Python programs. In *Proc. of the 34th European Conf. on Object-Oriented Programming (ECOOP'20)*, volume 166 of *Leibniz Int. Proceedings in Informatics (LIPIcs)*, pages 17 :1–17 :29. Dagstuhl Publishing, Jul. 2020.

[13] R. Monat, A. Ouadjaout, and A. Miné. A multilanguage static analysis of Python programs with native C extensions. In *Proc. of the 28th Int. Static Analysis Symp. (SAS'21)*, volume 12913 of *LNCS*, pages 323–345. Springer, Oct. 2021.

[14] A. Ouadjaout and A. Miné. A library modeling language for the static analysis of C programs. In *Proc. of the 27th Int. Static Analysis Symp. (SAS'20)*, volume 12389 of *LNCS*, pages 223–246. Springer, Nov. 2020.

ÉLÉMENT DE PORTFOLIO 04

Publication

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : Philippe Aubry, Jérémy Marrez, Annick Valibouze. Computing real solutions of fuzzy polynomial systems. Fuzzy Sets and Systems, 2020, 399, pp.55 - 76. DOI : 10.1016/j.fss.2020.01.004. Accès libre : hal-02457332.

URL de l'élément : `https://doi.org/10.1016/j.fss.2020.01.004`

Fichier de élément : `AubryMarrezValibouze_VersionJournal_FSS_2020.pdf`

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

Cet article a été choisi par l'équipe APR pour son interface entre plusieurs domaines de l'informatique avec des ramifications de collaborations au sein de SU (Sorbonne Université) en Physique théorique et en Statistiques. Cette thématique de modélisation de problèmes avec des données incertaines comporte d'importantes applications dans des domaines variés comme en ingénierie, en économie et en sciences sociales [1, 2].

Il marque également la réussite d'un travail abouti suite à un stage de Master 2 soutenu financièrement par la commission de soutien à la publication du LIP6 (publication dans une grande revue, implantation en libre accès).

Il s'inscrit à l'interface des domaines informatiques que sont l'Algorithmique, intégrant le parallélisme, l'IA et l'utilisation de la résolution de systèmes d'équations algébriques en Calcul Formel ainsi qu'une implémentation dans le système de Calcul Formel SageMath.

L'implantation d'une version simplifiée et non parallèle de l'algorithme SolveFuzzySystem de l'article par l'un des trois auteurs, Jérémy Marrez, débuta lors de son stage de Master 2 co-encadré au sein de notre équipe APR par les deux autres auteurs de l'article [3]. Cette implantation fut utilisée en 2021 par des étudiants du Master 2 ISDS-IMA de SU, élèves en 3-ième année de l'ISUP, école de Statistiques (centenaire), pour étudier la faisabilité des réseaux neuronaux flous dans le cadre d'une collaboration des auteurs de l'article avec le physicien Bertrand Laforge (Pr SU, LPNHE, UMR 7585) dans l'objectif de concevoir une stratégie d'analyse phénoménologique visant à étudier la nouvelle physique au LHC, le Grand Collisionneur de Hadrons.

3 PRÉSENTATION DE CET ÉLÉMENT

Cet article présente un algorithme efficace appelé SolveFuzzySystem, ou SFS, permettant de trouver les solutions réelles des systèmes algébriques dont les coefficients sont des nombres flous L-R symétriques à support fini et de fonctions de dispersion bijectives.

Les solutions réelles d'un tel système sont déduites des solutions de systèmes algébriques à coefficients réels. L'algorithme est basé sur de nouveaux résultats universels puisqu'indépendants des fonctions de dispersion. Ces résultats théoriques incluent la gestion des signes des solutions des systèmes flous.

L'article décrit une version parallèle de l'algorithme SFS ainsi qu'une implantation séquentielle pour le cas des nombres flous triangulaires dans le paquetage Fuzzy du logiciel libre de Calcul Formel SageMath.

4 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] A. Tacu J. Aluja and editors H. Teodorescu. *Fuzzy Systems in Economy and Engineering*. Publishing House of The Romanian Academy, 1994.

[2] Weldon Lodwick. *Fuzzy surfaces in GIS and geographical analysis*. CRC Press, Taylor & Francis, 2008.

[3] Jérémy Marrez. Fuzzy package, real solving of fuzzy polynomial systems. `https://github.com/JeremyMarrez/Fuzzy/blob/8a74ebf5b25cf1889accec8ef29181506fdcd569/Real%20solving%20of%20fuzzy%20polynomial%20systems`, 2019.

ÉLÉMENT DE PORTFOLIO 05

Logiciel ou bibliothèque logicielle

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : L'environnement OMicroB pour la programmation de microcontrôleurs

URL de l'élément : `https://github.com/stevenvar/OMicroB`

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

OMicroB (*OCaml on Microcontroller Boards*) est un logiciel/environnement de programmation de haut niveau pour la programmation de microcontrôleurs à faibles ressources.

Nous l'avons inclus dans ce portfolio car il s'agit d'une approche originale de programmation qui privilégie la sûreté de fonctionnement, la portabilité du code et l'utilisation parcimonieuse de la mémoire.

Cet environnement est issu de premiers résultats du projet OCaPiC [11] d'exécution du langage OCaml par une machine virtuelle implémentée directement en assembleur PIC. OMicroB généralise cette approche « machine virtuelle » en abstrayant plusieurs familles de microcontrôleurs cibles (voir figure 1) pour expérimenter de façon portable des styles de programmation de haut niveau (fonctionnel/impératif et objet/modulaire) en OCaml, dans un cadre confortable de typage statique.

OMicroB [9] a été développé pour la thèse de Steven Varoumas [5] pour tester de manière effective une extension synchrone dédiée à la gestion des interactions ainsi que l'analyse générique de code-octet pour le calcul du pire temps d'exécution [6]. Le portage d'OMicroB sur l'architecture LCHIP dans le cadre du projet éponyme [1] (FUI : 2016-2020) a permis de proposer une diversification de voies d'exécution de logiciels critiques.

3 PRÉSENTATION DE CET ÉLÉMENT

OMicroB est un environnement de programmation de microcontrôleurs apportant expressivité, sûreté d'exécution et portabilité. Il se compose d'une implantation spécialisée de la machine virtuelle OCaml ainsi que de sa bibliothèque d'exécution (en particulier son gestionnaire automatique de mémoire).

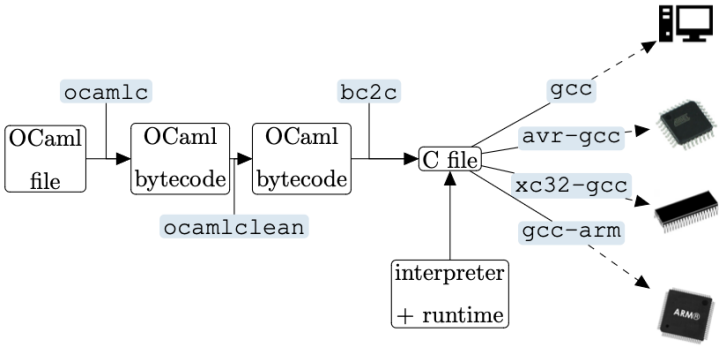


FIGURE 1 – Chaîne de compilation d'OMicroB

La figure 1 montre la chaîne de compilation d'OMicroB sur PC ou ciblant différentes familles de microcontrôleurs. Le programme OCaml est compilé par le compilateur standard OCaml (`ocamlc`). Le fichier de code-octet produit est alors passé à la commande `ocamlclean` qui enlève le code mort, puis au traducteur `bc2c` qui engendre un programme C portable contenant dans un tableau l'ensemble du code-octet ainsi que l'état mémoire obtenu par évaluation partielle (une partie de l'exécution du programme source étant effectuée à la compilation autant que

possible). Ce programme est alors compilé par le compilateur C de la cible et lié avec l'interprète de code-octet et la bibliothèque d'exécution spécifique pour produire un exécutable transférable vers la cible.

Le typage statique fort d'OCaml garantit l'absence d'erreur de typage du programme, et un mode simulateur permet de tester le programme sur ordinateur avant son transfert sur microcontrôleur. L'utilisation parcimonieuse de la mémoire d'OMicroB assure la faible empreinte mémoire des programmes s'exécutant sur des microcontrôleurs à faibles ressources (moins de 32 Ko de code, moins de 4 Ko de mémoire). Différents montages ont été réalisés dans le domaine du divertissement (tempéreuse, jeux à deux joueurs) mais aussi en ciblant le logiciel critique avec l'extension synchrone OCaLustre [8] et l'utilisation dans le projet LCHIP, porté par la société Clearsy, qui propose pour la redondance des calculs cette voie d'exécution spécifique sur une carte munie de deux micro-contrôleurs.

Nous l'utilisons comme un laboratoire pour la montée en abstraction [7] afin de :

- s'abstraire des microcontrôleurs ;
- s'abstraire des composants électronique pour les montages ;
- s'abstraire de la concurrence en proposant un modèle synchrone.

OMicroB est aussi utilisé en cours de « compilation avancée » de M1 du parcours STL (Science et Technologie du Logiciel) de Sorbonne Université, et a été présenté dans un cours invité à la conférence JFLA [10]. L'environnement OMicroB devient mature et propice à une plus large diffusion.

De plus nous utilisons actuellement une version spécifique d'OMicroB appelée O2B [2] pour exécuter OCaml sur un processeur softcore (comme le Nios2) embarqué sur du matériel reconfigurable FPGA dans le but de s'interfaçer avec d'autres circuits [4]. O2B est notamment utilisé en combinaison avec le compilateur Macle pour l'accélération matérielle de programmes OCaml sur FPGA [3].

4 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Thierry Lecomte, David Déharbe, Denis Sabatier, Etienne Prun, Patrick Péronne, Emmanuel Chailloux, Steven Varoumas, Adilla Susungi, and Sylvain Conchon. Low Cost High Integrity Platform. In *ERTS 2020 - 10th European Congress on Embedded Real Time Systems*, January 2020.

[2] Jocelyn Sérot and Emmanuel Chailloux. OCaml sur circuit FPGA. In *JFLA 2021 - 32 èmes Journées Francophones des Langages Applicatifs*, en ligne, France, April 2021.

[3] Loïc Sylvestre, Emmanuel Chailloux, and Jocelyn Sérot. Accelerating OCaml Programs on FPGA. *International Journal of Parallel Programming*, 2023.

[4] Loïc Sylvestre, Jocelyn Sérot, and Emmanuel Chailloux. A Virtual Machine Approach for High-level FPGA Programming. In *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 1–1, New York City, United States, May 2022. IEEE.

[5] Steven Varoumas. *Modèles de programmation de haut niveau pour microcontrôleurs à faibles ressources*. Theses, Sorbonne Université, November 2019.

[6] Steven Varoumas and Tristan Crolard. WCET of OCaml Bytecode on Microcontrollers : An Automated Method and Its Formalisation. In Sebastian Altmeyer, editor, *19th International Workshop on Worst-Case Execution Time Analysis (WCET 2019)*, volume 72 of *OpenAccess Series in Informatics (OASIcs)*, pages 5 :1–5 :12, Stuttgart, Germany, July 2019. Schloss Dagstuhl.

[7] Steven Varoumas, Basile Pesin, Benoît Vaugon, and Emmanuel Chailloux. Programming microcontrollers through high-level abstractions. In *VMIL 2020 - 12th ACM SIGPLAN International Workshop on Virtual Machine and Intermediate Languages*, pages 5–14, November 2020.

[8] Steven Varoumas, Benoît Vaugon, and Emmanuel Chailloux. Concurrent Programming of Microcontrollers, a Virtual Machine Approach. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, pages 711–720, January 2016.

[9] Steven Varoumas, Benoît Vaugon, and Emmanuel Chailloux. A Generic Virtual Machine Approach for Programming Microcontrollers : the OMicroB Project. In *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, January 2018.

[10] Steven Varoumas, Benoît Vaugon, and Emmanuel Chailloux. La programmation de microcontrôleurs dans des langages de haut niveau - Cours invité. In *JFLA 2018*, Vingt-neuvièmes Journées Francophones des Langages Applicatifs (JFLA 2018), BANYULS, France, January 2018.

[11] Benoît Vaugon, Philippe Wang, and Emmanuel Chailloux. Programming Microcontrollers in Ocaml : the OCaPIC Project. In *International Symposium on Practical Aspects of Declarative Languages (PADL 2015)*, volume 9131 of *LNCS*, pages 132–148. Springer Verlag, June 2015.

ÉLÉMENT DE PORTFOLIO 06

Publication

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : Causal Computational Complexity of Distributed Processes

URL de l'élément : `https://hal.science/hal-02074534v1`

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

Cet élément a été choisi parce qu'il représente :

1. un lien entre deux mondes des méthodes formelles (la théorie de la concurrence et la complexité implicite) ;
2. une collaboration entre l'équipe APR et *Imperial College* (puis *Oxford*).

L'équipe APR s'efforce de bâtir des ponts entre des domaines distincts de l'algorithmique et de la programmation. À ce titre, cet article représente un exemple d'un rapprochement entre deux thématiques non-connexes :

- Le *monde de la complexité implicite* d'une part : il s'agit d'un domaine, actif depuis une vingtaine d'années, qui vise à produire des résultats de complexité (en temps, en espace) *a priori* pour des algorithmes à partir de contraintes syntaxiques (restriction du langage) ou d'analyses statiques (systèmes de types). L'idée principale est de construire un cadre formel dans lequel tous les programmes *définissables* (soit syntaxiquement, soit après une étape de vérification de types) satisfont une borne de complexité.
- Le monde des *algèbre de processus* d'autre part : un domaine dans lequel des abstractions mathématiques de programmes (ou protocoles) concurrents et distribués sont définies et étudiées.

Cet élément a été publié dans une conférence internationale majeur du domaine (LICS).

Le caractère pertinent de cet élément s'exprime dans :

- la définition d'une notion de complexité nouvelle (complexité causale des messages générés dans un réseau de services) ;
- l'adaptation d'une technique classique d'un domaine (la complexité implicite) à un autre domaine (les algèbres de processus) ;
- l'introduction d'analyses spécifiques supplémentaires qui prennent en compte le caractère particulier du cadre concurrent.

3 PRÉSENTATION DE CET ÉLÉMENT

Cet article décrit une méthode d'analyse pour les algèbres de processsus (des modèles de programmes et protocoles distribués) qui décrit que le nombre global de messages générés par un appel au service est borné par un polynôme sur la taille de cet appel.

Le formalisme utilisé dans l'article est celui du pi-calcul, un langage mathématique (une algèbre de processus) décrivant les actions observables des différentes composantes d'un système. Ce formalisme permet de décrire facilement des services, acceptant des requêtes, générant des messages (par exemple, des appels à d'autres services), et renvoyant finalement une réponse à la requête initiale. Des services arithmétiques (calculant des fonctions simples des entiers dans les entiers) sont utilisés comme exemples, bien que l'analyse s'applique de la même façon à des services manipulant des données complexes. Les services peuvent être construits de manière récursive (un service peut envoyer une requête à lui-même pour obtenir une réponse intermédiaire). Certaines définitions de services récursifs génèrent un nombre exponentiel de messages (par exemple, un service récursif pour le calcul de la factorielle, qui appelle un service récursif pour le calcul de la multiplication, qui lui même appelle un service récursif pour le calcul de l'addition). L'objectif de l'analyse présentée dans le portfolio est de détecter et de rejeter de tels services.

3.1 Complexité Causale

L'article s'attelle d'abord à définir une notion de complexité *en messages* pour les réseaux de services. L'idée est, pour la première fois dans ce domaine, de lier la complexité d'un processus au nombre de communications générées par un appel externe. Cela nécessite l'introduction d'une notion de complexité causale, qui permet de lier à chaque communication, la requête initiale qui l'a causée, et ainsi d'obtenir une définition dynamique de la complexité : plusieurs requêtes différentes peuvent être traitées simultanément. Ce que garantit l'analyse, c'est que les messages causalement produits par une requête sont bornés par une fonction en la taille de la requête.

3.2 Adaptation de [1]

L'article [1] est une article important du domaine de la complexité implicite, qui caractérise les fonctions polynomiales à l'aide de règles syntaxiques simples : toutes les fonctions séparent leurs paramètres en deux ensembles : ceux qui sont $\ll$ sûrs $\gg$ et les autres. L'analyse empêche les produits de récursion (les résultats obtenus par un appel récursif) à être utilisés comme argument non-sûr d'une fonction. Les auteurs montrent que cette discipline de séparation suffit à caractériser les fonctions récursives ayant une complexité en temps.

Dans cet article, on adapte dans un premier temps l'analyse de [1] à un cadre concurrent, à l'aide d'un système de types. Les règles de typage garantissent une division sûr/non-sûr des contenus des messages, et donc des requêtes aux différents services.

3.3 Perturbations concurrentes et solution

Pourtant, cette analyse n'est pas correcte. En effet, le caractère concurrent du cadre peut tromper l'analyse, en perturbant le flot de calcul d'un service à l'aide de messages externes portant des contenus non-bornés et susceptibles de produire des calculs arbitrairement grand (et donc des messages arbitrairement nombreux). Pour que les contraintes du système de types soient respectées, il est nécessaire d'ajouter une étude du flot de calul (de la provenance et du devenir des contenus des messages échangés au sein du système).

La combinaison des deux analyses (systèmes de types et analyse d'origines) permet d'obtenir la borne polynomiale souhaitée.

4 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Stephen J. Bellantoni and Stephen A. Cook. A new recursion-theoretic characterization of the polytime functions. *Comput. Complex.*, 2 :97–110, 1992.