

Giới thiệu về **lý thuyết hình thái** trong **khoa học máy tính**

Bế Trọng Nghĩa

Ngày 21 tháng 11 năm 2023

CẢNH BÁO: RẤT ÍT HÌNH - RẤT NHIỀU CHỮ!!!

Trong bài thuyết trình này...

- Ngôn ngữ lập trình.

Trong bài thuyết trình này...

- ~~Ngôn ngữ lập trình.~~ → Bộ của ngôn ngữ lập trình.

Trong bài thuyết trình này...

- ~~Ngôn ngữ lập trình.~~ → Bộ của ngôn ngữ lập trình.
- Phương pháp hình thức.

Trong bài thuyết trình này...

- ~~Ngôn ngữ lập trình.~~ → Bộ của ngôn ngữ lập trình.
- ~~Phương pháp hình thức.~~ → Mẹ của phương pháp hình thức.

Trong bài thuyết trình này...

- ~~Ngôn ngữ lập trình.~~ → Bố của ngôn ngữ lập trình.
- ~~Phương pháp hình thức.~~ → Mẹ của phương pháp hình thức.
- Toán học và logic học.

Trong bài thuyết trình này...

- ~~Ngôn ngữ lập trình.~~ → Bố của ngôn ngữ lập trình.
- ~~Phương pháp hình thức.~~ → Mẹ của phương pháp hình thức.
- Toán học và logic học và lịch sử và triết học và...

Lý thuyết

hình thái?



Wikipedia

https://vi.wikipedia.org/wiki/Lý_thuyết_hình_thái_kinh_tế · [Translate this page](#) ·

Lý thuyết hình thái

Trong toán học, logic và khoa học máy tính, một **lý thuyết hình thái** hoặc một hệ hình thái là một hệ thống hình thức trong đó mọi đối tượng đều có một hình ...



Wikipedia

https://vi.wikipedia.org/wiki/Lý_thuyết_hình_thái_đồng_luân · [Translate this page](#) ·

Lý thuyết hình thái đồng luân

Bìa cuốn sách Homotopy Type Theory: nền tảng thống nhất của toán học. Trong logic toán và khoa học máy tính, **lý thuyết hình thái đồng luân** (tiếng Anh: homotopy ...



lyluanchinhtrivatruyenthong.vn

<https://lyluanchinhtrivatruyenthong.vn/...> · [Translate this page](#) ·

Học thuyết Hình thái kinh tế - xã hội của C.Mác một cách tiếp ...

1 Jun 2021 — Xã hội tư bản ra đời do quá trình hình thành hành động hợp lý của các cá nhân mà nguồn gốc của nó là tri thức khoa học kết hợp với ý thức (động ...



Công ty Luật Minh Khuê

<https://luatminhkhue.vn/hinh-thai-...> · [Translate this page](#) ·

Hình thái kinh tế xã hội là gì? Ví dụ 5 hình thái kinh tế xã hội

1 Nov 2022 — + Tính tất yếu của thời kỳ quá độ lên chủ nghĩa xã hội được lý giải từ các căn cứ sau đây: Một là, chủ nghĩa tư bản và chủ nghĩa xã hội khác ...

★★★★★ Rating: 5 · 3 votes

Hình thái kinh tế xã hội là gì ? · Sự ra đời của hình thái kinh tế...

GIÁ TRỊ KHOA HỌC CỦA LÝ LUẬN HÌNH THÁI KINH TẾ

28 Jan 2021 — **Lý luận hình thái kinh tế - xã hội** là lý luận cơ bản của chủ nghĩa duy vật lịch sử do C. Mác xây dựng lên, có vị trí quan trọng trong triết học ...



VNU Journal of Science

<https://js.vnu.edu.vn/SSH/article/view/PDF> ·

HỌC THUYẾT HÌNH THÁI KINH TẾ - XÃ HỘI VỚI VIỆC ...

by VT Dương · 2003 — XIX, Số 4, 2003 biến đổi thực tế, để nhận thức qui luật vận động của lịch sử, đồng thời bổ sung, phát triển **lý luận**. Cách mạng Việt Nam hiện nay diễn ra trong ...



Hội nông dân tỉnh Hà Tĩnh

<https://hoionongdanhatinh.vn/news/...> · [Translate this page](#) ·

Từ “học thuyết hình thái kinh tế xã hội” của chủ nghĩa Mác ...

4 Dec 2022 — ... thuyết hình thái kinh tế-xã hội của Mác là sự thống nhất biện chứng giữa lý luận và thực tiễn, giữa tính khoa học và tính đảng. Vì thế, đã ...



Studocu

<https://www.studocu.com/document/...> · [Translate this page](#) ·

Học thuyết hình thái kinh tế-xã hội - CHỦ'ÔNG 3

Hay chương chủ nghĩa duy vật lịch sử học **thuyết hình thái kinh tế xã hội** sản xuất vật chất là cơ sở cho sự tồn tại và phát triển của xã hội khái niệm sản ...



Images for ly thuyet hinh thai ·



mác xít



thuyết trình



triết học



phân kỳ



mác lenin



Lý thuyết hình thái
∈ Chủ nghĩa *Marx–Lenin*



Lý thuyết hình thái
∈ *Chủ nghĩa Marx–Lenin*

🧐👉 Thực ra không phải vậy 🤯🤔

Thực trạng tại Việt Nam

- *Lý thuyết hình thái (LTHT)* ban đầu được ra như một giải pháp cho **khủng hoảng nền tảng toán học**, nhưng sau này lại có mối quan hệ đặc biệt quan trọng với *khoa học máy tính* (về cơ bản *LTHT* là lĩnh vực nghiên cứu các *hệ thống kiểu*).

Thực trạng tại Việt Nam

- *Lý thuyết hình thái (LTHT)* ban đầu được ra như một giải pháp cho **khủng hoảng nền tảng toán học**, nhưng sau này lại có mối quan hệ đặc biệt quan trọng với *khoa học máy tính* (về cơ bản *LTHT* là lĩnh vực nghiên cứu các *hệ thống kiểu*).
- *Lý thuyết hình thái* gần như không có tài liệu bằng tiếng Việt và không có đề tài nghiên cứu cụ thể nào về lĩnh vực này được công bố.

Thực trạng tại Việt Nam



Sẽ có thêm người quan tâm đến lĩnh vực này chứ?

TOÁN HỌC

Tôi có ngồi nhầm lớp không?

Ở ĐÂY CHÚNG TA BÀN VỀ TOÁN HỌC?

Ở ĐÂY CHÚNG TA BÀN VỀ TOÁN HỌC?

  Thực ra không hẳn vậy  

Lý thuyết
hình thái

Lịch sử của lý thuyết hình thái

Ctrl + H...

Thời “tiền sử”

- *Ý tưởng cốt lõi*: Phân lớp các đối tượng toán học.

Thời “tiền sử”

- *Ý tưởng cốt lõi*: Phân lớp các đối tượng toán học.
- Ví dụ: Một số định nghĩa từ bộ *Cơ sở* của Euclid.

Thời “tiền sử”

- *Ý tưởng cốt lõi*: Phân lớp các đối tượng toán học.
- Ví dụ: Một số định nghĩa từ bộ *Cơ sở* của Euclid.
 1. *Điểm* là cái không [thể] chia nhỏ.
 2. *Đường* là một lượng dài không có chiều rộng.
 3. Một *đường thẳng* là một *đường* (bất kỳ) mà trên đó các *điểm* nằm ngang bằng.

Thời “tiền sử”

- *Ý tưởng cốt lõi*: Phân lớp các đối tượng toán học.
- Euclid đã định nghĩa và phân loại 3 đối tượng trong ví dụ.

Thời “tiền sử”

- *Ý tưởng cốt lõi*: Phân lớp các đối tượng toán học.
- Euclid đã định nghĩa và phân loại 3 đối tượng trong ví dụ.
- Ngoài ra, trong bộ *Cơ sở*, Euclid cũng đã chỉ ra các “đối tượng” thuộc “lớp” nào → Tránh kết quả không mong muốn.

Thời “tiền sử”

- *Ý tưởng cốt lõi*: Phân lớp các đối tượng toán học.
- Euclid đã định nghĩa và phân loại 3 đối tượng trong ví dụ.
- Ngoài ra, trong bộ *Cơ sở*, Euclid cũng đã chỉ ra các “đối tượng” thuộc “lớp” nào → Tránh kết quả không mong muốn.
- Ví dụ:

Thời “tiền sử”

- *Ý tưởng cốt lõi*: Phân lớp các đối tượng toán học.
- Euclid đã định nghĩa và phân loại 3 đối tượng trong ví dụ.
- Ngoài ra, trong bộ *Cơ sở*, Euclid cũng đã chỉ ra các “đối tượng” thuộc “lớp” nào → Tránh kết quả không mong muốn.
- Ví dụ:

$$\text{Đường} \cap \text{Đường} = \text{Điểm}$$

Thời “tiền sử”

- *Ý tưởng cốt lõi*: Phân lớp các đối tượng toán học.
- Euclid đã định nghĩa và phân loại 3 đối tượng trong ví dụ.
- Ngoài ra, trong bộ *Cơ sở*, Euclid cũng đã chỉ ra các “đối tượng” thuộc “lớp” nào → Tránh kết quả không mong muốn.
- Ví dụ:

$$\text{Đường} \cap \text{Đường} = \text{Điểm}$$

$$\text{Điểm} \cap \text{Điểm} = ???$$

Thời “tiền sử”

- Tuy nhiên, ý tưởng ấy mới chỉ được áp dụng một cách **ngầm định** cho đến mãi về sau...

Thời “tiền sử”

- Tuy nhiên, ý tưởng ấy mới chỉ được áp dụng một cách **ngầm định** cho đến mãi về sau...
- Từ thế kỷ XIX trở đi, các *hệ thống toán học* (HTTH) ngày càng thiếu “trực quan” hơn:

Thời “tiền sử”

- Tuy nhiên, ý tưởng ấy mới chỉ được áp dụng một cách **ngầm định** cho đến mãi về sau...
- Từ thế kỷ XIX trở đi, các *hệ thống toán học* (HTTH) ngày càng thiếu “trực quan” hơn:
 1. Các HTTH trở nên *trừu tượng* hơn bao giờ hết.

Ví dụ: Giải tích toán học và hình học phi Euclid.

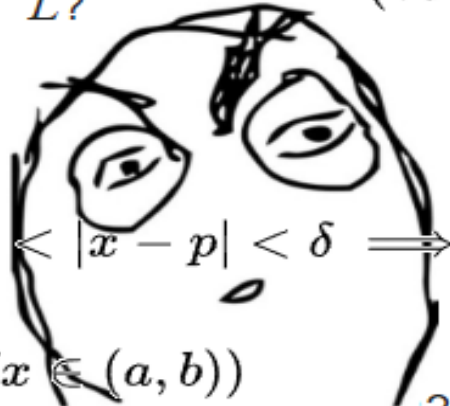
(ε, δ) -definition of limit

??? ??

$$\lim_{x \rightarrow p} f(x) = L$$

$(\forall \varepsilon > 0)$

$L?$



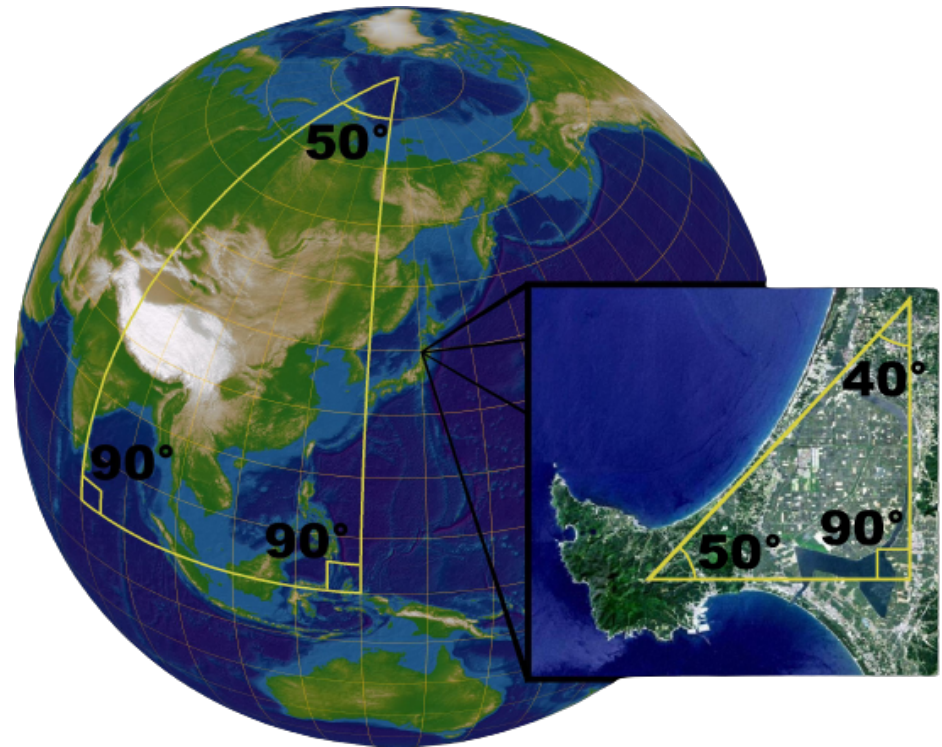
$(0 < |x - p| < \delta \implies |f(x) - L| < \varepsilon)$

$(\forall x \in (a, b))$

???

??

???



Thời “tiền sử”

- Tuy nhiên, ý tưởng ấy mới chỉ được áp dụng một cách **ngầm định** cho đến mãi về sau...
- Từ thế kỷ XIX trở đi, các *hệ thống toán học* (HTTH) ngày càng thiếu “trực quan” hơn:
 1. Các HTTH trở nên *trừu tượng* hơn bao giờ hết.
 2. Các HTTH được *hình thức hóa*.

Ví dụ: Khái niệm giới hạn hàm số theo định nghĩa (ε, δ) bởi Bolzano, Cauchy và Weierstrass.

Thời “tiền sử”

- Tuy nhiên, ý tưởng ấy mới chỉ được áp dụng một cách **ngầm định** cho đến mãi về sau...
- Từ thế kỷ XIX trở đi, các *hệ thống toán học* (HTTH) ngày càng thiếu “trực quan” hơn:
 1. Các HTTH trở nên *trừu tượng* hơn bao giờ hết.
 2. Các HTTH được *hình thức hóa*.
 3. Các HTTH được thực hiện qua đối tượng không phải con người: *Máy tính!*

Thời “tiền sử”

- Nguy cơ các **ngịch lý** xảy ra vì sự “**phân loại ngầm định**” không còn đủ “trực quan”!

Thời “tiền sử”

- Nguy cơ các **nghịch lý** xảy ra vì sự “**phân loại ngầm định**” không còn đủ “trực quan”!
- Sẽ ra sao nếu...

Một *nghịch lý logic* xuất hiện từ những *luật suy diễn* **hợp lệ**?

Thời “tiền sử”

- Năm 1879, Gottlob Frege đã xuất bản cuốn sách *Begriffsschrift* (*Hệ ghi ý*), đặt nền móng cho sự *hình thức hóa logic học*.

Thời “tiền sử”

- Năm 1879, Gottlob Frege đã xuất bản cuốn sách *Begriffsschrift* (Hệ ghi ý), đặt nền móng cho sự *hình thức hóa logic học*.
- Sau đó, ông tiếp tục nghiên cứu và xuất bản nhiều cuốn sách quan trọng như *Die Grundlagen der Arithmetik* (Nền tảng số học) và *Grundgesetze der Arithmetik* (Các định luật cơ bản của số học), củng cố lập luận rằng: *Số học là một nhánh của logic học và có thể được suy ra từ logic*.

Chuẩn bị lên đời...

- Tuy nhiên, trong hệ thống toán học của ông, có một định luật như sau:

Chuẩn bị lên đời...

- Tuy nhiên, trong hệ thống toán học của ông, có một định luật như sau:

Định luật cơ bản V: $\varepsilon' f(\varepsilon) = \varepsilon' g(\varepsilon) \Leftrightarrow \forall x (f(x) = g(x))$

“Khoảng giá trị” của hàm $f(\varepsilon)$ bằng “khoảng giá trị” của hàm $g(\varepsilon)$ khi và chỉ khi với mọi x ta có $f(x) = g(x)$.

Chuẩn bị lên đời...

- Tuy nhiên, trong hệ thống toán học của ông, có một định luật như sau:

Định luật cơ bản V: $\varepsilon' f(\varepsilon) = \alpha' g(\alpha) \Leftrightarrow \forall x (f(x) = g(x))$

- Trong ngôn ngữ của *lý thuyết tập hợp*:

Tập hợp A và tập hợp B bằng nhau khi và chỉ khi mọi phần tử x thuộc A cũng đồng thời thuộc B .

Chuẩn bị lên đời...

- Về mặt logic, một *tập hợp* là một nhóm các đối tượng *thỏa mãn những đặc tính (vị từ) nhất định*.

Chuẩn bị lên đời...

- Về mặt logic, một *tập hợp* là một nhóm các đối tượng *thỏa mãn những đặc tính (vị từ) nhất định*.
- Ví dụ:

S là tập hợp các số nguyên dương.

$S = \{x \mid P(x)\}$ với $P(x)$ là vị từ: “ x là số nguyên dương”.

Chuẩn bị lên đời...

- Về mặt logic, một *tập hợp* là một nhóm các đối tượng *thỏa mãn những đặc tính (vị từ) nhất định*.
- Ví dụ:

S là tập hợp các số nguyên dương.

$S = \{x \mid P(x)\}$ với $P(x)$ là vị từ: “ x là số nguyên dương”.

- Về cơ bản, *lý thuyết tập hợp* được định nghĩa bởi các *luật suy diễn logic* và *tiên đề tập hợp*.

Câu hỏi:

Cho tập hợp T gồm các tập hợp x không phải là phần tử của chính nó.

$$T = \{x \mid x \text{ là tập hợp}, x \notin x\}$$

T có phải là phần tử của chính nó không?

Nếu T là phần tử của chính nó,
thì T phải không là phần tử của chính nó.

$$T \in T \Leftrightarrow T \notin T$$

Nếu T không phải phần tử của chính nó,
thì T là phần tử của chính nó.

$$T \notin T \Leftrightarrow T \in T$$

MÂU THUẤN

Ảo thật đây?

Sự ra đời của lý thuyết hình thái

- Nhà toán học Bertrand Russell đã chỉ ra mâu thuẫn trên, đây chính là *nghịch lý Russell* nổi tiếng.

Sự ra đời của lý thuyết hình thái

- Nhà toán học Bertrand Russell đã chỉ ra mâu thuẫn trên, đây chính là *nghịch lý Russell* nổi tiếng.
- Mặc dù đây là một vấn đề tưởng như đơn giản, song quan niệm về *nền tảng toán học dựa trên logic* bị ảnh hưởng nặng nề.

Sự ra đời của lý thuyết hình thái



Nguyên lý bùng nổ:

Từ một mâu thuẫn, **mọi mệnh đề dù đúng hay sai đều có thể được suy ra!**

Sự ra đời và phát triển của lý thuyết hình thái

- Không thể chấp nhận việc nền tảng toán học có mâu thuẫn
→ Cần giải pháp khắc phục mâu thuẫn.

Sự ra đời và phát triển của lý thuyết hình thái

- Không thể chấp nhận việc nền tảng toán học có mâu thuẫn
→ Cần giải pháp khắc phục mâu thuẫn.
- *Nghịch lý Russell* (và nhiều nghịch lý khác) xảy ra bởi sự “tùy tiện” trong việc *tự tham chiếu*.

Sự ra đời và phát triển của lý thuyết hình thái

- Không thể chấp nhận việc nền tảng toán học có mâu thuẫn
→ Cần giải pháp khắc phục mâu thuẫn.
- *Nghịch lý Russell* (và nhiều nghịch lý khác) xảy ra bởi sự “tùy tiện” trong việc *tự tham chiếu*.
- Russell đã đưa ra kết luận: Để tránh mâu thuẫn, **bất cứ thứ gì liên quan đến toàn bộ tập hợp thì không được là một trong những tập hợp đó.**

Sự ra đời và phát triển của lý thuyết hình thái

- Russell đã phát triển *LTHT* như một **nền tảng toán học được-chứng-minh-là-hoàn-toàn-không-có-mâu-thuẫn** (nhất quán), tuy nhiên điều này đã được chứng minh là không thể theo *những định lý bất toàn* của Gödel.
 - Định lý 1: Một hệ hình thức **nhất quán** có khả năng biểu diễn *số học cơ bản* thì sẽ **có những mệnh đề không chứng minh được**.
 - Định lý 2: Một hệ hình thức **nhất quán** không thể **tự chứng minh nó nhất quán**.

Sự ra đời và phát triển của lý thuyết hình thái

- Russell đã phát triển *LTHT* như một **nền tảng toán học được-chứng-minh-là-hoàn-toàn-không-có-mâu-thuẫn**, tuy nhiên điều này đã được chứng minh là không thể theo *những định lý bất toàn* của Gödel.
- Đồng thời, vai trò **nền tảng toán học** của *LTHT* dần bị thay thế bởi các lý thuyết khác, đặc biệt là *lý thuyết tập hợp ZFC*.

Sự ra đời và phát triển của lý thuyết hình thái

- Russell đã phát triển *LTHT* như một **nền tảng toán học được-chứng-minh-là-hoàn-toàn-không-có-mâu-thuẫn**, tuy nhiên điều này đã được chứng minh là không thể theo *những định lý bất toàn* của Gödel.
- Đồng thời, vai trò **nền tảng toán học** của *LTHT* dần bị thay thế bởi các lý thuyết khác, đặc biệt là *lý thuyết tập hợp ZFC*.
- Tuy nhiên, những chức năng khác của *LTHT* cũng đã được khám phá, đặc biệt là trong lĩnh vực *khoa học máy tính*.

Một số cột mốc tiêu biểu sau đó

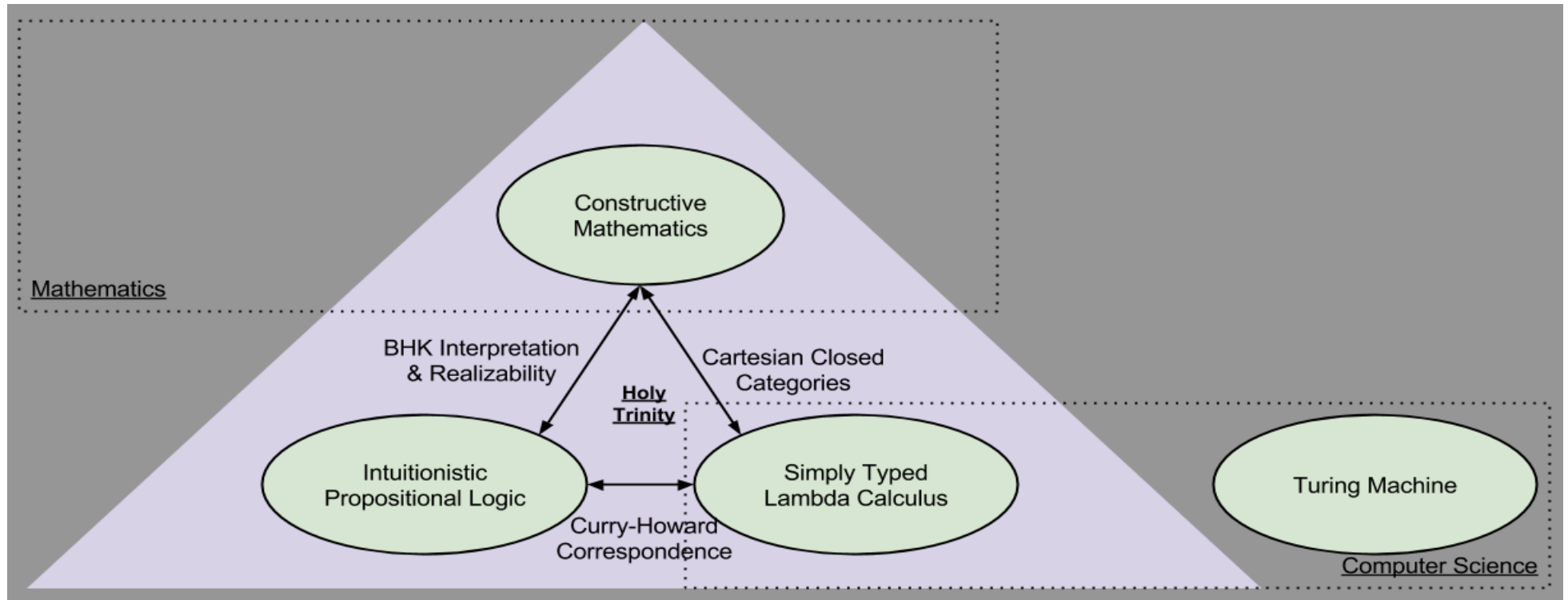
1. *Tương ứng Curry–Howard* của Curry và Howard (1934–1969).

– Chứng minh *mỗi quan hệ trực tiếp* giữa **chương trình máy tính** và **chứng minh toán học**.

– Sự tương quan này được biết đến với cái tên **Chứng minh-là-chương trình** hoặc **Mệnh đề-là-kiểu**.

– Là nền tảng của những liên hệ toán–tin, tiêu biểu là *Tương ứng Curry–Howard–Lambek* (mở rộng thêm *lý thuyết phạm trù*).

– Mọi chứng minh toán học đều có thể biểu diễn bằng một chương trình máy tính, và ngược lại; và còn hơn thế nữa!!!



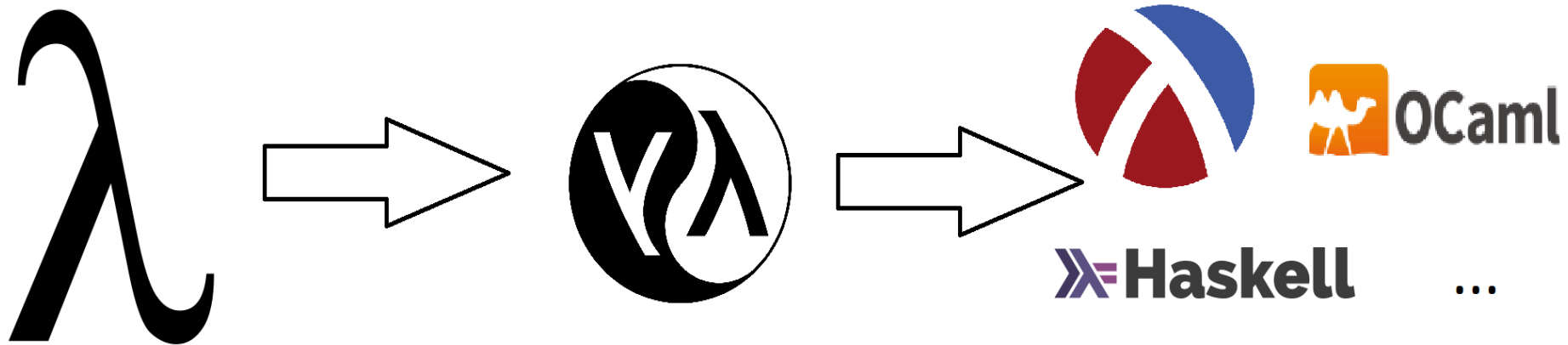
Một số cột mốc tiêu biểu sau đó

2. *Phép tính lambda* của Church (1936-1940).

- Một *ngôn ngữ lập trình* đơn giản, đồng thời là *mẫu hình tính toán* giống như *máy Turing*.

Hàm f khả tính lambda
 \Leftrightarrow Hàm f khả tính Turing

- Nền tảng cho nhiều ngôn ngữ lập trình sau này (đặc biệt là *ngôn ngữ lập trình hàm*).



Một số cột mốc tiêu biểu sau đó

3. *Automath* của de Bruijn (1967–2003).

- Một ngôn ngữ lập trình được thiết kế để biểu diễn các chứng minh toán học trong máy tính.
- Dự án tiên phong cho những công cụ hỗ trợ chứng minh định lý sau này.

Một số cột mốc tiêu biểu sau đó

4. *Lý thuyết hình thái trực giác* của Martin-Löf (1971–1984).

- *LTHT* theo quan điểm của *chủ nghĩa kiến thiết*, với đặc tính nổi bật là sử dụng **kiểu phụ thuộc**.
- Nền tảng cho nhiều *ngôn ngữ lập trình* sau này.

Một số cột mốc tiêu biểu sau đó

5. *Phép tính kiến thiết* của Coquand và Huet (1967–2003).

- Cơ sở của **công cụ hỗ trợ chứng minh định lý Coq**.
- Nền tảng cho nhiều *công cụ hỗ trợ chứng minh định lý* sau này.



Một số cột mốc tiêu biểu sau đó

6. *Nền tảng đơn diệp và lý thuyết hình thái đồng luân* của Voevodsky (2006-2013+).

- Bước phát triển mới của *lý thuyết hình thái trực giác*.
- *Lý thuyết hình thái* tiếp tục phát triển song hành cùng *khoa học máy tính*, và thậm chí quay lại với vai trò **nền tảng toán học**.

Một số cột mốc tiêu biểu sau đó

1. *Tương ứng Curry–Howard* của Curry và Howard (1934–1969).
2. *Phép tính lambda* của Church (1936–1940).
3. *Automath* của de Bruijn (1967–2003).
4. *Lý thuyết hình thái trực giác* của Martin-Löf (1971–1984).
5. *Phép tính kiến thiết* của Coquand và Huet (1967–2003).
6. *Nền tảng đơn điệp và lý thuyết hình thái đồng luân* của Voevodsky (2006–2013+).
7. ???

Lịch sử là thế.

Rốt cuộc lý thuyết hình thái là cái gì?

Sơ lược về lý thuyết hình thái

Bài thuyết trình này dài quá ông ơi...

Một vài lưu ý

- Vì có nhiều loại *LTHT*, phần này sẽ chỉ giới thiệu qua một vài khái niệm nền tảng của *LTHT* nói chung.

Một số khái niệm cơ bản

- **Kiểu và khái niệm:** Mọi *khái niệm* đều có một *kiểu*.
- Ký pháp: *khái niệm* : *kiểu*.
- Trong ngôn ngữ lập trình, khái niệm có thể là *biến, hằng, hàm...*

Ví dụ (C++):

```
int life = 42;
```

Trong ví dụ này, `life` là *khái niệm (biến)*, và `int` là *kiểu*.

Một số khái niệm cơ bản

- **Luật tính toán:** hay còn gọi là *luật rút gọn*.
- Cho hai ví dụ:

$$1 + 4 : \mathbb{N} \text{ và } 5 : \mathbb{N}$$

Có thể nói $1 + 4 : \mathbb{N}$ rút gọn thành $5 : \mathbb{N}$.

- Những khái niệm không rút gọn được gọi là *khái niệm chính tắc*.
- Ví dụ: $5 : \mathbb{N}$ không rút gọn được hơn nữa.

Một số khái niệm cơ bản

- Một số kiểu đặc biệt:
 - *Kiểu rỗng*: Không có khái niệm nào thuộc kiểu này.

Ứng với khái niệm *sai* trong logic,
và \emptyset trong lý thuyết tập hợp.

Một số khái niệm cơ bản

- Một số kiểu đặc biệt:

- *Kiểu đơn vị*: Có duy nhất một khái niệm thuộc kiểu này.

Ứng với khái niệm *tập chỉ* có 1 phần tử trong lý thuyết tập hợp.

Một số khái niệm cơ bản

- Một số kiểu đặc biệt:

- *Kiểu tích*: Xét kiểu tích của hai kiểu A và B .

Ta có: $(a, b) : A \times B$ là một cặp có thứ tự, trong đó $a : A$ và $b : B$.

Ứng với khái niệm \wedge trong logic
và \cap trong lý thuyết tập hợp.

Một số khái niệm cơ bản

- Một số kiểu đặc biệt:

- *Kiểu tổng*: Xét kiểu tổng của hai kiểu A và B .

Khái niệm thuộc kiểu $A + B$ sẽ tồn tại dưới dạng $a : A$ hoặc $b : B$.

Ứng với khái niệm \vee trong logic và
 \cup trong lý thuyết tập hợp.

Một số khái niệm cơ bản

- Trong các ngôn ngữ lập trình:
 - Kiểu tích thường được biểu diễn dưới dạng struct.
 - Kiểu tổng thường được biểu diễn dưới dạng enum.

Ứng dụng của lý thuyết hình thái

Chờ mãi phần này...

*Hệ thống kiểu
cho các ngôn ngữ lập trình*

Hệ thống kiểu

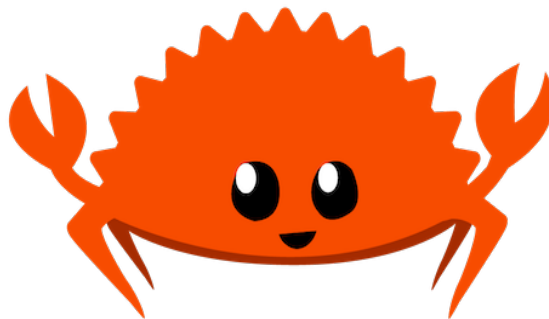
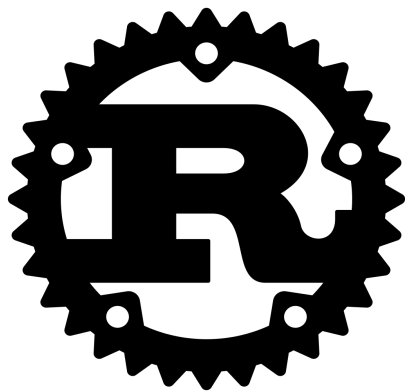
- *Hệ thống kiểu* là thành phần không thể thiếu với mọi ngôn ngữ lập trình hiện đại.
- Về cơ bản, một hệ thống kiểu bao gồm các luật liên quan đến **kiểu** và **khái niệm** (ví dụ như *biến, hàm, biểu thức...*).

Hệ thống kiểu

- Về cơ bản, một hệ thống kiểu bao gồm các luật liên quan đến **kiểu** và **khái niệm** (*ví dụ như biến, hàm, biểu thức...*).
- Ví dụ:
 - **Hành động** nào được phép thực hiện lên **khái niệm**?
Ví dụ: Trong C++, `double x1` có thể được ép kiểu sang `int x2`.
 - **Khái niệm** được phép có **giá trị** nào?
Ví dụ: Trong Java, hàm `main` hợp lệ phải có kiểu `void`.

Hệ thống kiểu

- Ví dụ thực tế: Hệ thống kiểu của ngôn ngữ lập trình Rust!



Hệ thống kiểu của ngôn ngữ Rust

- Rust là ngôn ngữ lập trình *đa năng, đa mẫu hình*, với mục tiêu đem đến cho người sử dụng *hiệu năng, tính đảm bảo và hiệu suất làm việc*.

Hệ thống kiểu của ngôn ngữ Rust

- **Về hiệu năng:** Rust là ngôn ngữ lập trình hệ thống, cho nên không sử dụng *môi trường thực thi* hay *bộ thu gom rác*, hỗ trợ **giảm lượng tài nguyên cần dùng**.

Hệ thống kiểu của ngôn ngữ Rust

- Về **hiệu năng**: Rust là ngôn ngữ lập trình hệ thống, cho nên không sử dụng *môi trường thực thi* hay *bộ thu gom rác*, hỗ trợ **giảm lượng tài nguyên cần dùng**.
- Về **tính đảm bảo**: Rust có *hệ thống kiểu* được thiết kế tốt, cùng với việc sử dụng cơ chế “*quyền sở hữu*” để quản lý bộ nhớ, giúp **đảm bảo an toàn cho tác vụ đa luồng và quản lý bộ nhớ**.

QUYỀN SỞ HỮU VÀ QUYỀN KHÁC ĐỐI VỚI TÀI SẢN

Chương XI

QUY ĐỊNH CHUNG

Mục 1. M... C LẬP, THỰC HIỆN QUYỀN SỞ HỮU, QUYỀN KHÁC ĐỐI VỚI TÀI

SẢN

159.

Quyền sở hữu... chiếm hữu, quyền sử dụng và quyền định đoạt tài sản của chủ sở hữu theo quy...

Điều 159. Quyền... với tài sản

1. Quyền khác đối với tài sản là quyền của chủ thể trực tiếp nắm giữ, chi phối tài sản thuộc quyền sở hữu của chủ thể khác.
2. Quyền khác đối với tài sản bao gồm:
 - a) Quyền đối với bất động sản liền kề;
 - b) Quyền hưởng dụng;
 - c) Quyền bề mặt.

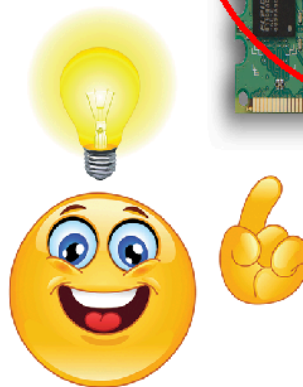
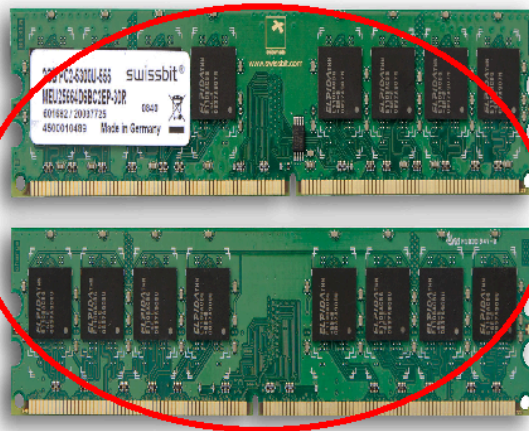
Điều 160. Nguyên tắc xác lập, thực hiện quyền sở hữu, quyền khác đối với tài sản

1. Quyền sở hữu, quyền khác đối với tài sản được xác lập, thực hiện trong trường hợp Bộ luật này, luật khác có liên quan quy định.

Quyền khác đối với tài sản vẫn có hiệu lực trong trường hợp quyền sở hữu được chuyển giao, trừ trường hợp Bộ luật này, luật khác có liên quan quy định khác.

2. Chủ sở hữu được thực hiện mọi hành vi theo ý chí của mình đối với tài sản nhưng không được trái với quy định của luật, gây thiệt hại hoặc làm ảnh hưởng đến lợi ích quốc gia, dân tộc, lợi ích công cộng, quyền và lợi ích hợp pháp của người khác.

3. Chủ thể có quyền khác đối với tài sản được thực hiện mọi hành vi trong phạm vi quyền được quy định tại Bộ luật này, luật khác có liên quan nhưng không được gây thiệt hại hoặc làm ảnh hưởng đến lợi ích quốc gia, dân tộc, lợi ích công cộng, quyền và lợi ích hợp pháp của chủ sở hữu tài sản hoặc của người khác.



Hệ thống kiểu của ngôn ngữ Rust

- Các khái niệm liên quan đến “*quyền sở hữu*”
 - **Quyền sở hữu:** Mọi giá trị đều thuộc sở hữu bởi một biến.

```
fn main() {  
    let life = String::from("42");  
}
```

Trong ví dụ này, biến `life` sở hữu giá trị thuộc kiểu `String` là `"42"`.

– **Nhượng quyền:** Quyền sở hữu có thể được chuyển giữa các biến/hàm, với điều kiện chỉ có một chủ sở hữu ở mọi thời điểm.

```
fn main() {  
    let life = String::from("42");  
    let anotherlife = life;  
  
    println!("Another Life: {}", anotherlife);  
    println!("Life: {}", life); // hetcuu💀💀💀  
}
```

Trong ví dụ này, biến `life` đã **nhượng quyền** sở hữu cho biến `anotherlife`.

```
fn main() {  
    let life = String::from("42");  
    let anotherlife = life;  
  
    println!("Another Life: {}", anotherlife);  
    println!("Life: {}", life); // hetcuu💀💀💀  
}
```

Câu lệnh cuối không thể được thực hiện, bởi quyền sở hữu giá trị đã bị chuyển cho biến khác.

Ngoài ra, với những biến có kiểu dữ liệu được cài đặt **cơ chế sao chép** (VD: kiểu nguyên thủy), **việc gán sẽ không nhượng quyền**.

– **Cho mượn:** Nếu không **nhượng quyền**, giá trị vẫn có thể được “**mượn**” thông qua **tham chiếu**.

```
fn main() {  
    let life = String::from("42");  
    let borrowlife = &life;  
  
    println!("Borrow Life: {}", borrowlife);  
    println!("Life: {}", life);  
}
```

Trong ví dụ này, biến borrowlife đã được **mượn** quyền truy cập vào biến life để **đọc**.

– **Thời gian tồn tại:** Khi ra ngoài phạm vi cho phép, biến sẽ bị hủy.

```
fn main() {  
    {  
        let bigchunggus = String::from("420");  
    }  
    println!("{}", 🍄🍄🍄, bigchunggus); // hetcuu💀💀💀  
}
```

Trong ví dụ này, biến bigchunggus đã **hết thời gian tồn tại** do đã **vượt ra ngoài phạm vi** khai báo, do đó câu lệnh cuối không thể được thực hiện.

Hệ thống kiểu của ngôn ngữ Rust

- Cơ chế “*quyền sở hữu*” giúp Rust có **hiệu năng** của một ngôn ngữ hệ thống (như C/C++), đồng thời không phải thực hiện quá trình quản lý bộ nhớ thủ công; cũng như tính **an toàn** khi **sử dụng bộ nhớ** hay **thực hiện các tác vụ đa luồng**.
- Việc xuất hiện cơ chế này là do hệ thống kiểu của Rust được xây dựng như một *hệ thống kiểu affine*.

Toán học cơ giới hóa

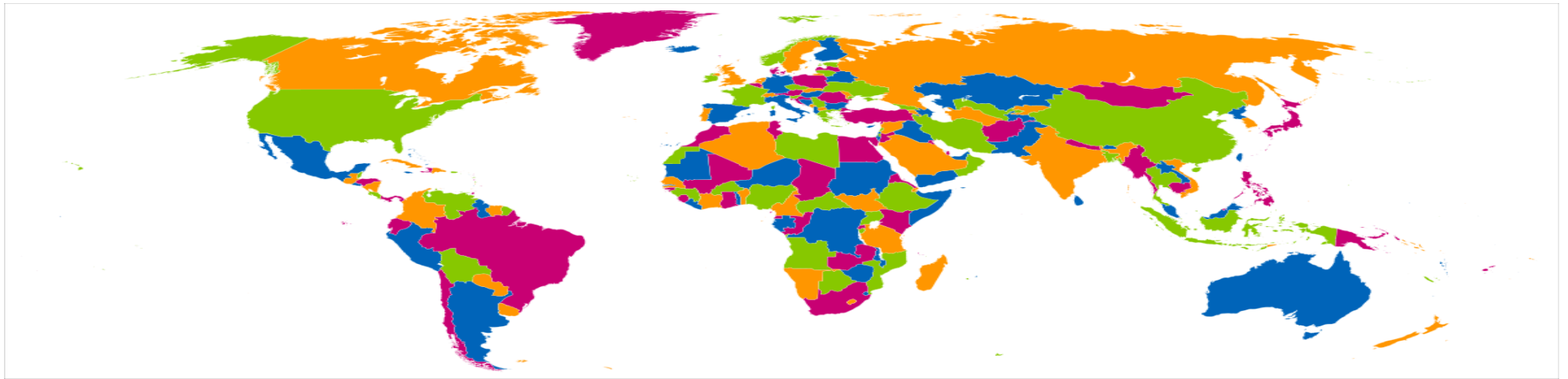
Cái gì vậy trời...

Toán học cơ giới hóa

- *Cơ giới hóa*, ở đây là việc phát triển các công cụ toán học trên nền tảng máy tính, cũng như tận dụng máy tính để thực hiện các tác vụ toán học.
- *LTHT* là cơ sở cho nhiều mặt của sự cơ giới hóa toán học, đặc biệt là *lý thuyết chứng minh*.

Chứng minh toán học bằng máy tính

- Một số thành tựu tiêu biểu:
 - **Định lý bốn màu (1976)**: Là định lý lớn đầu tiên được chứng minh bằng máy tính, bởi Appel, Haken, và Koch.



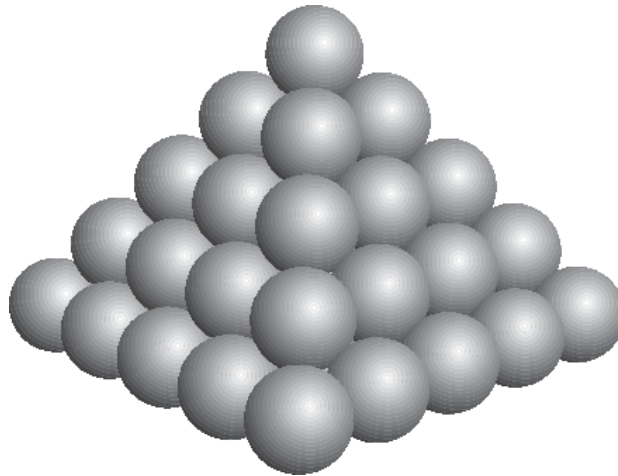
Chứng minh toán học bằng máy tính

- Một số thành tựu tiêu biểu:
 - Cần bao nhiêu số khởi điểm để bắt đầu giải được Sudoku?
(2012): Kết quả là 17, bởi McGuire, Tugemann, và Civario.

			8		1			
							4	3
5								
				7		8		
						1		
	2			3				
6							7	5
		3	4					
			2			6		

Chứng minh toán học bằng máy tính

- Một số thành tựu tiêu biểu:
 - **Giả thuyết Kelper** (2003–2014): Là giả thuyết hơn 400 năm sau mới được **chứng minh hình thức**, bởi Hales và cộng sự.



Phương pháp hình thức

- Ngoài hỗ trợ chứng minh toán học, các *phương pháp hình thức* cũng là ứng dụng của *LTHT*.
- *Phương pháp hình thức* là các kỹ thuật và công cụ chặt chẽ về mặt toán học để **đặc tả, thiết kế và xác minh** hệ thống phần mềm và phần cứng.

Phương pháp hình thức

- Các phương pháp thường thấy có thể kể đến như:
 - Đặc tả hình thức.
 - Kiểm chứng hình thức.
 - Kiểm thử dựa trên đặc tính.

Phương pháp hình thức

- Một số ứng dụng thực tế:



Vi nhân hệ điều hành hiệu năng cao



CPU

Cảm ơn thầy và các bạn
đã quan tâm theo dõi!