Les systèmes faiblement synchrones avec trois machines sont Turing-complets

Cinzia Di Giusto, Davide Ferré, Étienne Lozes, Nicolas Nisse

▶ **To cite this version:**

Cinzia Di Giusto, Davide Ferré, Étienne Lozes, Nicolas Nisse. Les systèmes faiblement synchrones avec trois machines sont Turing-complets. AlgoTel 2024 – 26èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, May 2024, Saint-Briac-sur-Mer, France. hal-04551070

HAL Id: hal-04551070
https://hal.science/hal-04551070

Submitted on 18 Apr 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Les systèmes faiblement synchrones avec trois machines sont Turing-complets[†]

Cinzia Di Giusto[1] et Davide Ferré[2] et Étienne Lozes[1] et Nicolas Nisse[2]

[1] *Université Côte d'Azur, CNRS, I3S, France*
[2] *Université Côte d'Azur, Inria, CNRS, I3S, France*

Les automates finis communicants permettent de modéliser les systèmes distribués asynchrones à passage de messages. Dans les systèmes faiblement synchrones, les processus communiquent par le biais de phases au cours desquelles chaque processus envoie d'abord tous ses messages, puis reçoit tous les messages des autres processus. Ces systèmes bénéficient d'une forme limitée de synchronisation et, pour certains modèles de communication, cette restriction est suffisante pour que le problème d'accessibilité devienne décidable. En particulier, nous explorons le cas de la communication *p2p* (FIFO), pour lequel le problème d'accessibilité est connu pour être indécidable pour quatre processus, mais décidable pour deux. Nous montrons que le problème d'accessibilité pour les systèmes faiblement synchrones de trois processus est indécidable. Ce résultat est fortement inspiré par notre étude de la largeur arborescente des diagrammes de séquence de messages (MSC pour *Message Sequence Chart*) qui peuvent être générés par de tels systèmes. En ce sens, la principale contribution de ce travail est un système faiblement synchrone avec trois processus qui génère des MSCs d'une largeur arborescente arbitrairement grande.

Mots-clefs : Automates finis communiquants, Problème d'accessibilité, Graphes, Largeur arborescente

1 Introduction

CFM. Les automates finis communicants (*communicating finite-state machines*, CFM) constituent un modèle simple, mais expressif, de systèmes distribués asynchrones à passage de messages, où chaque machine (processus) effectue une séquence d'actions d'envoi et de réception de messages (soit $\mathbb{M}$, l'ensemble des messages), où une action d'envoi peut correspondre à une action de réception d'un autre processus (notons qu'*a priori*, il n'est pas nécessaire qu'un message envoyé soit reçu). La Figure 1 (gauche) représente les CFMs de 3 processus a, b et c. Dans cet exemple, le processus a dans l'état 0 peut envoyer un message $m_1 \in \mathbb{M}$ à la machine (au processus) b, auquel cas, a passe dans l'état 1. Dans l'état 1, le processus a peut envoyer un nombre arbitraire de messages m_2 au processus c et repasse dans l'état 0 s'il reçoit (lit) un message m_3 du processus b. Sauf mention contraire, un processus ne peut pas s'envoyer un message à lui-même.

MSC. Une exécution d'un tel système peut être représentée graphiquement par un diagramme de séquence de messages (*Message Sequence Chart*, MSC). Chaque processus du système a sa propre "chronologie" sur le MSC (les lignes verticales représentent le "déroulement du temps" de chaque processus), où les actions sont énumérées dans l'ordre dans lequel elles sont effectuées, et des flèches (d'une ligne verticale d'un processus à une autre) relient une action d'envoi à l'action de réception correspondante. Le MSC de la Figure 1 (droite) représente l'une des nombreuses exécutions possibles du système de la Figure 1 (gauche).

Modèles de communication. L'ensemble de tous les MSCs qu'un système peut générer est déterminé à la fois par les machines (par les CFMs), puisque la séquence d'actions de chaque ligne de temps doit être une séquence d'actions dans le CFM correspondant, mais aussi par le *modèle de communication* utilisé par les machines. Informellement, un modèle de communication est une classe de MSCs qui sont considérés comme "réalisables" dans le cadre de ce modèle de communication. Par exemple, un MSC est considéré comme

[†] Ce travail a été publié dans [GFLN23]. La version longue est accessible ici : `https://hal.science/hal-04182953v1/file/HALconference.pdf`.

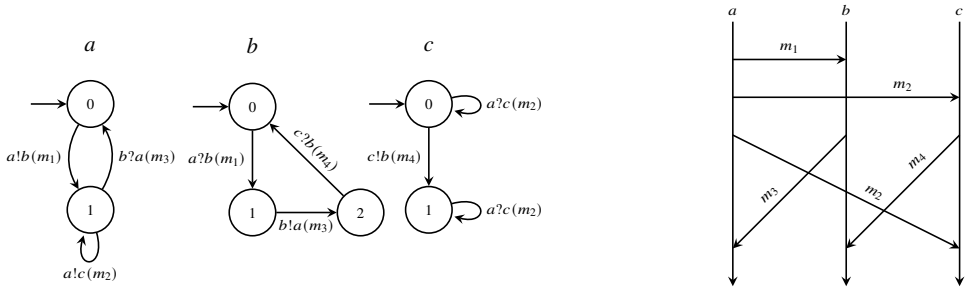

FIGURE 1: À gauche, exemple d'un système de 3 processus a, b et c. Une transition $x!y(m)$ (resp. $x?y(m)$) entre deux états E_1 et E_2 de l'automate décrivant le processus x signifie que x passe de l'état E_1 à E_2 en envoyant le message $m \in \mathbb{M}$ au processus y (resp., lorsque y reçoit un message m du processus x). À droite, la MSC d'une exécution possible (synchrone) de ces automates : a envoie le message m_1 qui est reçu par b, puis le message m_2 est envoyé par a et reçu par c. Enfin, les messages m_2, m_3 et m_4 sont envoyés par a, b et c et ensuite reçus respectivement par c, a et b.

réalisable pour un modèle de communication *synchrone* si, pour chaque message m, il existe un unique chemin (orienté) de l'événement d'envoi de m à l'événement de réception de m. À l'opposé, le modèle de communication *asynchrone* n'impose aucune contrainte. Parmi les nombreux modèles de communication qui ont été envisagés [DFLL23], citons le modèle *boîte aux lettres* où chaque machine détient une file d'attente FIFO unique pour tous les messages entrants, et le modèle *p2p* où chaque paire (source/destination de messages) de processus a une file FIFO dédiée. Ces différents modèles forment une "hiérarchie de synchronisme", le modèle *p2p* étant "moins" synchrone que le modèle boîte aux lettres, lui-même "moins" synchrone que le modèle synchrone. Dans le modèle *faiblement synchrone*, les processus communiquent par phases au cours desquelles les messages sont d'abord tous envoyés, puis les messages sont tous reçus, pour chaque processus. Graphiquement, les MSCs suivant ce modèle de communication sont la concaténation de MSCs plus petits et indépendants, au sein desquels aucune émission n'a lieu après une réception. Par exemple, le MSC de la Figure 1 (droite) est faiblement synchrone, car il est la concaténation de trois "phases" (à savoir $\{m_1\}$, $\{m_2\}$, et $\{m_3, m_4, m_2\}$), au sein desquelles tous les envois d'un processus donné se produisent avant toutes les réceptions de ce même processus. Ces différents modèles de communication ont été proposés afin de déterminer un compromis entre l'expressivité d'un modèle de communication (jusqu'à être Turing complet) et le fait qu'on puisse résoudre (efficacement) les questions que l'on s'y pose.

Accessibilité. Par exemple, un problème classique est celui qui prend un CFM, un modèle de communication et un MSC en entrées et veut décider si le MSC correspond à une exécution possible du CFM dans le modèle de communication. Un autre problème important, celui de l'*accessibilité*, prend un CFM, un modèle de communication et une configuration (un état donné pour chaque processus et pour chaque file d'attente des messages) en entrées et demande si cette configuration peut être atteinte par le CFM dans ce modèle de communication à partir de la configuration initiale. Ce dernier problème est connu pour être décidable pour une communication synchrone, puisque l'espace d'état du système est fini, et aussi pour le modèle de communication asynchrone par réduction aux réseaux de Petri [May84]. On sait également que l'accessibilité est décidable pour les systèmes faiblement synchrones de boîtes aux lettres [BDF+21], alors qu'elle est indécidable pour les systèmes p2p avec au moins quatre machines. En revanche, l'accessibilité est décidable pour deux machines (puisque tout MSC p2p avec deux machines satisfait aussi le modèle boîte aux lettres).

Largeur arborescente. Notons qu'un MSC peut être vu comme un graphe (orienté) sous-cubique (tout sommet a degré au plus 3) en considérant chaque événement (envoi ou réception d'un message) comme un sommet, et deux événements successifs d'un même processus, ou deux événements correspondants à l'envoi et à la reception d'un message, correspondent à des sommets adjacents. Une façon de résoudre des problèmes qui se posent au sujet d'un CFM est d'utiliser la structure des graphes des MSC correspondants à ce CFM. La largeur arborescente (qui informellement mesure la "proximité" d'un graphe à un arbre) est un paramètre de graphe important dans ce contexte. En particulier, il est connu que de nombreux problèmes (ceux exprimables en logique monadique du second ordre) peuvent être résolus en temps polynomial dans la classe des graphes

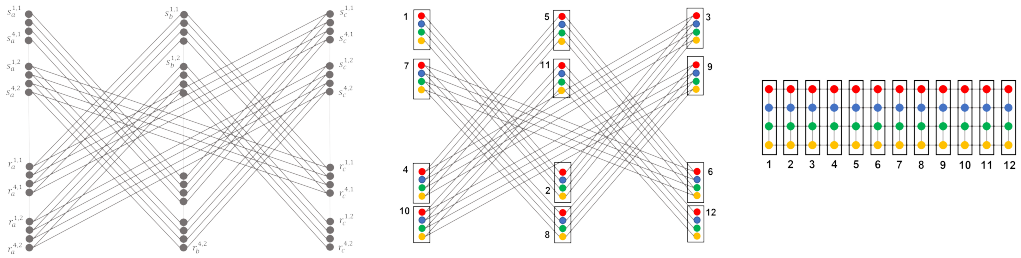

FIGURE 2: À gauche, un MSC avec 3 processus, p2p faiblement synchrone, dont le graphe correspondant contient une grille 4×12 comme mineur (les deux autres figures illustrent l'obtention de la grille). Notons que pour une meilleure lisibilité, certains arcs ont déjà été contractés dans la figure de gauche (les sommets de degré 4, sont en fait la contraction de deux sommets, adjacents sur une même ligne de temps, de degré trois).

de largeur arborescente bornée [CM93]. Ainsi, si un CFM et un modèle de communication ne génèrent que des graphes de largeur arborescente bornée, alors des problèmes comme celui de l'accessibilité, ou celui de déterminer si le langage reconnu par le système est vide ou non, peuvent être résolus efficacement. Par exemple, la classe des MSC de type boîte aux lettres faiblement synchrone a une largeur arborescente bornée [BDF+21].

Nos contributions. Dans ce travail, nous traitons des systèmes p2p faiblement synchrones avec trois machines, et concluons que l'accessibilité est indécidable pour ces systèmes. Nous pensons que l'apport principal est la technique de preuve que nous avons utilisée pour obtenir ce résultat. Notre résultat est basé sur une étude de la largeur arborescente des MSCs qui peuvent être générés par ces systèmes. La première contribution de ce travail est une famille de systèmes p2p faiblement synchrones avec seulement trois machines qui peuvent générer des MSCs d'une largeur arborescente arbitrairement grande. La deuxième contribution, fortement inspirée du système précédent, est de montrer que les systèmes p2p faiblement synchrones avec trois processus sont Turing-complet. Pour ce faire, nous établissons une bijection entre les calculs d'un automate FIFO (connu pour être un modèle de calcul Turing-complet) d'une part, et un sous-ensemble des MSCs de largeur arborescente non bornée.

2 Systèmes avec largeur arborescente non bornée

Du fait de la limitation d'espace, nous ne donnons que les grandes lignes de la preuve du théorème suivant.

Théorème 2.1 *La classe des systèmes p2p faiblement synchrones (à une seule phase) avec trois processus a une largeur arborescente non bornée.*

Tout d'abord, rappelons que la largeur arborescente est close par mineur. C'est-à-dire que la largeur arborescente d'un graphe G qui contient un graphe H comme *mineur* (H peut être obtenu de G en supprimant des sommets et/ou des arêtes et/ou en contractant des arêtes) est au moins la largeur arborescente de H. De plus la largeur arborescente d'une grille $n \times m$ vaut $\min\{n, m\}$. Pour prouver le théorème, il suffit donc de concevoir une famille de MSCs p2p faiblement synchrones, avec trois processus, dont les graphes correspondants contiennent des grilles arbitrairement grandes comme mineurs.

Pour tous $x, y \in \mathbb{N}$, nous concevons un système où chaque processus envoie tout d'abord x "blocs" de $2y$ messages, puis reçoit x blocs de $2y$ messages (exceptés le premier bloc du premier processus et le dernier bloc du troisième processus qui ne contiennent que y messages). Dans l'exemple de la Figure 2 (que l'on peut généraliser facilement), $x = 2$ et $y = 4$. Ce système est p2p faiblement synchrone (à une phase) et nous montrons que le graphe correspondant contient une grille $y \times 6x$ comme mineur, ce qui prouve le théorème.

Notre résultat s'étend directement au modèle de communication CO (*Causally Ordered*) où l'ordre des messages doit respecter la causalité. De plus, si on autorise un processus à s'envoyer des messages à lui-même, en utilisant une construction similaire, nous prouvons :

Théorème 2.2 *Pour tout graphe H, il existe un système p2p faiblement synchrone avec quatre processus (dont un peut s'envoyer des messages) générant un MSC dont le graphe contient H comme mineur.*

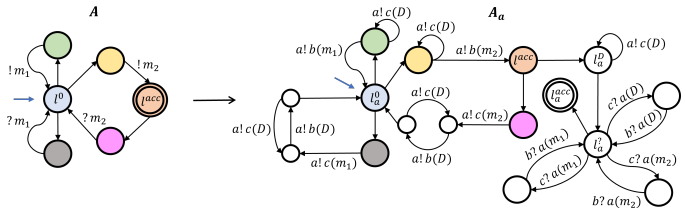

FIGURE 3: À droite, l'automate A_a construit à partir d'un exemple d'automate $\mathcal{A}$ (gauche) d'un processus FIFO.

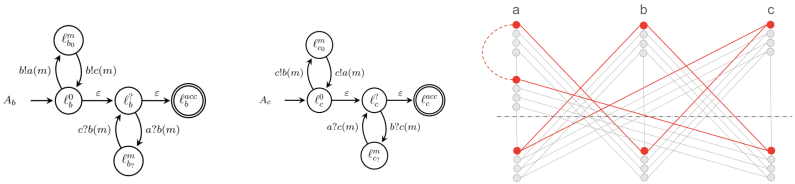

FIGURE 4: À gauche, les automates A_b et A_c. À droite, un message que s'envoie la machine $\mathcal{A}$ (figuré en pointillés rouge) et son équivalent réalisé par la machine $\mathcal{A}'$ grace à un échange de 6 messages (en rouge plein) entre les 3 processus a, b et c.

3 Indécidabilité du problème d'accessibilité

Il est connu que, dans la classe des machines à un processus (qui donc peut s'envoyer des messages) et avec un modèle de communication FIFO, le problème de l'accessibilité est indécidable. Nous montrons que toute machine $\mathcal{A}$ dans cette classe peut être simulée par un MFC $\mathcal{A}'$ p2p faiblement synchrone avec 3 processus. C'est-à-dire que nous construisons un MFC $\mathcal{A}'$ qui reconnait exactement le même langage que $\mathcal{A}$. Cela nous permet de prouver le théorème suivant, pour lequel nous ne donnons qu'une esquisse de preuve.

Théorème 3.1 *La classe des systèmes p2p faiblement synchrones avec* 3 *processus est Turing-complet.*

Soit $\mathcal{A}$ une machine FIFO à un processus avec un ensemble de messages $\mathbb{M}$. Nous construisons un MFC $\mathcal{A}'$ (p2p et faiblement synchrone) à l'aide des automates A_a, A_b et A_c de trois processus a, b et c (sur un alphabet $\mathbb{M} \cup \{D\}$ augmenté d'un "dummy" message D) tel que tout mot $m' \in \mathbb{M} \cup \{D\}$ est reconnu par $\mathcal{A}'$ ssi le mot m (obtenu de m' en ignorant la lettre D) est reconnu par $\mathcal{A}$. Intuitivement, la machine $\mathcal{A}$ est simulée par l'automate A_a (voir un exemple sur la Figure 3), où l'envoi d'un message m, $\mathcal{A}!\mathcal{A}(m)$, devient un envoi de a à b, $A_a!A_b(m)$, et la réception $\mathcal{A}?\mathcal{A}(m)$ devient un envoi de a à c, $A_a!A_c(m)$. Afin de corréler ces deux envois dans une discipline FIFO, A_b et A_c "devinent" le contenu de la file FIFO et le transmettent aux deux autres machines. Dans la deuxième moitié de la phase (réception), A_a vérifie que A_b et A_c ont bien deviné le même contenu de file, et A_b et A_c effectuent des réceptions qui permettent de créer un "lien FIFO" entre l'envoi $A_a!A_b(m)$ et l'envoi $A_a!A_c(m)$. Les deux automates A_b et A_c sont fortement inspirés des systèmes proposés dans la Section 2 comme on peut le visualiser sur la Figure 4 où l'on visualise comment un message (rouge pointillé) de $\mathcal{A}$ est "traduit" dans $\mathcal{A}'$ (en rouge plein).

Le théorème 3.1 s'étend directement au modèle de communication CO et nous pensons que notre technique de preuve permettra d'obtenir des résultats similaires pour d'autres modèles de communication.

References

[BDF+21] B. Bollig, C. Di Giusto, A. Finkel, L. Laversa, E. Lozes, and A. Suresh. A unifying framework for deciding synchronizability. In *32nd Int. Conf. on Concurrency Theory (CONCUR)*, volume 203 of *LIPIcs*, pages 14:1–14:18, Virtual Conference, 2021.

[CM93] Bruno Courcelle and Mohamed Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comput. Sci.*, 109(1&2):49–82, 1993.

[DFLL23] Cinzia Di Giusto, Davide Ferré, Laetitia Laversa, and Étienne Lozes. A partial order view of message-passing communication models. *Proc. ACM Program. Lang.*, 7(POPL):1601–1627, 2023.

[GFLN23] C. Di Giusto, D. Ferré, É. Lozes, and N. Nisse. Weakly synchronous systems with three machines are turing powerful. In *17th Int. Conf. on Reachability Problems (RP'23)*, volume 14235 of *LNCS*, pages 28–41. Springer, 2023.

[May84] Ernst W. Mayr. An algorithm for the general petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.