THÔNG TIN - DIỄN ĐÀN

TOÁN HỌC THỜI 4.0 VÀ DỰ ÁN "FORMAL ABSTRACT IN MATHEMATICS" (FAB)

Hà Huy Khoái[*], Nguyễn Thị Huyền Châu[**], Nguyễn Thị Trà My[**], Mai Thúy Nga[***], Ngô Thị Thanh Nga[****]

Nhận bài: 20/07/2021; Chấp nhận đăng: 10/08/2021

Tóm tắt

Trong bài báo này chúng tôi đề cập về vai trò của máy tính trong nghiên cứu Toán học dưới tác động của cách mạng công nghiệp lần thứ 4, và giới thiệu về dự án FAB "Formal Abstract in Mathematics", một dự án mà Trường Đại học Thăng Long đã tham gia thực hiện cùng với hai trường đại học lớn của Mỹ là University of Pittsburgh và Carnegie Mellon University. Chúng tôi cũng giới thiệu những nét căn bản về hệ chứng minh hình thức Lean (ngôn ngữ được dùng để viết hình thức hóa các định lý trong 100 định lý phổ biến của toán học, ở giai đoạn đầu của dự án) và CNL (Controlled Natural Language, sử dụng ở giai đoạn sau của dự án).

Từ khóa: *Trừu xuất hình thức; Chứng minh hình thức; Ngôn ngữ tự nhiên có kiểm soát, Colada, Lean*,

1. Toán học thời 4.0 và dự án FAB

1.1. Archimede với đòn bẩy

Người ta thường nhắc câu nói nổi tiếng của Archimede: "Hãy cho tôi một điểm tựa, tôi sẽ nâng được cả Trái đất!" Nói cách khác, với Archimede thì cái đòn bẩy tạo sự bình đẳng về sức lực cơ bắp giữa người yếu và người khoẻ.

Vậy thì cái gì có thể xem là đòn bẩy cho "sức lực" của trí tuệ? Không gì khác hơn là máy tính (tất nhiên với những chương trình ngày càng mạnh do con người tạo ra). Ngay với những lĩnh vực tư duy có tính cá thể cao như Toán học, máy tính đang mang đến những thay đổi khó hình dung. "Nghề làm Toán trong thời đại 4.0" có thể sẽ rất khác với những gì vẫn được hiểu hàng ngàn năm nay. Cuộc cách mạng công nghiệp lần thứ 4 không chỉ tạo ra làn gió mới, mà thậm chí là cơn bão trong Toán học, và kéo theo nó, không những là hứng khởi, hân hoan, mà cả sự nghi ngờ, tranh cãi nữa.

[*] Viện Toán học và Khoa học ứng dụng Thăng Long (TIMAS), Trường Đại học Thăng Long
[**] Phòng thí nghiệm Trí tuệ nhân tạo, Trường Đại học Thăng Long
[***] Bộ môn Tin, Khoa Toán - Tin, Trường Đại học Thăng Long
[****] Bộ môn Toán, Khoa Toán - Tin, Trường Đại học Thăng Long

Những vấn đề cốt lõi lại một lần nữa được đặt ra: thế nào là "chân lý Toán học", thế nào là một "chứng minh"? Phần viết này nhằm giới thiệu với bạn đọc một số vấn đề xuất hiện khi sử dụng máy tính trong chứng minh Toán học, và hơn nữa, trong việc thay thế dần lao động của nhà Toán học.

1.2. Chứng minh Toán học

Vào thế kỉ VI và VII trước Công nguyên, các học giả Hy Lạp đã đưa ra cái mà về sau gọi là *suy luận logic*: đó là chuỗi các suy luận – các phép tam đoạn luận – chúng *buộc* người đối thoại chấp nhận khẳng định Q một khi đã đồng ý một khẳng định P trước đó. Ta biết rằng, từ thế kỷ thứ V trước công nguyên, các nhà tư tưởng Hy Lạp đã là những bậc thầy về nghệ thuật sắp xếp lý luận thành một chuỗi liên tiếp các kết luận logic, điều này được thấy rõ trong các tác phẩm của những nhà ngụy biện, cũng như trong các đoạn đối thoại của Plato. Họ khám phá ra rằng, những lý luận này có thể lấy bất cứ hoạt động nào của con người làm đối tượng, đặc biệt là những công thức toán học và hình học, hầu hết trong số đó đến từ nền văn minh Ai Cập và Babylon. Những lý luận này trở thành *chứng minh* kết nối những định lý với nhau. Những chứng minh đầu tiên được tìm thấy trong *Analytica Posteriora* của Aristotle, theo đó, các định lý được suy ra từ một chuỗi lập luận mà người đọc hiểu được không cần giải thích gì thêm, và một số tiên đề được chấp nhận.

Chứng minh toán học theo hình mẫu của Aristotle và Euclid có hai đặc điểm nổi bật là *bỏ qua nhiều suy luận logic trung gian* và dựa nhiều vào *trực giác*.

Bỏ qua nhiều lập luận logic trung gian nghĩa là trong các lập luận, nhà Toán học luôn dừng lại ở mức mà người đọc "hiểu được". Chính vì thế mà trong công việc của thầy giáo Toán, một phần lớn là làm cho học sinh hiểu được những lập luận logic trung gian đã bị bỏ qua trong chứng minh.

Mặt khác, các yếu tố *trực giác* thường được sử dụng, nhất là trong các chứng minh hình học, và cả trong những lĩnh vực Toán học hiện đại như Tô pô. Các nhà Toán học sử dụng trực giác để làm ngắn gọn trình bày chứng minh, và tất nhiên khi cần, họ phải có lập luận logic chặt chẽ mà không viện đến trực giác.

Với việc bỏ qua một số lập luận logic và sự tham gia của trực giác, vấn đề đặt ra là: các định lý Toán học có *đáng tin cậy hay không?* Chúng ta thường tin và sử dụng trong công việc của mình những định lý mới, những kết quả mới của Toán học, nếu đó là kết quả của nhà Toán học "có uy tín", và đăng tải trên những tạp chí "đáng tin cậy". Tuy nhiên, rõ ràng điều này không đảm bảo tuyệt đối cho sự đúng đắn của các định lý đó. Nhà Toán học nào cũng có thể sai lầm khi bỏ qua các lập luận logic trung gian, và người duyệt đăng cũng vậy. Trong bài này, chúng tôi cũng sẽ đưa ra một số ví dụ về sai lầm của các nhà Toán học.

Với những định lý mà chứng minh tương đối ngắn, người ta có thể kiểm tra từng dòng chứng minh để bảo đảm những lập luận trung gian đã bị bỏ qua thực sự là đúng đắn. Tuy nhiên điều này không dễ khi chứng minh dài khoảng 100 trang, hay hơn nữa. Đặc biệt, với những chứng minh có sự trợ giúp của máy tính thì tính đúng đắn của nó là điều không dễ chấp nhận. Đặc điểm chung của những chứng minh có trợ giúp của máy tính là: bằng những lập luận toán học, người ta đưa bài toán về việc kiểm tra *hữu hạn* trường hợp. Việc kiểm tra này được giao cho máy tính. Tuy nhiên, trong nhiều chứng minh, số hữu hạn trường hợp

này là quá lớn, đến nỗi người ta khó có thể tin hoàn toàn vào công việc của máy tính. Sự tranh luận trong giới toán học về việc có thể xem chứng minh với sự trợ giúp của máy tính là một *chứng minh* hay không trở nên đặc biệt sôi nổi sau khi xuất hiện chứng minh của Bài toán 4 màu.

1.3. Bài toán 4 màu và chứng minh hình thức

Có thể dễ dàng chứng minh rằng, với mọi bản đồ tuỳ ý, ta có thể dùng 5 màu để tô, mỗi nước một màu, sao cho hai nước có chung phần biên giới sẽ được tô bằng hai màu khác nhau.

Năm 1852, Francis Guthrie đưa ra giả thuyết rằng, *có thể dùng 4 màu để tô bản đồ tuỳ ý với yêu cầu hai nước có phần biên giới chung phải có màu khác nhau.*

Giả thuyết trên được Cayley phát biểu chính thức vào năm 1878, và nổi tiếng với tên gọi *Bài toán 4 màu.*

Từ sau khi ra đời, Bài toán 4 màu đã nhận được rất nhiều "chứng minh", và rồi lại được chỉ ra "chứng minh" là sai, kể cả chứng minh của một số nhà toán học nổi tiếng. Xin kể ra đây vài ví dụ:

- Năm 1879, Kempe công bố một chứng minh.

- Năm1880, Tait công bố một chứng minh khác.

- Năm 1890, Heawood chỉ ra cái sai trong chứng minh của Kempe (11 năm sau khi chứng minh được công bố).

- Năm 1891, Petersen phát hiện chứng minh của Tait cũng sai!

- Năm 1976, một sự kiện xôn xao giới toán học: Appel và Haken công bố chứng minh *Giả thuyết 4 màu,* một chứng minh có sự trợ giúp của máy tính.

Có thể tóm tắt chứng minh của Appel và Haken như sau. Trước tiên, ta đưa bài toán 4 màu về bài toán trên đồ thị. Mỗi nước được thay bởi một đỉnh của đồ thị, hai đỉnh của đồ thị được nối bởi một cạnh khi và chỉ khi hai nước tương ứng có chung một phần biên giới. Như vậy, giả thuyết 4 màu được đưa về giả thuyết sau: *có thể dùng 4 màu để tô các đỉnh của một đồ thị phẳng tuỳ ý, sao cho hai đỉnh kề luôn có màu khác nhau.*

Bằng những lập luận Toán học, Appel và Haken đưa bài toán tổng quát trên về việc kiểm tra trên 1478 đồ thị cụ thể. Phần "lập luận toán học" dài hơn 1000 trang, và để kiểm tra 1478 đồ thị, cần phải viết chương trình máy tính.

Liệu có thể tin vào một chứng minh như thế hay không? Phần lớn các nhà toán học chưa thừa nhận rằng, bài toán 4 màu đã được giải.

Năm 1996, Robertson, Sanders, Seymour và Thomas tiến một bước dài khi công bố công trình "*Một chứng minh mới của Định lý bốn màu*", trong đó phần lập luận toán học còn khoảng 20 trang, và viết chương trình trên ngôn ngữ C để kiểm tra trên các lớp đồ thị cụ thể, mà số lượng của chúng từ 1.478 được rút xuống còn 633. Có thể tin vào chứng minh của họ được không? Nếu cho rằng 20 trang lập luận toán học có thể kiểm tra kỹ lưỡng thì có gì đảm bảo chương trình máy tính được viết chính xác? Và có gì đảm bảo máy tính chạy trong nhiều giờ (hàng ngàn giờ) để kiểm tra mà không gặp sai sót?

Trên thực tế, người ta thấy rằng, trung bình cứ một dòng lệnh thì người lập trình mắc 1,5 lỗi. Nói chung các lỗi này được lập trình viên kiểm tra và sửa chữa ngay, nhưng trong bản cuối cùng gửi đi sử dụng, trung bình cứ 100 dòng lệnh thì có một lỗi chưa được chữa. Thường thì các lỗi nhỏ không để lại ảnh hưởng gì lớn trong ứng dụng, và lỗi được xem là một phần trong "văn

hoá lập trình"! Tuy nhiên, có một số lỗi trong lập trình dẫn đến hậu quả nghiêm trọng, như vụ nổ của tàu vũ trụ Ariane 5 tốn hàng trăm triệu đô-la. Shamir, một trong ba người sáng lập hệ mã RSA, có lần phát biểu trên New York Times rằng, một lỗi nhỏ về toán trong con chip sử dụng rộng rãi có thể dẫn đến sai sót trong mã hoá, và đặt an ninh thương mại toàn cầu vào tình trạng nguy hiểm.

Như vậy, không thể hoàn toàn tin vào những chứng minh của các nhà toán học, vì họ có thể sai. Cũng không thể hoàn toàn tin vào các chứng minh có sự trợ giúp của máy tính, vì chương trình có thể mắc lỗi, máy tính có thể hoạt động sai!

Làm thế nào để có thể đạt được sự tin cậy cao nhất vào các kết quả toán học?

Vấn đề trên đây dẫn đến sự phát triển mạnh mẽ trong những năm gần đây của lý thuyết chứng minh và kiểm tra tự động. Xa hơn nữa, toán học đang đứng trước một mục tiêu rất "lãng mạn": sẽ đến lúc máy tính có thể thay thế con người trong lao động toán học.

Điều khác biệt giữa việc "kiểm tra" của máy và người là gì? Người phản biện một bài báo gửi đến tạp chí sẽ chấp nhận khâu nào đó của chứng minh trong bài báo là đúng khi thấy nó "hiển nhiên đúng". Mức độ "hiển nhiên" này rất phụ thuộc vào trình độ của người thẩm định. Máy tính không như vậy: "nó" chỉ chấp nhận một kết luận là đúng nếu mọi suy luận logic đều được trình bày, dẫn dắt từ những chân lý đầu tiên, tức là các tiên đề.

Để làm ví dụ, ta thử xem máy tính chứng minh "1+1=2" như thế nào:

$$1+1=1+S(0)=S(1+0)=S(1)=2,$$

ở đây S là ký hiệu "phần tử đi liền sau" trong hệ tiên đề số học của Peano, và các phép tính thực hiện theo các quy định về quan hệ giữa phép cộng và "đi liền sau".

Trong chứng minh trên, không có lập luận nào bị bỏ qua, không có chỗ nào dựa vào trực giác. Tiếc rằng, các định lý khác không đơn giản như "1+1=2", và để quay về đến tận các tiên đề, một định lý đơn giản có thể cần đến hàng chục ngàn dòng. Chính vì thế mà khi nhóm Bourbaki đặt cho mình nhiệm vụ xây dựng lại cơ sở toán học, họ đã nói là không nhằm mục tiêu dẫn đến các chứng minh hình thức, vì nó đòi hỏi dung lượng quá lớn cho mỗi chứng minh.

Lý thuyết *chứng minh hình thức* (formal proof) có mục tiêu nhờ máy tính làm thay con người cái công việc nhọc nhằn là kiểm tra từng khâu chứng minh. Vấn đề là làm sao cho máy tính "hiểu" được ngôn ngữ toán học, từ đó có thể kiểm tra từng khâu chứng minh xem có lập luận nào mâu thuẫn hoặc có lập luận nào đã bị bỏ trống. Chứng minh hình thức là một phần trong lĩnh vực rộng lớn hơn, là tự động hoá quá trình tư duy toán học, *từ phát hiện giả thuyết* đến *xây dựng lý thuyết*.

Thực ra, ý tưởng xây dựng toàn bộ toán học như một ngôn ngữ hình thức đã có từ lâu, đặc biệt là khi Hilbert khởi xướng chủ nghĩa hình thức (formalism) trong toán học, với mục tiêu đưa toán học vượt qua những khủng hoảng trong cơ sở của lý thuyết tập hợp Cantor. Tham vọng của Hilbert là hình thức hoá toàn bộ toán học, xuất phát từ các tiên đề và các quy tắc logic, và khi đó định lý sẽ là một mệnh đề "đúng ngữ pháp". Tuy nhiên, chương trình của Hilbert sụp đổ sau khi xuất hiện công trình nổi tiếng của Goedel: *định lý về tính không đầy đủ* (*incompleteness theorem*,

còn gọi là *định lý bất toàn*). Theo định lý Goedel, một hệ tiên đề phi mâu thuẫn thì sẽ không đầy đủ, tức là luôn tồn tại những mệnh đề không thể chứng minh hay bác bỏ nếu chỉ sử dụng những lập luận nội tại của hệ tiên đề đó. Hệ quả hiển nhiên của nó là không thể hình thức hoá toàn bộ toán học.

Mặc dù không thể đạt được mục tiêu cuối cùng, chủ nghĩa hình thức của Hilbert mang lại diện mạo mới cho toán học. Các lý thuyết toán học trở nên chặt chẽ hơn, và nhiều hệ chứng minh hình thức ra đời. Nổi tiếng nhất là các hệ Coq, HOL Light, Isabelle. Người ta xây dựng được nhiều chứng minh hình thức cho những định lý nổi tiếng của toán học, như Định lý về tính không đầy đủ của Goedel (1986), Luật thuận nghịch toàn phương của Gauss (1990), Định lý cơ bản của giải tích (1996), Định lý cơ bản của đại số (2000), Bài toán bốn màu (2004), Định lý điểm bất động Brouwer (2005), Định lý đường cong Jordan (2005), Định lý thặng dư Cauchy (2007), Định lý số nguyên tố (2008), Giả thuyết Kepler (2014).

Để làm rõ một số vấn đề đặt ra đối với chứng minh hình thức, chúng ta lại sẽ bắt đầu bằng một sự kiện nổi tiếng trong toán học: chứng minh của Thomas Hales về Giả thuyết Kepler, và việc kiểm tra chứng minh đó.

Năm 1998, Thomas Hales làm xôn xao giới toán học khi chứng minh Giải thuyết Kepler tồn tại đã 400 năm.

Năm 1611 Thomas Harriot hỏi Kepler rằng làm cách nào để xếp các viên đạn đại bác hình cầu sao cho đảm bảo xếp được nhiều nhất. Kepler đưa ra giả thuyết: tốt nhất là xếp như các bà bán cam ngoài chợ, tức là bắt đầu xếp một lớp quả cầu trong lưới lục giác, sau đó đặt một lớp tiếp theo ở điểm thấp nhất mà bạn có thể đặt bên trên lớp đầu tiên, và cứ tiếp tục như vậy.

Cách xếp này cho mật độ sử dụng không gian tối ưu, khoảng $\frac{\pi}{\sqrt{18}} \approx 0.7404804898$

Thomas Hales đưa ra một chứng minh rất khó kiểm tra: chứng minh của ông bao gồm 300 trang lập luận toán học và chương trình tính toán khoảng 50.000 dòng lệnh! Năm 1996, ông gửi công trình của mình đến tạp chí Toán học uy tín nhất là *Annals of Mathematics*. Toà soạn phải làm một việc chưa từng có: nhờ 12 người phản biện. Các phản biện tiến hành seminar trong 9 năm trời để kiểm tra, và không tìm thấy một sai sót nào. Tuy nhiên họ thừa nhận là không thể đủ sức kiểm tra toàn bộ, và đề nghị toà soạn đăng vì "tin rằng chứng minh hoàn toàn đúng"! Bài báo của Hales được đăng năm 2005. Đây là trường hợp hiếm hoi, khi một bài báo được đăng mà những người phản biện không dám tin chắc 100% chứng minh trong bài báo là đúng! Tuy nhiên, cũng cần lưu ý rằng, công trình chỉ được đăng sau 9 năm, kể từ khi nó được gửi đến toà soạn.

Thomas Hales không bằng lòng với việc công trình của mình chỉ được "tin là đúng", mà chưa được kiểm tra trọn vẹn. Ông quyết định nhờ máy tính kiểm tra, vì chỉ có máy tính mới có thể tiến hành công việc nhọc nhằn đó.

Để kiểm tra chứng minh của mình về tính đúng đắn của giả thuyết Kepler, Hales xây dựng một đề án lấy tên là Flyspeck, gợi ý từ ba chữ FPK (Formal Proof of Kepler Conjecture). Dự án này kết thúc vào năm 2014, và như vậy, chứng minh của Hales được kiểm tra đầy đủ.

Tuy nhiên, có thể tin được hay không vào "sự kiểm tra" đó? Nói cách khác, cần kiểm tra tính đúng đắn của việc kiểm tra chứng minh bằng máy tính! Khi xây dựng đề án của mình, Hales dùng hệ HOL Light (Lightweight implementation of Higher Order Logic). HOL Light vừa là một hệ tiên đề toán học, vừa là một chương trình máy tính. Phần quan trọng nhất trong HOL Light là những dòng lệnh liên quan các tiên đề và các quy tắc logic, gọi là *hạt nhân* của hệ thống. Mỗi một lỗi trong hạt nhân có thể dẫn đến sai lầm thảm hoạ của cả hệ thống. Chẳng hạn, một tiên đề được viết sai có thể phá vỡ tính phi mâu thuẫn của hệ thống. Rất may, hạt nhân này chỉ chiếm 500 dòng lệnh, và như vậy có thể kiểm tra kỹ lưỡng để tin rằng trong hạt nhân không có lỗi. Tuy nhiên kinh nghiệm sử dụng các hệ chứng minh hình thức cho thấy đối với mỗi hệ, người ta tìm ra lỗi sau khoảng 10-15 năm sử dụng.

Mặt khác, theo định lý Goedel, mỗi hệ chứng minh hình thức không thể "tự mình" kiểm tra được chân lý. Để khắc phục điều đó, người ta dùng hệ hình thức này để kiểm tra tính đúng đắn của hệ hình thức khác. Giả sử mỗi hệ hình thức có xác suất sai lầm p, thì khi kiểm tra một hệ nào đó bằng cách sử dụng n hệ khác, ta có thể đưa xác suất sai lầm đến p^n. Với n đủ lớn thì xác suất này gần bằng 0.

Những lập luận trên đây chỉ ra rằng, những chân lý toán học, nếu không phải là quá phức tạp, thì dù được chứng minh chỉ bởi các nhà toán học, hay có sự trợ giúp của máy tính, đều chỉ là *chân lý tương đối!* Điều này thực ra không dễ chấp nhận đối với nhiều nhà toán học truyền thống.

Một số nhà toán học còn cực đoan đến mức cho rằng, nếu một chân lý toán học được chứng minh mà không cần đến sự trợ giúp của máy tính thì đó chỉ có thể là…chân lý tầm thường! Họ cho rằng chỉ sau 20, 30 năm nữa thôi, tất cả những gì mà toán học ngày nay làm được sẽ trở thành quá dễ với máy tính! Với những nhà toán học theo quan điểm đó, Toán học cuối cùng sẽ có số phận như môn cờ vua, khi mà nhà vô địch thế giới có thể bị đánh bại dễ dàng bởi một máy đánh cờ!

1.4. FAB

Câu chuyện về hành trình chứng minh giả thuyết Kepler là một câu chuyện cá nhân, xuất phát từ nghịch cảnh của nhà toán học Thomas Hales. Song nó cũng là câu chuyện về cách mà con người nhìn ra cơ hội để vượt qua nghịch cảnh. Kể từ sau Flyspeck, Thomas Hales còn muốn tiếp tục phát triển một tầm nhìn rộng hơn kết quả mang tính cá nhân này. Như chia sẻ khi sang thăm Trường Đại học Thăng Long, Hales nói: Lúc đó, tôi đã quyết định biến đây thành mục tiêu cuộc đời. Máy tính biết tính là bởi con người biết tính. Con người biết kiểm tra chứng minh, máy tính cũng có thể sẽ làm được vậy. Tôi muốn có thể xây dựng các phản biện viên ảo hỗ trợ con người, để những tiền lệ như tôi không xảy ra nữa.

Về bản chất, công việc của nhà toán học là: trên cơ sở các quan sát những sự kiện toán học hay những hiện tượng tự nhiên và xã hội, phát hiện và phỏng đoán một số kết quả có thể có, tức là đặt ra các giả thuyết. Bước tiếp theo là chứng minh hoặc bác bỏ giả thuyết đã đặt ra. Để làm việc

này, có những lúc đòi hỏi xây dựng những khái niệm và lý thuyết mới. Một chứng minh chính là một chuỗi suy luận logic để tạo ra con đường nối từ giả thiết đến kết luận, thông qua những định lý đã có sẵn. Xuất phát từ các giả thiết, nhà toán học bắt đầu vận dụng những kiến thức mà mình có được, bổ sung thêm những kiến thức mới cần thiết, và từ đống "hỗn độn" đó, tìm ra con đường đi cần thiết.

Làm thế nào để máy tính thực hiện được công việc nêu trên của nhà toán học? Trước tiên cần cung cấp cho máy tính các kiến thức toán học. Sau đó khi đã có đủ nhiều kiến thức, với khả năng xử lý tín hiệu lớn, máy tính sẽ có lợi thế hơn rất nhiều so với con người trong việc phát hiện ra quy luật, tức là đưa ra các giả thuyết, đồng thời tìm kiếm rất nhanh con đường đi từ giả thiết đến kết luận.

Và với định hướng trên, dự án FAB được thành lập, viết tắt của Formal Abstracts in Mathematics. Đây là dự án có mục tiêu hình thức hoá và kiểm tra hình thức các kết quả toán học, nghĩa là chuyển các chứng minh toán học về một dạng ngôn ngữ mà máy tính có thể hiểu được, và khi máy tính hiểu được thì nó có thể kiểm tra tự động.

Song khác Flyspeck sử dụng ngôn ngữ hình thức HOL-Light, với FAB, Hales chọn một ngôn ngữ khác: Lean. Lean được Microsoft Research phát hành vào 2013, là một công cụ chứng minh hình thức còn đang phát triển nhưng hứa hẹn. Khác tên gọi của nó (Lean = gọn ghẽ), Lean đầy tham vọng khi kết nối chứng minh tự động với chứng minh tương tác, lập trình phổ thông đồng thời chứng minh định lý, cũng đồng thời là một siêu ngôn ngữ khi dùng Lean mở rộng được chính Lean. Xây dựng thư viện Lean còn chủ yếu có sự tham gia của nhóm nghiên cứu đến từ Carnegie Mellon University ở Pittsburgh mà Hales trực tiếp liên hệ. Nói cách khác, dự án FAB được nhận hỗ trợ từ chính những người sinh ra Lean.

Tháng 12 năm 2017, lần đầu tiên giáo sư Thomas Hales đến làm việc với Khoa Toán - Tin và Viện TIMAS của Trường Đại học Thăng Long, bắt đầu với ba buổi giảng về lý thuyết kiểu (Type theory) vào ngày 27, 28 và 29. Trong thời gian này giáo sư Hales đã bày tỏ mong muốn xây dựng một nhóm làm việc ở Trường Đại học Thăng Long để cùng cộng tác làm việc liên quan đến hình thức hóa toán học. Ý thức được đây là một lĩnh vực mới mẻ đầy thử thách song cũng là cơ hội để hợp tác với một giáo sư hàng đầu thế giới trong một dự án có tầm quan trọng với Toán học, một nhóm giảng viên của trường đã quyết định tham gia vào cùng phát triển tầm nhìn của Thomas Hales.

Dự án FAB đã được quỹ Alfred P. Sloan Foundation chính thức phê duyệt tài trợ, mã số G-2018-10067, bắt đầu từ ngày 04 tháng 4 năm 2018 đến hết tháng 3 năm 2021, với sự hợp tác của các nhóm nghiên cứu thuộc ba trường đại học là University of Pittsburgh, Carnegie Mellon University và Trường Đại học Thăng Long, đặt dưới sự lãnh đạo của giáo sư Thomas Hales, nhà toán học hàng đầu trong lĩnh vực chứng minh hình thức. Nhóm FAB của Đại học Thăng Long đã hình thành gồm 8 thành viên:

- 3 giảng viên của Bộ môn Tin (TS. Mai Thúy Nga, NCS. Phạm Phương Thanh, NCS. Nguyễn Đức Thắng)
- 2 giảng viên của Bộ môn Toán (TS. Nguyễn Thị Nhung, TS. Ngô Thị Thanh Nga)
- 3 giảng viên Phòng thí nghiệm Trí tuệ nhân

tạo (TS. Nguyễn Thị Huyền Châu, ThS. Nguyễn Đức Hoàn, ThS. Nguyễn Thị Trà My)

Như vậy, dự án này không chỉ là thành quả hợp tác nghiên cứu giữa 3 trường đại học, còn là sự hợp tác của các phòng và bộ môn của Trường Đại học Thăng Long, với kiến thức và chuyên môn bổ trợ lẫn nhau trong quá trình thực hiện. Trong thời gian 3 năm, hoạt động của nhóm FAB Thăng Long trong khuôn khổ dự án có thể chia thành hai giai đoạn:

Giai đoạn đầu: hướng tới hình thức hóa một số bài toán quan trọng trong Toán học bằng ngôn ngữ Lean. Nhóm FAB đại học Thăng Long đảm nhiệm hình thức hóa các định lý chưa được xử lý bằng Lean trong danh sách 100 định lý toán học kinh điển. Để thực hiện được mục tiêu này, cần nghiên cứu tìm hiểu các kiến thức về lý thuyết kiểu, các kỹ năng lập trình toán học bằng ngôn ngữ Lean, kỹ năng đào phá thư viện Mathlib, và lập kế hoạch phối hợp làm việc hiệu quả với nhóm nghiên cứu ở Mỹ. Trong quá trình vừa hợp tác vừa học tập này, có thể kể đến một loạt các hội thảo và khóa học quan trọng mà các thành viên của nhóm đã được tham gia và được đào tạo trực tiếp như:

- Trường hè Hà Nội về trừu xuất hình thức, " Hanoi summer school on formal abstracts", tổ chức tại Trường Đại học Thăng Long năm 2018 từ ngày 05/06 đến ngày 14/06.

- Tiểu ban "Special session on formal mathematics" (tiểu ban chuyên biệt về Toán học hình thức) tại hội nghị Toán học Việt-Mỹ, "Vietnam-USA joint Mathematical Meeting", diễn ra tại Quy Nhơn, Bình Định tháng 6/2019.

- Hội thảo về Lean và trừu xuất hình thức, "Conference on Lean and Formal Abstract", được tổ chức ở Viện toán học, Viện hàn lâm khoa học Việt Nam từ 17 đến 20/06/2019.

Sau hơn một năm làm việc, nhóm FAB của Trường Đại học Thăng Long đã hình thức hóa hơn 50 định lý trong danh sách phụ trách. Các kết quả liên quan đều trải qua quy trình kiểm tra chéo và nghiệm thu với nhóm nghiên cứu của Thomas Hales. Đồng thời, nhóm FAB Thăng Long và Hales đã cùng trao đổi thảo luận để rút ra một số kinh nghiệm khi làm việc với Lean như sau: Đầu tiên, một số định lý tạm thời chưa thể hình thức hóa được tại thời điểm đó, vì thư viện Mathlib của Lean chưa xây dựng được các cấu trúc Toán cần thiết. Tuy nhiên Lean là một ngôn ngữ mã nguồn mở, cộng đồng sử dụng Lean đã tích cực đóng góp cho thư viện ngôn ngữ này, và thư viện Lean đã mở rộng thêm 25% chỉ sau gần nửa năm. Mặt khác, giai đoạn đầu này cũng chỉ ra rằng đối với các nhà Toán học nói chung, làm việc với Lean một cách trực tiếp là không dễ dàng. Thomas Hales nhận ra nhu cầu cần một cầu nối hay là một giao diện trung gian để hỗ trợ các nhà Toán học làm việc với Lean, dựa trên đó mới có cơ sở xây dựng một thư viện với dữ liệu đủ lớn về hình thức hóa toán học hỗ trợ nhiều hơn cho việc phát triển học máy về sau. Từ đây, ông đưa ra tầm nhìn cho giai đoạn tiếp theo của dự án.

Giai đoạn hai: Trong lần làm việc tại Việt Nam vào tháng 6 năm 2019, giáo sư Hales bắt đầu định hướng nhóm FAB Thăng Long tìm hiểu về các ngôn ngữ tự nhiên có kiểm soát (CNL, viết tắt của Controlled Natural Language). Mục tiêu tiếp theo của Hales là xây dựng một ngôn ngữ CNL cùng một bộ biên dịch tương ứng, dịch từ file dạng Latex sang ngôn ngữ CNL này. Ngôn ngữ và bộ biên dịch này gọi chung là Colada. Colada cùng lúc được xây dựng theo tiêu chí đủ tính hình

thức để có thể xây dựng bộ dịch tự động sang Lean. Như vậy, ý tưởng ở đây là dùng Colada làm trung gian giữa Latex - ngôn ngữ rất phổ thông với các nhà Toán học, với Lean - ngôn ngữ hiệu quả cho việc hình thức hoá.

Trong khung mục tiêu đó, nhóm FAB Thăng Long đảm nhiệm việc kiểm thử Colada bằng cách viết các file dạng Latex theo cú pháp khả chuyển sang CNL, từ đó phát hiện các điểm chưa hợp lý, phản hồi lại cho giáo sư Hales để điều chỉnh Colada. Các file cần chuyển từ Latex sang cú pháp khả chuyển CNL là hơn 1.000 file nằm trong thư mục Number Theory của Planet Math, xem chi tiết tại https://planetmath.org/numbertheory. Cũng trong giai đoạn này, nhóm FAB Thăng Long đã tham dự hội thảo Phương pháp hình thức trong toán học/Cùng nhau với Lean 2020, "Formal Method in Mathematics/ Lean Together 2020", được tổ chức tại Carnegie Mellon University, Pittsburgh, Pennsylvania, Mỹ tháng 1 năm 2020. Qua hội thảo này, các thành viên nghe báo cáo về phiên bản mới nhất của Lean (Lean 4), các kết quả về hình thức hóa toán học không chỉ viết trên Lean mà cả trên Coq (một trợ lý chứng minh hình thức khác). Cũng trong một buổi trình bày chuyên biệt về dự án FAB tại hội thảo, giáo sư Hales đã trình bày ý tưởng táo bạo về CNL, đề ra một giải pháp khả thi hơn cho hình thức hóa toán học. Từ tháng 2 năm 2020 trong bối cảnh thế giới và đặc biệt là nước Mỹ chịu ảnh hưởng nặng nề của đại dịch Covid-19, nhiều nghiên cứu của dự án cũng phần nào bị ảnh hưởng. Mặc dầu vậy nhóm cũng đóng góp cho dự án Colada các kết quả giá trị, đồng thời dự án cũng đã gia hạn thời gian nộp hoàn thiện báo cáo tổng kết đề tài vào tháng 6 năm 2022.

Sau đây trong phần 2, chúng tôi sẽ trình bày tổng quan về công cụ Lean và việc hình thức hoá toán học. Kế đó, sang phần 3, chúng tôi liệt kê những cân nhắc của dự án về một lớp ngôn ngữ trung gian CNL giữa Lean và các nhà toán học. Cuối cùng ở phần 4, chúng tôi đặc tả chi tiết hơn về CNL này và các kết quả thu được.

2. Lean và việc xây dựng cơ sở dữ liệu toán

2.1. Giới thiệu về Lean

Dự án Lean được Leonardo de Moura tại Microsoft Research đề xuất vào năm 2013. Trải qua 8 năm phát triển, hiện tại phiên bản mới nhất là Lean 4, đây là một dự án dài hạn vẫn đang được tiếp tục triển khai. Lean là một phần mềm mã nguồn mở được đưa ra thị trường dưới giấy phép Apache phiên bản 2.0, nhờ đó cho phép người dùng có thể tự do sử dụng và phát triển các thư viện toán học và thư viện mã.

Ngoài cách cài đặt trên máy tính rồi chạy trên phần mềm đã cài đặt, Lean có thể sử dụng trực tiếp trên mạng, khi này một phiên bản tập lệnh Java của Lean, một thư viện tiêu chuẩn các định nghĩa và định lý cũng như một công cụ soạn thảo được tải xuống đường dẫn của bạn và thực thi lệnh ở đó. Đây là cách nhanh chóng và thuận tiện để thử nghiệm với hệ thống này. Nhưng phiên bản gốc trên máy sẽ nhanh hơn và linh hoạt hơn so với phiên bản mạng, các chế độ đặc biệt trong Visual Studio Code (viết tắt là VS code) và Emacs cung cấp hỗ trợ mạnh mẽ cho việc viết và gỡ lỗi các chứng minh, đặc biệt phù hợp cho làm việc chuyên sâu với Lean.

Lean cung cấp các API (viết tắt của Application Programming Interface), phương thức trung gian kết nối các thư viện và các ứng dụng khác nhau, để tích hợp vào các công cụ chứng minh toán học khác, ví dụ như Coq [17] và Matita [1]. Lean có

thể được sử dụng như một bộ kiểm tra các chứng minh, các định nghĩa và các định lý được kiểm tra song song sử dụng phần lõi trên máy chủ. Khi được sử dụng như một trợ lý chứng minh (proof assistant), Lean cung cấp bộ soạn thảo hiệu quả có thể xử lý các logic bậc cao. Lean cũng cho phép người dùng cung cấp các định nghĩa và các định lý bằng các kiểu khai báo tương tự các trợ lý chứng minh khác như Isabell [14], HOL-Light [2] và Coq.

Đặc điểm nổi bật của Lean là được tạo ra từ một hạt nhân nhỏ gọn nhưng hiệu năng cao, có bộ dịch chạy rất nhanh. Lean là ngôn ngữ lập trình hàm, lại có kiểu phụ thuộc (dependent type) nên đây là một ngôn ngữ rất mạnh để hình thức hóa các cấu trúc toán học một cách chính xác như nó vốn có.

2.2. Các kết quả về Lean của nhóm Formal Abstract Thăng Long

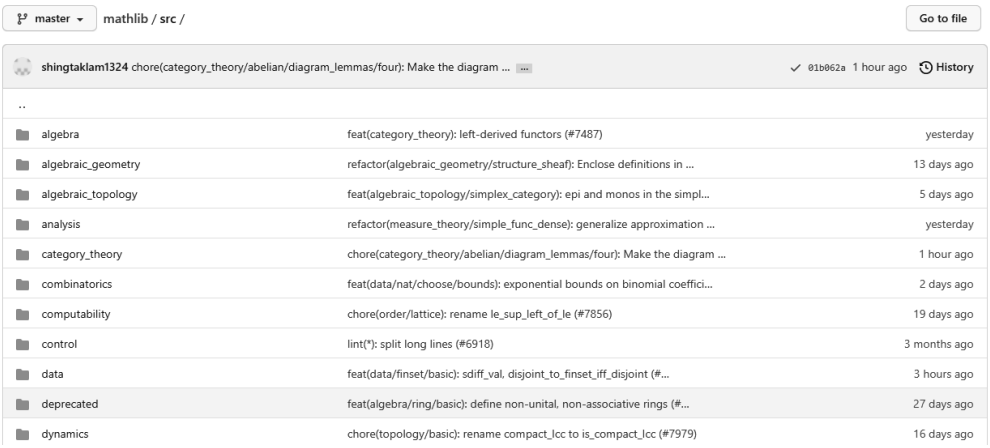
- **Tìm hiểu về thư viện Mathlib**

Để xây dựng một chứng minh toán học hình thức, người ta phải xây dựng các định nghĩa, các tiên đề, các kết quả dạng bổ đề, định lý, hệ quả cùng với chứng minh hình thức của chúng. Các kết quả này, nếu được định nghĩa đủ tốt, có thể được chấp nhận thêm vào thư viện chung của Lean - thư viện mathlib.

Mathlib là một cơ sở dữ liệu về toán học hình thức được cài đặt trong trợ lý chứng minh Lean. Mathlib được chia ra theo các lĩnh vực trong toán học như Đại số đại cương (General Algebra), Đại số tuyến tính (Linear Algebra), Tô pô (Topology), Giải tích (Analysis), Hình học (Geometry), v.v. Mỗi chủ đề toán học được đưa vào một thư mục, các chủ đề lớn lại chia tiếp thành các chủ đề nhỏ theo phân cấp cây, Hình 1 là hình ảnh một phần cây thư mục trong mã nguồn của mathlib.

Thư viện mathlib được công khai trên github [18]. Mathlib được nhóm phát triển ở CMU kiểm duyệt, và vẫn tiếp tục được gửi về các đóng góp nhờ cộng đồng mã mở Lean. Mathlib được cộng đồng các lập trình viên phát triển rất nhanh, vào tháng 5 năm 2020 thư viện Mathlib đã có 170.000 dòng code, tăng 25% so với 5 tháng trước, với 42.000 các khai báo.

Để đọc hiểu được mã nguồn này đòi hỏi kiến thức về ngôn ngữ lập trình Lean lẫn hiểu biết sâu sắc các khái niệm về toán học, điều này yêu cầu sự kết hợp chặt chẽ giữa các giảng viên Toán và Tin để bổ trợ kiến thức trong quá trình tìm hiểu và khai thác thư viện Mathlib.



Hình 1. Cây thư mục mã nguồn trong thư viện Mathlib

- **Hình thức hóa các định lý trong 100 định lý phổ biến**

100 định lý phổ biến nhất trong toán học [19] liên quan đến nhiều chủ đề toán học khác nhau như đại số, giải tích, lý thuyết tập hợp, lý thuyết xác suất, v.v..

Quy trình hình thức hóa các định lý này như sau: Phân tích nội dung định lý, phân loại và liên kết các khái niệm được dùng, từ đó tìm kiếm trong thư viện mathlib đối tượng mà ta cần dùng. Nếu không có, cần tự xây dựng đối tượng đó trên cơ sở những kết quả trong mathlib.

Nhóm FAB của Đại học Thăng Long đã hình thức hóa được hơn 50 định lý trong số 100 định lý phổ biến này, quá trình viết mã tạo ra file .lean và đồng thời tạo tài liệu trực tuyến tương ứng cho mã là file .rst .

Ví dụ về mã nguồn xây dựng cho định lý số hoàn hảo (Perfect Number Theorem) như đoạn code trong hình dưới. Trong dòng 1 chúng ta khai báo việc sử dụng thư viện số nguyên tố trong Mathlib, dòng 2 định nghĩa hàm kiểm tra một số có phải là số chẵn hay không. Từ dòng 7 đến dòng 11 là xây dựng hàm trả về các ước số của một số tự nhiên N. Từ dòng 14 đến dòng 16 là hàm tính tổng các phần tử trong một danh sách. Dòng 19 định nghĩa hàm kiểm tra một số có phải là số hoàn hảo hay không. Từ dòng 22 đến dòng 27 là xây dựng định lý số hoàn hảo dựa trên việc gọi các hàm đã xây dựng trước đó.

```
1  import data.nat.prime
2  definition isEven(n:ℕ ): Prop := 2 ∣ n
3
4  open nat
5
6  --create a list of divisor
7  def list_divisor : ℕ → ℕ  → list ℕ
8  |x 0 := []
9  |x 1 := [1]
10 |x (succ n) := if (succ n)∣x  then succ n:: (list_divisor x  n)
11                               else (list_divisor x n)
12
13 --calcul the sum of list's elements
14 def sum_List:list ℕ  →  ℕ
15 |[]:=0
16 |(a::l):= a + sum_List l
17
18 --check perfect number
19 definition isPerfectNumber(p:ℕ):bool := sum_List(list_divisor p (p-1))=p
20
21 --perfect number theorem
22 theorem Perfect_Number_Theorem: ∀m: ℕ, (isEven(m)) ∧  isPerfectNumber(m)
23     → ∃ p:ℕ, (nat.prime p) ∧ (m = 2^(p-1)* (2^p-1)) ∧  nat.prime (2^p - 1)
24     :=
25 begin
26   admit,
27 end
```

Hình 2. Mã nguồn của định lý số hoàn hảo

3. Từ Lean đến CNL

3.1. Tại sao cần chuyển hướng?

Như đã nói ở phần 1, các thế mạnh của Lean cùng sự phát triển của cộng đồng mã mở Lean đã dẫn đến dự án FAB lựa chọn dùng ngôn ngữ này cho hình thức hoá cơ sở dữ liệu toán học. Nhưng bên cạnh đó, vẫn còn nhiều khó khăn.

Ngôn ngữ hình thức cho chứng minh tự động đã phát triển nhanh chóng hơn 20 năm qua, các kết quả của nó đã được chấp nhận trong các xuất bản ở các tạp chí lớn về toán (sự hình thức hoá định lý Feit-Thompson và giả thuyết Kepler), đã đóng góp vào thành công của các dự án kiểm định công nghiệp (dự án trình biên dịch CompCert và vi nhân SeL4), đồng thời trở thành chủ đề được quan tâm thường xuyên trên các diễn đàn của cộng đồng làm toán phổ thông (mathoverflow, reddit). Tuy nhiên, sẽ vẫn còn rất lâu cho đến khi ngôn ngữ này đạt mức độ phổ biến đại chúng.

Lý do đầu tiên: Các ngôn ngữ hình thức rất khó để thành thạo. Tựa như các đoạn mã của một chương trình máy tính được phép đặt ở nhiều file miễn là có cách tham chiếu, không có ràng buộc hiển nhiên nào về phạm vi của các thông tin ngữ cảnh trong một hệ thống trợ lý chứng minh kiểu Lean. Do đó, các thông tin ngữ cảnh này cũng có thể trải rộng qua nhiều file và không dễ theo dõi. Tuy nhiên điều này cũng dần được cải thiện nhờ giao diện mở ra cửa sổ tìm kiếm chéo trong các môi trường phát triển Lean.

Thứ hai, điểm mạnh của Lean và cũng là điểm làm nhiều nhà Toán học mới tìm hiểu đau đầu là: Trong Lean, cái hiển nhiên chẳng còn là hiển nhiên nữa. Kevin Buzzard phát bực khi cố gắng để Lean chấp nhận những sự thật kiểu như 1 khác 2. Còn Thomas Hales thì đùa rằng nếu không đang là sinh viên Stanford hay Carnegie Mellon University định viết khoá luận về Lean, hoặc sinh viên Imperial College London được Kevin Buzzard hướng dẫnthì dùng Lean sẽ là một thách thức.

Song, rõ ràng dự án của Hales sẽ cần nhiều hơn một đội ngũ các chuyên gia, và chỉ thành công khi thu hút một cộng đồng những nhà toán học cùng đóng góp.

Vậy cần phải làm thế nào?

Hales nghĩ tới một cầu nối giữa ngôn ngữ toán học tự nhiên và ngôn ngữ hình thức, hay như cách ông mô tả "một dạng chất dẫn để đến với Lean".

Đầu tiên hãy xem xét 3 phiên bản biểu diễn của cùng một phát biểu toán học dưới đây:

```
[VERSION 1]
(!V. packing V
           ==> (?c. !r. &1 <= r
               ==> &(CARD(V INTER ball(vec 0,r))) <=
               pi * r pow 3 / sqrt(&18) + c * r pow 2))
```

Hình 3. Version HOL Light

Phiên bản 1 là phát biểu giả thuyết Kepler biết bởi trình trợ lý chứng minh HOL Light. Dễ thấy, phát biểu này hoàn toàn khó hiểu với các nhà toán học bình thường. Nó sử dụng các ký tự ascii để biểu diễn khái niệm toán học theo một cách hơi kỳ dị: ví dụ như ký tự ! để biểu diễn lượng từ mọi, ? cho lượng từ tồn tại, ...

[VERSION 2]

$$\begin{aligned}&(\forall\, V.\ \text{packing}\ V\\ &\Rightarrow(\exists\, c.\ \forall\, r.\ \&1 \le r\\ &\Rightarrow \&(\text{CARD}(V\ \cap\ \text{ball}(\text{vec}\ 0, r))) \le\\ &\pi * r^3/\sqrt{(\&18)} + c * r^2))\end{aligned}$$

Hình 4. Version HOL Light cải tiến

Phiên bản 2 thay các ký tự trên bằng những ký tự gần gũi toán học hơn, tuy nhiên vẫn là một công thức thiếu tự nhiên. Một nhà toán học hiển nhiên biết số nguyên 18 cũng là một số thực mà chẳng cần phải ép kiểu tường minh.

[VERSION 3]

For every packing V, there exists a real number c such that for all real numbers $r \geq 1$, the number of points of V in the open ball $\mathbf{B}(0, r)$ is at most

$$\frac{\pi\ * r^3}{\sqrt{18}} + c\ * r^2.$$

Hình 5. Version CNL

Sau chót, phiên bản 3 tỏ ra dễ đọc nhất. Nó không lạm dụng sự biểu tượng hoá, mặt khác vẫn tận dụng được các ký pháp công thức gọn gàng và sáng rõ của Toán học. Một đặc điểm quan trọng là tuy dễ hiểu hơn nhiều, phiên bản này không hề kém tính hình thức so với phiên bản 1. Đây chính là minh hoạ cho chất dẫn mà Hales muốn hướng tới - một ngôn ngữ hình thức mà có dáng vẻ tự nhiên. Dự án Colada ra đời, là một sự kết hợp giữa các đặc trưng ngôn ngữ của Forthel, Latex, và Lean, với mục đích là tạo ra một CNL - Controlled Natural Language - ngôn ngữ tự nhiên có kiểm soát, kết nối được Lean với các nhà toán học.

Hãy lần lượt xem xét các cơ sở để hiện thực hoá ý tưởng này:

3.2. Ngôn ngữ tự nhiên trong Toán học

Cơ sở đầu tiên là dựa trên quan sát cách các nhà toán học dùng ngôn ngữ tự nhiên.

Ngôn ngữ tự nhiên vốn không phải một ngôn ngữ có tính hình thức mạnh mẽ. Tuy nhiên, ngôn ngữ tự nhiên của toán học là ngôn ngữ tự nhiên dùng cho một mục đích cụ thể, ở một ngành đặc trưng bởi sự trừu tượng chặt chẽ. Ngôn ngữ này, do đó, mang sẵn một số thuận lợi nhất định đối với việc hình thức hoá.

Đầu tiên thì nó chỉ là một tập con của ngôn ngữ tự nhiên. Quan trọng hơn, nó có độ bất biến ngữ nghĩa lớn. Một khó khăn trong chuyển đổi ngôn ngữ đời sống sang ngôn ngữ hình thức nằm ở xác suất lớn là mỗi thay đổi về từ vựng hay thứ tự tổ hợp thì lại kéo theo một/nhiều thay đổi về ngữ nghĩa, sự chênh lệch đôi khi vi tế mà lại quan trọng, ví dụ như các sắc thái cảm xúc của phát biểu. Trong khi đó, tuy về hình thức, ngôn ngữ Toán đang sử dụng lại ngôn ngữ tự nhiên, về mặt nội dung, ta lại chỉ cần quan tâm đến khía cạnh Toán học. Với Toán học, có những từ ngữ là quan trọng, có những từ ngữ chỉ là "chèn vào cho đủ", có những khác biệt cú pháp biến đổi nội dung, và có những khác biệt coi như tương đương. Hệ quả là, so với ngôn ngữ tự nhiên, ngôn ngữ Toán ổn định ngữ nghĩa hơn. Các biểu diễn khác nhau của cùng một ý tưởng có thể chỉ xuất phát từ thói quen cá nhân, hơn là một chủ ý Toán học. Điều đó cũng nghĩa là tồn tại khả năng quy chuẩn chúng về một dạng chung mà không gây mất mát thông tin. Các nhà toán học đa phần đều hiểu được các sự khác nhau này không mấy khó khăn, điều này nói lên trong cộng đồng làm Toán đã chia sẻ sẵn một vài quy tắc thống nhất cách hiểu, cả khi chưa phát biểu tường minh. Nếu dạng chuẩn hoá phản ánh được các quy tắc ngầm này, có thể tin rằng nó sẽ tự động hiểu được bởi đa số các nhà toán học.

Từ những điều trên, có thể nghĩ tới thiết kế một tập con chuẩn hoá cho toán học, có từ vựng và cú pháp chặt chẽ hơn ngôn ngữ hiện tại, nhưng vẫn bảo lưu các ưu điểm trôi chảy và dễ đọc của ngôn ngữ tự nhiên.

3.3. Tại sao lại là CNL?

Cũng không phải chỉ đến bây giờ người ta mới nghĩ tới ý tưởng kết hợp ngôn ngữ tự nhiên và ngôn ngữ hình thức. Năm 1970, Victor Krushskov - nhà Toán học Xô Viết cha đẻ của ngành công nghệ thông tin đồng thời là nhà tiên phong trong ngành điều khiển học của nước Nga, cũng đã đề xuất ý tưởng kết hợp ngôn ngữ tự nhiên và hình thức, khi ấy phục vụ một chương trình chứng minh tự động của Nga tên là Evidence Algorithm.

Tại sao một nhà toán học chọn phân rã chuỗi chứng minh của ông ta thành một lược đồ bao gồm các bổ đề đặt chính xác ở các mốc này mà không phải ở mốc kia? Điều ấy không luôn là một sắp xếp ngẫu nhiên, cũng không phải một quy luật tổng quát, mà phản ánh linh tính cá nhân nào đó về cách mỗi người giải quyết vấn đề trong thế giới thực. Nhưng ngôn ngữ hình thức bất lực trong mô tả những sự thật kiểu ấy, các chủ ý mang tính biểu tượng sẽ biến mất hoàn toàn trong chuỗi logic sinh từ máy tính. Một ngôn ngữ kết hợp giữa hình thức và tự nhiên có thể lưu giữ các thông tin không tường minh song có ích như thế.

Ý tưởng này nằm trong hiếm hoi những điều Tây Âu và Đông Âu đồng ý bất chấp Chiến tranh lạnh. Các phiên bản ngôn ngữ lai cho tiếng Anh lần lượt ra đời, trong đó có ForTheL - Formal Theory Language, ngôn ngữ lõi cho bộ chứng minh SAD – System for Automated Deduction.

SAD được đề xuất vào 1980, nhưng 2008 mới được cài đặt trong luận án tốt nghiệp của một nhà toán học trẻ người Uckraine là Andrei Paskevich. Từ bước ngoặt này, vào 2017, SAD lại được mở rộng để kết hợp với ngôn ngữ chứng minh Isabelle, nằm trong dự án Naproche-Isabelle tại đại học Bonn của giáo sư Peter Koepke.

Và Koepke, 2 năm sau, là người Hales đã gặp để bàn về CNL.

Bằng cách đi từ Controlled Natural Language

(CNL), tạm dịch là ngôn ngữ tự nhiên có kiểm soát, Hales muốn hướng đến thiết kế một ngôn ngữ lập trình nhân tạo có thể dùng cho giao tiếp toán học, được thiết kế có chủ ý với cú pháp và ngữ nghĩa chính xác mà máy tính có thể đọc được, nhưng lại dựa trên một ngôn ngữ tự nhiên duy nhất (tiếng Anh), mà có thể dễ dàng sử dụng một cách rộng rãi, ít nhất là bởi những người làm toán mà sử dụng thông thạo ngôn ngữ tự nhiên.

Thomas Hales đã quyết định lựa chọn CNL với những lý do sau đây:

- Tiếp cận ngôn ngữ Toán học từ góc độ ngôn ngữ học truyền thống là một lĩnh vực gần như bị bỏ ngỏ nên kết quả còn rất sơ khai. Tuy dựa trên ngôn ngữ tự nhiên, các nhà ngôn ngữ học gần như phải tổng hợp và xây dựng lại bộ từ vựng và cú pháp đặc trưng cho Toán.
- Tiếp cận kiểu học máy cũng chưa cho thấy kết quả khả quan. Trí tuệ nhân tạo và các phương pháp xử lý ngôn ngữ tự nhiên vẫn còn cần nhiều thời gian để có thể đọc hiểu các phát biểu toán học theo ngữ nghĩa như nó được viết hiện nay. Ở đây, ta gặp kịch bản kiểu con rắn cắn đuôi. Chúng ta cần học máy để hình thức hoá tự động nhiều xuất bản toán học. Mặt khác, lại cần sẵn nhiều xuất bản đã được hình thức hoá để có tập dữ liệu đủ lớn cho học máy thành công.
- Như phân tích trong 3.1, các nhà toán học vẫn còn chưa sử dụng rộng rãi các trợ lý chứng minh, bởi mất nhiều thời gian để thành thạo các ngôn ngữ này. Đến đây chúng ta thấy, tồn tại một khoảng cách lớn giữa ngôn ngữ toán và ngôn ngữ hình thức.
- CNL lại cho phép rút ngắn khoảng cách này. Sẽ dễ dàng hơn nếu chúng ta xuất phát từ ngôn ngữ Toán học để xây dựng một điểm giữa nối với Lean, thay vì làm ngược lại. Thứ nhất, điều này đặt ưu tiên là người dùng – các nhà toán học tương lai sẽ sử dụng trợ lý chứng minh. Ngôn ngữ CNL cần thoả mãn các yêu cầu thực tiễn từ những người dùng này trước, sau đó mới tinh chỉnh cho phù hợp với nền tảng toán của Lean. Thứ hai, tiếp cận này cũng giảm bớt độ phức tạp thuật toán, bởi bắt đầu từ phía dễ dàng tổng quát hoá và kiểm thử hơn.

3.4. Tại sao là Forthel?

Trong số các ngôn ngữ CNL, Forthel tỏ ra là ứng viên hứa hẹn là cầu nối giữa các nhà toán học và Lean, đầu tiên bởi nó dựa trên các cấu trúc ngôn ngữ khá đơn giản song lại vẫn cho ra kết quả dễ hiểu. Các nhà toán học không cần một đào tạo đặc biệt nào đều có thể đọc hiểu các văn bản Forthel. Forthel tiếng Nga có nghĩa là "mẹo", và trong Forthel có rất nhiều mẹo hữu ích. Ví dụ như người dùng có thể tự định nghĩa thêm các "thành ngữ", có thể thêm các từ đồng nghĩa, tự động bỏ qua các mạo từ, và các từ gọi là "chèn thêm" (filler) cũng sẽ được nhảy qua không dịch, v.v. Những mẹo (trick) này giúp Forthel vừa linh hoạt và trong sáng như ngôn ngữ tự nhiên, vừa không quá đa nghĩa như ngôn ngữ tự nhiên.

Kế đó, cú pháp của Forthel tuy không phi ngữ cảnh, nhưng gần như phi ngữ cảnh. Các luật sinh của Forthel, do đó, được trình bày theo cú pháp BNF (Backus – Naur Form) của một ngôn ngữ phi ngữ cảnh. Trong khoa học máy tính, ngôn ngữ phi ngữ cảnh biểu diễn bởi BNF đã được nghiên cứu phát triển nhiều lý thuyết xây dựng chương trình dịch. Cú pháp của Forthel cũng đồng thời đủ phổ quát và cho phép mở rộng tập luật sinh khi cần thiết, để đáp ứng các yêu cầu dịch sang ngôn ngữ mới cũng như là các yêu cầu làm giàu

nội dung biểu diễn toán học về sau. Cuối cùng và quan trọng nhất, nếu ForTheL tuy chỉ xử lý logic bậc nhất, vẫn có thể là đầu vào cho một ngôn ngữ xử lý logic bậc cao như Isabelle, hoàn toàn có cơ sở để tin rằng nó có thể làm điều tương tự với một ngôn ngữ bậc cao khác là Lean.

Vì những lý do trên, Forthel phù hợp làm khuôn mẫu để mở rộng thành một ngôn ngữ CNL riêng có khả năng đáp ứng các quy tắc của Giải tích Quy nạp (Calculus of Indutive Constructions) – vốn là nền tảng toán học của Lean.

4. Colada – một CNL cầu nối giữa Latex và Lean

4.1. Giới thiệu chung

Colada là một hệ thống kiểm soát ngôn ngữ tự nhiên CNL của toán học cho phép biên dịch sang trợ lý chứng minh Lean. Tên gọi này được viết tắt từ **CO**ntrolled **LA**nguage **DA**ta, và được truyền cảm hứng từ ly cocktail Pina Colada, theo như chia sẻ của giáo sư Hales. Thiết kế của ngôn ngữ này phát triển từ những CNL đã có sẵn, đặc biệt như là Forthel-Naproche-SAD (gọi tắt là Forthel). Những chương trình được viết bằng Colada sẽ được soạn thảo bằng LATEX và cho kết quả đầu ra là một tài liệu đã được kiểm tra kiểu bằng Lean. Hệ thống hiện nay vẫn đang trong giai đoạn hoàn thiện, đặc biệt là phần biên dịch sang Lean.

Mục tiêu muốn hướng đến là một ngôn ngữ cho phép thu hẹp khoảng cách giữa công việc hình thức hoá và cách viết văn bản toán học hiện tại. Việc viết tài liệu bằng Colada sẽ tương tự như soạn thảo trên Latex. Việc dịch tài liệu bằng ngôn ngữ này sẽ tương tự như dịch bộ kí tự toán học trong Latex, có thể sẽ có thêm một số quy ước đơn giản. Và Giải tích Quy nạp (Calculus of Inductive Construction - CiC) là cơ sở logic và toán học của hệ thống giống như trong Lean. Do đó, các chương trình được viết bằng Colada sẽ được biên dịch sang CiC.

Thiết kế chung. Bước đầu, ngôn ngữ Colada có thể được xem như sự kết hợp của các cú pháp của Latex, Lý thuyết kiểu và Forthel, được kết nối với nhau bằng một hệ thống ngữ nghĩa chung là CiC. Từ góc nhìn đơn giản hơn, chương trình viết bằng Colada cũng chính là các tài liệu soạn thảo bằng Latex với hệ thống ngữ nghĩa xác định trước. Ngôn ngữ này được thiết kế như một ngôn ngữ mà trong đó các định nghĩa và định lý toán học được diễn đạt một cách chính xác mà dễ đọc. Dễ đọc ở đây được hiểu là những người làm toán thuần tuý, không nhất thiết phải biết sử dụng các ngôn ngữ lập trình tin học, có thể đọc hiểu và sử dụng được. Người đọc sẽ được cảnh báo khi nội dung được viết chính xác theo định dạng mà máy tính đọc được nhưng không đảm bảo tính chính xác về logic toán học. Ngôn ngữ này còn bao gồm các cú pháp cho chứng minh toán học, người đọc sẽ được cảnh báo khi bài chứng minh của mình được viết bằng Colada không thể được hình thức hoá trong một trợ lý chứng minh như Lean.

Cơ sở toán học. Ngữ nghĩa của Colada dựa chủ yếu trên lý thuyết kiểu hơn là lý thuyết tập hợp. Đây là 2 cách khác nhau để xây dựng phần lớn toán học thành một hệ thống toàn diện. Lý thuyết kiểu khá giống lý thuyết tập hợp nhưng có một số khác biệt quan trọng. Ta có biểu đồ Venn là cách biểu diễn phổ biến cho tập hợp, nó mô tả tập hợp là một sự tụ tập của các phần tử và các tập hợp có thể giao nhau bất kì. Một kiểu cũng có thể được mô tả như là sự tụ tập của các phần tử, tuy nhiên các kiểu luôn luôn rời nhau (đây chính là sự khác biệt cơ bản và quan trọng so với tập hợp). Khó khăn chính khi cố gắng sử dụng lý

thuyết tập hợp làm hệ thống cơ sở trên máy tính là sự giao nhau của các tập hợp. Các đối tượng sẽ bị định nghĩa chồng lẫn lên nhau nếu theo lý thuyết tập hợp, gây cản trở cho việc xử lý thông tin và tính toán. Trong khi đó lý thuyết kiểu với tính chất rời nhau của các kiểu lại cho phép tổ chức dữ liệu một cách mạch lạc, rõ ràng và có trật tự. Tuy nhiên, đây cũng chính là thách thức cho người thiết kế vì ta vẫn cần một chút giao nhau của các kiểu. Ví dụ ta không muốn số tự nhiên 0 và số thực 0 là hai đối tượng hoàn toàn tách biệt nhau như trong lý thuyết kiểu. Tóm lại, bằng việc sử dụng lý thuyết kiểu, ta sẽ có các đối tượng được xác định bằng một kiểu duy nhất và bất kì đối tượng nào cũng đều phải có kiểu, ngay cả đối tượng kiểu!

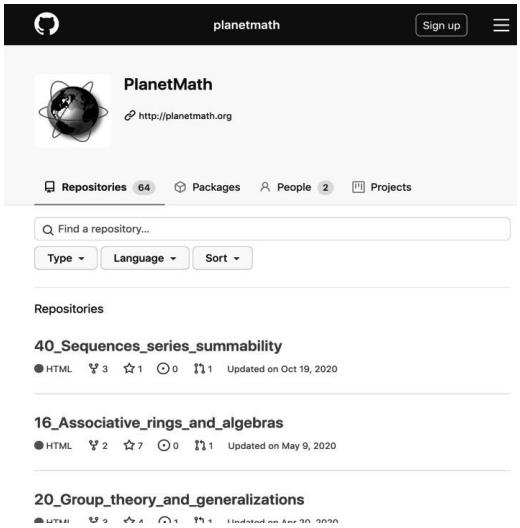
Cơ sở phần mềm. Phương ngữ được dùng có thể được xem như là sự kết hợp của ba loại cú pháp khác nhau: cú pháp của Forthel, cú pháp của Latex và cú pháp của Lean. Cấu trúc từ vựng của Colada được định nghĩa trong sedlex, một công cụ tạo từ vựng cho Ocaml. Và ở đây menhir được sử dụng làm công cụ phân tích cú pháp dựa trên Ocaml cho trình phân tích LR(1) (trình phân tích cú pháp kiểu LR có nhìn trước một ký tự [12]). Mặc dù phương ngữ của Colada không phải LR(1), điều này ngăn cản menhir tự động tạo trình phân tích cú pháp, thì phần mềm vẫn sẽ kiểm tra được lỗi ngữ pháp. Bộ ngữ pháp được xây dựng bao gồm 350 danh ngữ và khoảng 700 quy tắc, cùng với 150 từ khoá phụ thuộc ngữ cảnh, và chưa kể đến những ngữ pháp do người dùng tự định nghĩa mở rộng khi viết chương trình. Sự phức tạp này là cần thiết để có thể diễn đạt được các phát biểu cũng như công thức toán học phổ biến, cùng cú pháp của lý thuyết kiểu và CiC. Colada giữ hầu hết tính năng của Forthel, chẳng hạn như cách xử lý từ đồng nghĩa, cụm danh từ, động từ và tính từ, cùng cơ chế mở rộng ngữ pháp. Ngoài ra có thêm tính năng bổ sung như phân tích mức độ ưu tiên toán tử (với mức độ ưu tiên do người dùng chỉ định), xác định phạm vi, cú pháp macro Latex, và lý thuyết kiểu phụ thuộc bao gồm kiểu quy nạp, cấu trúc và hàm lambda. Trình phân tích cú pháp của Colada được triển khai trong Ocaml, xây dựng nên thư viện tổ hợp phân tích cú pháp mà John Harrison đã dùng để phân tích cú pháp cho HOL Light. Tuy nhiên trình phân tích cú pháp này vẫn đang trong giai đoạn hoàn thiện. Ví dụ, nó vẫn không có khả năng chuyển đổi cây cú pháp của một văn bản đã phân tích cú pháp sang dạng biểu diễn mà Lean có thể đọc được (đây là một mục tiêu lớn của dự án). Colada luôn làm cho ngữ pháp trở nên phức tạp bằng việc sử dụng kết quả dịch phù hợp dài nhất từ bộ quy tắc. Để tránh điều này, người dùng có thể chèn thêm dấu ngoặc đơn và nếu một kí hiệu được gán cho nhiều nghĩa thì khi đó nghĩa được sử dụng gần nhất sẽ được áp dụng.

4.2. Công việc của nhóm FAB Thăng Long

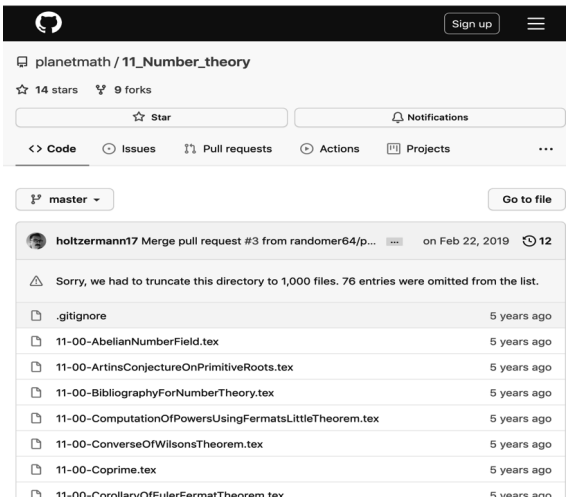
Nằm trong dự án Formal Abstracts của giáo sư Thomas Hale giai đoạn 2, nhóm chúng tôi bắt đầu công việc bằng việc tìm hiểu về CNL, lý thuyết kiểu và Forthel, cùng với ngôn ngữ lập trình Ocaml. Từ đó chúng tôi học cách sử dụng Colada và đưa ra những phản hồi về hệ thống tới đội ngũ thiết kế nhằm mục tiêu hoàn thiện và phát triển phần mềm.

Nhiệm vụ đầu tiên là viết lại phần nội dung Lý thuyết số (Number Theory) trong thư viện Planet Math sang Colada với khoảng hơn 1.000 file. Thư viện toán học PlanetMath là một cộng đồng trực tuyến nhằm mục đích giúp kiến thức toán học

dễ tiếp cận hơn. Nội dung của PlanetMath giống như một bách khoa toàn thư về toán học với các mục được phân chia và sắp xếp theo các lĩnh vực cụ thể của toán học. Bài viết được các thành viên trong nhóm đóng góp và phê duyệt . Sản phẩm cuối cùng sẽ là một thư viện bao gồm các file định dạng Latex được viết theo ngôn ngữ Colada và đã được phân tích cú pháp thành công. Nội dung cần dịch bao gồm các kí hiệu, định nghĩa, và định lý. Nhận xét và chứng minh sẽ chưa được dịch tại giai đoạn này. Tuy nhiên, nhận xét và chứng minh vẫn được viết vào trong file và để giữa các mục 'remark' để trình biên dịch bỏ qua phần này khi biên dịch file. Mục tiêu chính là có được những tài liệu được viết bằng Colada mà có thể đọc được như file gốc trong PlanetMath. Trừ những định nghĩa mở rộng, file Colada sẽ có độ dài tương tự như file gốc. Bất kì vấn đề nào liên quan đến khả năng dịch nội dung của Colada hay những định nghĩa mở rộng và độ dài của file dịch đều cần được thảo luận và tìm ra giải pháp tốt nhất.

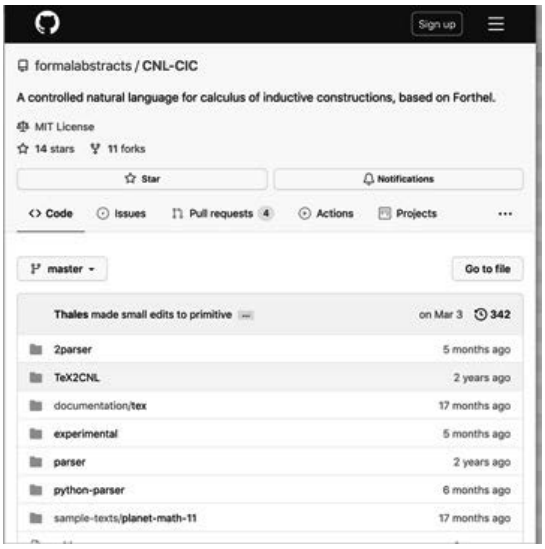


Hình 6. Trang web của PlanetMath



Hình 7. Phần Lý thuyết số cần phải dịch sang Colada

Công cụ làm việc. Trang web github của dự án là formalabstracts/CNL-CIC. Đây là nơi lưu trữ toàn bộ các thư viện của trình biên dịch cũng như thư viện các quy tắc ngữ pháp của Colada và các tài liệu hướng dẫn (Hình 8). Để quản lý các file kết quả của PlanetMath, chúng tôi chia sẻ một thư mục chung trong thư viện của Colada. Bằng cách sắp xếp như vậy, ta có thể kết nối văn bản soạn thảo với bộ dịch và thư viện ngữ pháp một cách dễ dàng.



Hình 8. Trang github của dự án CNL-CIC

Cách thức làm việc. Để tránh việc dịch trùng lặp các nội dung của lý thuyết số, chúng tôi chia nhau dịch các file khác nhau. Soạn thảo văn bản Latex được thực hiện trên ứng dụng Visual Code và sau đó thực hiện biên dịch sang Colada. Sau khi đã được phân tích ngữ pháp bằng Colada và đảm bảo file biên dịch tốt, chúng tôi sẽ tải kết quả lên thư mục chung trên github và yêu cầu xét duyệt để được trộn với thư viện có sẵn.

Kết quả. Hơn 30 file nội dung của Lý thuyết số đã được dịch thành công và trong đó có 8 file đã được trộn cùng thư viện chính thức của Colada. Các phản hồi được đưa ra liên tục và kịp thời để hoàn thiện phần mềm.

```
95  +
96  + \begin{theorem}
97  +       Assume that $a, n$ are positive integers and $\phi(n)$ is Euler's function.
     $gcd \ a\ n =1$  then $a^{\phi (n)} \equiv 1 \mod n$.
98  + \end{theorem}
99  +
100 + \begin{remark}
101 + \begin{proof}
102 + We will make use of Lagrange's Theorem: Let $G$ be a finite group and let $H$ be a
     subgroup of $G$. Then the order of $H$ divides the order of $G$.
103 +
104 + Let $G=(\Ints/n\Ints)^\times$ and let $H$ be the multiplicative subgroup of $G$
     generated by $a$ (so $H=\{ 1, a ,a^2,\ldots \}$). The fact that $\gcd(a,n)=1$ ensur
     that $a\in G$. Notice that the order of $H$, $h=|H|$ is also the order of $a$, i.e.
     the smallest natural number $n>1$ such that $a^n$ is the identity in $G$, i.e.
     $a^h\equiv 1 \mod n$. Also, recall that the order of $G=(\Ints/n\Ints)^\times$ is
     $\phi(n)$, where $\phi$ is the Euler $\phi$ function.
105 +
```

Hình 9. Định lý được dịch bằng Colada

Khó khăn. Do hệ thống mới ở bước đầu xây dựng nên việc sử dụng phần mềm chủ yếu nhằm mục đích kiểm nghiệm hoạt động của hệ thống. Chính vì vậy quá trình viết chương trình dịch gặp rất nhiều khó khăn như thiếu tài liệu hướng dẫn chi tiết, thiếu thông tin về các hàm, thiếu quy tắc ngữ pháp, v.v.. Vì vừa phải kết hợp việc soạn thảo chương trình dịch với phản hồi tới nhóm thiết kế để cập nhật thư viện nên mất rất nhiều thời gian để có thể hoàn thành một chương trình hoàn chỉnh. Hệ thống vẫn còn bước đầu sơ khai nên vẫn chưa thể hiểu được những quan hệ toán học phức tạp. Hơn nữa, do mỗi thành viên làm việc trên những file độc lập với nhau nên có những khái niệm được định nghĩa sử dụng macro có giá trị địa phương sẽ bị trùng lặp nhiều lần, điều này đòi hỏi phải có một hệ thống sắp xếp chặt chẽ để có thể quy chuẩn. Dự án còn một chặng đường dài để có thể chạm tới những mục tiêu đề ra.

5. Lời kết

Trong cuốn "Cái bóng của tư duy", Penrose có nói đại ý: cho đến nay, bất kỳ một tri thức nào của nhân loại cũng phải nằm trong một bộ óc cụ thể nào đó (nói nôm na, nếu "nhân loại" biết điều gì đó thì có nghĩa là ít nhất một người biết điều đó!). Tuy nhiên, bộ óc của mỗi cá nhân đều có giới hạn của nó. Như vậy, nếu nhân loại có ý định vượt qua mọi giới hạn của tri thức thì chỉ còn cách dựa vào máy tính. Sẽ đến ngày tri thức nhân loại được lưu giữ trong hệ thống máy tính, và thỉnh thoảng máy sẽ cho ta những tri thức mà ta không thể biết chúng từ đâu ra, tức là không biết chúng được "chứng minh" như thế nào.

Thật khó chấp nhận cái ngày mà máy tính cho ta các lý thuyết toán học mới, các kết quả mới, mà chỉ "nó" biết cách chứng minh.

Tuy nhiên, các nhà Toán học sẽ biết cách tìm chỗ đứng không thể thay thế được của mình trong cơn bão 4.0, với những máy tính biết tư duy.

Voevodsky (1966-2017), nhà toán học Nga được giải thưởng Fields năm 2002, đề ra một chương trình lớn được gọi là "Univalent Foundations of Mathematics", với tham vọng xây dựng lại toàn bộ cơ sở toán học theo ngôn ngữ hình thức. Khác với công việc của Bourbaki, dự án của Voevodsky khi hoàn thành sẽ giúp kiểm tra tự động các chứng minh toán học. Theo ông, đến ngày đó, khi nhà toán học gửi một công trình mới đến toà soạn, anh ta sẽ phải gửi thêm một chương trình kiểm tra tự động. Như vậy, người phản biện không cần kiểm tra tính đúng đắn của bài báo nữa, mà chỉ nhận xét về tầm quan trọng mà thôi. Tiếc rằng Voevodsky đã ra đi quá sớm, khi dự án đầy tham vọng của ông vẫn còn dang dở.

Nhưng nhìn vào dự án FAB với ý tưởng táo bạo của giáo sư Hales ở giai đoạn sau của dự án về việc sử dụng CNL cùng với bộ chuyển đổi từ văn bản toán học viết dưới dạng Latex và bằng ngôn ngữ tiếng Anh sang CNL, rồi từ CNL lại chuyển sang Lean để kiểm tra chứng minh, chúng ta có thể hi vọng về một tương lai không xa sẽ có chương trình kiểm tra tự động bằng máy tính, với đầu vào chỉ là các văn bản Latex. Khi đó sẽ không cần con người làm công việc nhàm chán là kiểm tra các bài báo của các nhà toán học gửi đến một tạp chí nào đó nữa, mà chỉ cần biến đổi đôi chút file Latex của tác giả, máy tính có thể kiểm tra tính đúng đắn của các chứng minh một cách chặt chẽ. Đương nhiên để thực hiện được điều đó cần tiếp tục có một dự án dài hơi nữa, phát triển tiếp những gì mà FAB đã mở ra , chúng tôi hi vọng sẽ còn được tiếp tục tham gia và đóng góp cho hướng phát triển mới này.

Nhờ sự động viên hỗ trợ, và tạo điều kiện của Ban lãnh đạo Trường Đại học Thăng Long cũng như Ban lãnh đạo Khoa Toán - Tin và Viện Timas, có thể nói dự án "Formal Abstract in Mathematics" đã được triển khai thành công, mặc dù kết quả đạt được còn khiêm tốn so với kỳ vọng, nhưng đã đánh dấu lần đầu tiên Nhà trường có hợp đồng ký kết hợp tác nghiên cứu khoa học với một trường đại học lớn ở Mỹ và đã thu về những ngoại hối đầu tiên từ nghiên cứu khoa học. Chúng tôi cũng gửi lời cảm ơn chân thành tới PGS. TSKH. Tạ Thị Hoài An, Viện Toán học, Viện Hàn lâm Khoa học và Công nghệ Việt Nam, người đã kết nối giáo sư Hales với Trường Đại học Thăng Long; và TS. Trần Nam Trung, Viện Toán học, Viện Hàn lâm Khoa học và Công nghệ Việt Nam, người đã chuẩn bị cho nhóm những bước đi đầu tiên làm quen với Lean.

Tài liệu tham khảo

[1] Asperti A., Ricciotti W., Sacerdoti Coen C., and Tassi E., (2011), The Matita Interactive Theorem

Prover, in Automated Deduction – CADE-23, vol. 6803, pp. 64–69.

[2] Bansal, K., Loos, S., Rabe, M., Szegedy, C., and Wilcox, S. (2019), HOList: An Environment for Machine Learning of Higher Order Logic Theorem Proving, in International Conference on Machine Learning, May 2019, pp. 454–463.

[3] Doorn, F.V., Ebner, G., and Lewis, R. Y., (2020), Maintaining a Library of Formal Mathematics, ArXiv200403673 Cs Math, vol. 12236, 2020, pp. 251–267.

[4] Hales, T., (1994), The status of the Kepler Conjecture, The Mathematical Intelligencer 16 (3), 47-58.

[5] Hales, T., (2005), A proof of the Kepler conjecture, Annals of Mathematics, 162 (2005), 1065–1185

[6] Hales, T., (2015), Formal proof, Notices of the AMS, N.11.

[7] Hales, T., (2019), An argument for controlled natural languages in mathematics, June 4.

[8] Hales, T., (2019), Reader's guide to the Colada Language.

[9] Hales, T., (2020), Controlled natural language for type theory, AITP 2020.

[10] Hales, T., (2020), Formal Methods in Mathematics, Conference of Lean Together 2020.

[11] Hales, T., (2019), Guidelines for the Planet-Math translation to Colada.

[12] Knuth, D. E., (July 1965), On the translation of languages from left to right (PDF), Information and Control, 8 (6): 607–639. DOI:10.1016/S0019-9958(65)90426-2.

[13] Moura, L.D., Kong, S., Avigad, J., Doorn, F.V, and Raumer, J.V, (1995), The Lean Theorem Prover (System Description), in Automated Deduction - CADE-25, vol. 9195, pp. 378–388.

[14] Nipkow, T., Paulson, L. C., and Wenzel, M., (2002), Isabelle/HOL: A Proof Assistant for Higher-Order Logic, Springer Science & Business Media.

[15] Penrose, R., (1989), Shadows of the Mind: A Search for the Missing Science of Consciousness, Oxford University Press.

[16] Voevodsky, V., (2014), Computer Proof Assistants and Univalent Foundations of Mathematics CMA 2014, Kuwait.

[17] The Coq Proof Assistant, https://coq.inria.fr/

[18] The Lean Prover Mathlib, https://github.com/leanprover-community/mathlib

[19] The Top 100 Theorems, http://pirate.shu.edu/~kahlnath/Top100.html

[20] PlanetMath, https://planetmath.org/