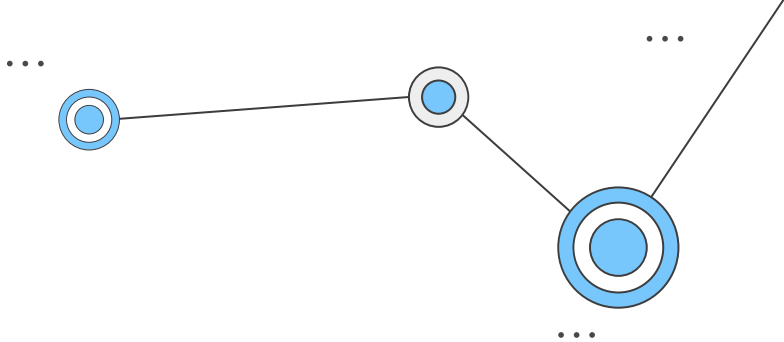
Các phương pháp hình thức (Formal Methods)

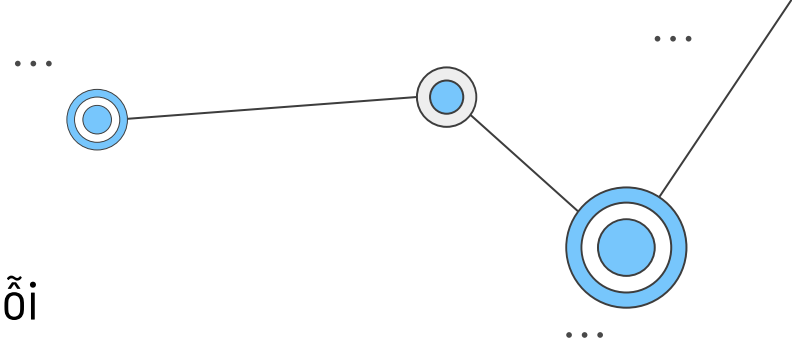
TS. Lê Khánh Trình

Thông tin giảng viên



- TS. Lê Khánh Trình
- Bộ môn Công nghệ phần mềm
- trinhlk@vnu.edu.vn

Thông tin môn học

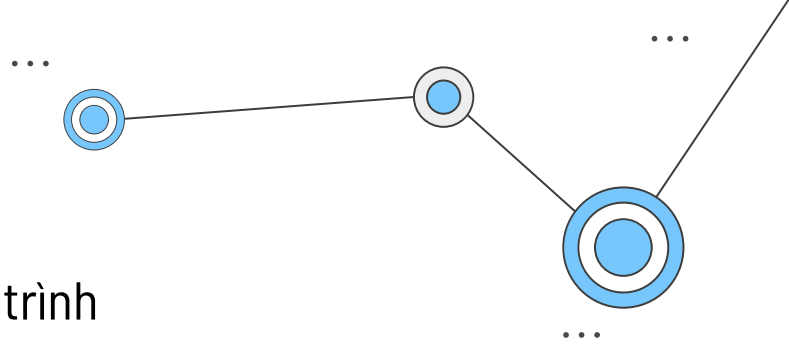


Sử dụng toán học để viết chương trình không có lỗi

- Trình biên dịch: lỗi cú pháp, lỗi về type
- Các phương pháp hình thức:
  - Có lỗi logic không? Lỗi đó là gì và ở đâu?
  - Xác định lỗi khi lập trình thay vì đợi báo cáo từ người dùng/kiểm thử

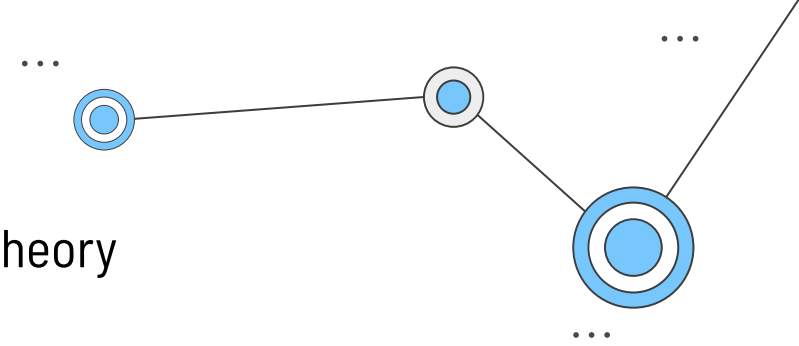
→ Đặc biệt quan trọng với những hệ thống ảnh hưởng tới tính mạng con người

Nội dung môn học



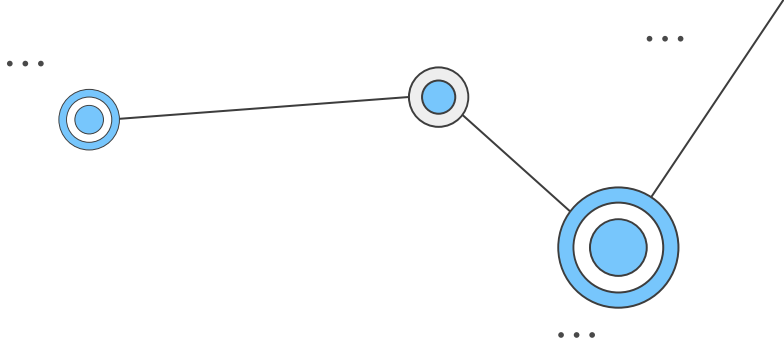
- Giới thiệu logic cơ bản sử dụng để hỗ trợ lập trình
- Đặc tả hình thức và cách làm mịn (refine) chúng thành chương trình
- Ở mỗi bước làm mịn, chứng minh một định lý nhỏ (để chắc chắn rằng bước đó thực hiện đúng)
- Nếu mỗi bước đúng ⟶ chương trình đúng

Nội dung môn học



- Binary Theory, Number Theory, Character Theory
- Collections: Bunches và Sets
- Sequences: String và Lists
- Function, Function Fine Points
- Quantifiers
- Specification
- Refinement & Programs
- Time Calculation
- Scope, Data Structures, Control Structures

Quản lý lớp học



- Lớp trưởng: Phạm Thị Kim Huệ
- Email: 25028019@vnu.edu.vn
- SĐT: 0869835976

Ứng dụng (applications)?

- Các giao thức liên lạc (communication protocols)
- Emails, ghi chú, v.v.
- Trình biên dịch (compilers)
- Các hệ thống chú trọng an toàn (safety-critical systems):
  - y tế
  - xe tự lái
  - điều khiển tên lửa
  - v.v
- Hệ thống liên lạc giữa máy bay - kiểm soát không lưu

Ứng dụng (applications)?

- Các giao thức liên lạc (communication protocols)
- Emails, ghi chú, v.v.
- Trình biên dịch (compilers)
- Các hệ thống chú trọng an toàn (safety-critical systems):
  - y tế
  - xe tự lái
  - điều khiển tên lửa
  - v.v
- Hệ thống liên lạc giữa máy bay - kiểm soát không lưu

Phần mềm nào cũng phải chính xác

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý
- Các biểu thức toán (math expressions)

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các biểu thức toán (math expressions)

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các bt toán (math expressions) —> lý thuyết lập trình —> chương trình đúng

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các bt toán (math expressions) —> lý thuyết lập trình —> chương trình đúng

Tại sao lại là lý thuyết?

Lý thuyết hình thức (formal theory)

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các bt toán (math expressions) —> lý thuyết lập trình —> chương trình đúng

Tại sao lại là lý thuyết?

Lý thuyết hình thức (formal theory)
= hình thức hoá (formalization) ~ các quy tắc để chứng minh, tính toán, vận dụng

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các bt toán (math expressions) —> lý thuyết lập trình —> chương trình đúng

Tại sao lại là lý thuyết? —> chứng minh

Lý thuyết hình thức (formal theory)

= hình thức hoá ~ các quy tắc để chứng minh, tính toán, vận dụng

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các bt toán (math expressions) —> lý thuyết lập trình —> chương trình đúng

Tại sao lại là lý thuyết? —> chứng minh, tính toán

Lý thuyết hình thức (formal theory)
= hình thức hoá ~ các quy tắc để chứng minh, tính toán, vận dụng

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các bt toán (math expressions) —> lý thuyết lập trình —> chương trình đúng

Tại sao lại là lý thuyết? —> chứng minh, tính toán, chính xác

Lý thuyết hình thức (formal theory)
= hình thức hoá ~ các quy tắc để chứng minh, tính toán, vận dụng

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các bt toán (math expressions) —> lý thuyết lập trình —> chương trình đúng

Tại sao lại là lý thuyết? —> chứng minh, tính toán, chính xác, hiểu

Lý thuyết hình thức (formal theory)

= hình thức hoá ~ các quy tắc để chứng minh, tính toán, vận dụng

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các bt toán (math expressions) —> lý thuyết lập trình —> chương trình đúng

Tại sao lại là lý thuyết? —> chứng minh, tính toán, chính xác, hiểu

Lý thuyết hình thức (formal theory)
= hình thức hoá ~ các quy tắc để chứng minh, tính toán, vận dụng

hình thức **!=** cẩn thận, chi tiết
không hình thức **!=** cẩu thả, sơ sài

Chương trình (programs) là ...

- Các lệnh (commands) cho máy tính xử lý —> thực thi
- Các bt toán (math expressions) —> lý thuyết lập trình —> chương trình đúng

Tại sao lại là lý thuyết? —> chứng minh, tính toán, chính xác, hiểu

Lý thuyết hình thức (formal theory)
= hình thức hoá ~ các quy tắc để chứng minh, tính toán, vận dụng

hình thức **!=** cẩn thận, chi tiết **—>** Sử dụng biểu thức toán học
không hình thức **!=** cẩu thả, sơ sài **—>** Sử dụng ngôn ngữ tự nhiên

Xây dựng chương trình

- Bắt đầu là **không hình thức - informal** (thảo luận)

Xây dựng chương trình

- Bắt đầu là **không hình thức - informal** (thảo luận)
- Kết thúc là **hình thức - formal** (lập trình)

Xây dựng chương trình

- Bắt đầu là **không hình thức - informal** (thảo luận)
- Kết thúc là **hình thức - formal** (lập trình)

Tiếp theo, kiểm thử (test)

Xây dựng chương trình

- Bắt đầu là **không hình thức - informal** (thảo luận)
- Kết thúc là **hình thức - formal** (lập trình)

Tiếp theo, kiểm thử (test), nhưng:

- Làm sao bạn biết chương trình có hoạt động không?
- Có những inputs khác bạn chưa kiểm thử thì sao?

Xây dựng chương trình

- Bắt đầu là **không hình thức - informal** (thảo luận)
- Kết thúc là **hình thức - formal** (lập trình)

Tiếp theo, kiểm thử (test), nhưng:

- Làm sao bạn biết chương trình có hoạt động không?
- Có những inputs khác bạn chưa kiểm thử thì sao?

Việc chứng minh (prove) chỉ ra được liệu chương trình có đúng với mọi inputs

Xây dựng chương trình

- Bắt đầu là **không hình thức - informal** (thảo luận)
- Kết thúc là **hình thức - formal** (lập trình)

Tiếp theo, kiểm thử (test), nhưng:

- Làm sao bạn biết chương trình có hoạt động không?
- Có những inputs khác bạn chưa kiểm thử thì sao?

Việc chứng minh (prove) chỉ ra được liệu chương trình có đúng với mọi inputs

Chứng minh/kiểm chứng (verification) sau quá trình phát triển

Xây dựng chương trình

- Bắt đầu là **không hình thức - informal** (thảo luận)
- Kết thúc là **hình thức - formal** (lập trình)

Tiếp theo, kiểm thử (test), nhưng:

- Làm sao bạn biết chương trình có hoạt động không?
- Có những inputs khác bạn chưa kiểm thử thì sao?

Việc chứng minh (prove) chỉ ra được liệu chương trình có đúng với mọi inputs

~~Chứng minh/kiểm chứng (verification) sau quá trình phát triển~~

Trong quá trình phát triển, có chứng minh ở mỗi bước

Xây dựng chương trình

- Bắt đầu là **không hình thức - informal** (thảo luận)
- Kết thúc là **hình thức - formal** (lập trình)

Tiếp theo, kiểm thử (test), nhưng:

- Làm sao bạn biết chương trình có hoạt động không?
- Có những inputs khác bạn chưa kiểm thử thì sao?

Việc chứng minh (prove) chỉ ra được liệu chương trình có đúng với mọi inputs

~~Chứng minh/kiểm chứng (verification) sau quá trình phát triển~~
Trong quá trình phát triển, có chứng minh ở mỗi bước
Sửa đổi chương trình, có chứng minh

Một số lý thuyết

- Bộ ba Hoare (Hoare triples): {P}S{R}

Một số lý thuyết

- Bộ ba Hoare (Hoare triples): {P}S{R}
- Điều kiện tiên quyết yếu nhất của Dijkstra: wp(S,R)
- Vienna Development Method (VDM)
- Z và B
- Lô-gic thời gian (temporal logic): □ ◊
- Xử lý đại số (process algebras): CSP, CCS, mu-calculus, pi-calculus, v.v.
- truy vết sự kiện (event traces)

Một số lý thuyết

- Bộ ba Hoare (Hoare triples): {P}S{R}
- Điều kiện tiên quyết yếu nhất của Dijkstra: wp(S,R)
- Vienna Development Method (VDM)
- Z và B
- Lô-gic thời gian (temporal logic): □ ◊
- Xử lý đại số (process algebras): CSP, CCS, mu-calculus, pi-calculus, v.v.
- truy vết sự kiện (event traces)
- **Kiểm tra mô hình (model checking)**

Một số lý thuyết

- Bộ ba Hoare (Hoare triples): {P}S{R}
- Điều kiện tiên quyết yếu nhất của Dijkstra: wp(S,R)
- Vienna Development Method (VDM)
- Z và B
- Lô-gic thời gian (temporal logic): □ ◊
- Xử lý đại số (process algebras): CSP, CCS, mu-calculus, pi-calculus, v.v.
- truy vết sự kiện (event traces)
- **Kiểm tra mô hình (model checking)**
  - kiểm thử tự động toàn diện (exhaustive auto. testing)

Một số lý thuyết

- Bộ ba Hoare (Hoare triples): {P}S{R}
- Điều kiện tiên quyết yếu nhất của Dijkstra: wp(S,R)
- Vienna Development Method (VDM)
- Z và B
- Lô-gic thời gian (temporal logic): □ ◊
- Xử lý đại số (process algebras): CSP, CCS, mu-calculus, pi-calculus, v.v.
- truy vết sự kiện (event traces)
- **Kiểm tra mô hình (model checking)**
  - kiểm thử tự động toàn diện (exhaustive auto. testing)
  - 10^{60} (~2^{200}) trạng thái

Một số lý thuyết

- Bộ ba Hoare (Hoare triples): {P}S{R}
- Điều kiện tiên quyết yếu nhất của Dijkstra: wp(S,R)
- Vienna Development Method (VDM)
- Z và B
- Lô-gic thời gian (temporal logic): □ ◊
- Xử lý đại số (process algebras): CSP, CCS, mu-calculus, pi-calculus, v.v.
- truy vết sự kiện (event traces)
- **Kiểm tra mô hình (model checking)**
  - kiểm thử tự động toàn diện (exhaustive auto. testing)
  - 10^{60} (~2^{200}) trạng thái ~ 200 bit

Một số lý thuyết

- Bộ ba Hoare (Hoare triples): {P}S{R}
- Điều kiện tiên quyết yếu nhất của Dijkstra: wp(S,R)
- Vienna Development Method (VDM)
- Z và B
- Lô-gic thời gian (temporal logic): □ ◊
- Xử lý đại số (process algebras): CSP, CCS, mu-calculus, pi-calculus, v.v.
- truy vết sự kiện (event traces)
- **Kiểm tra mô hình (model checking)**
  - kiểm thử tự động toàn diện (exhaustive auto. testing)
  - 10^{60} (~2^{200}) trạng thái ~ 200 bit ~ 6 biến

Một số lý thuyết

- Bộ ba Hoare (Hoare triples): {P}S{R}
- Điều kiện tiên quyết yếu nhất của Dijkstra: wp(S,R)
- Vienna Development Method (VDM)
- Z và B
- Lô-gic thời gian (temporal logic): □ ◊
- Xử lý đại số (process algebras): CSP, CCS, mu-calculus, pi-calculus, v.v.
- truy vết sự kiện (event traces)
- **Kiểm tra mô hình (model checking)**
  - kiểm thử tự động toàn diện (exhaustive auto. testing)
  - 10^{60} (~2^{200}) trạng thái ~ 200 bit ~ 6 biến
  - trừu tượng hoá, chứng minh (không tự động)

Một số lý thuyết

- Bộ ba Hoare (Hoare triples): {P}S{R}
- Điều kiện tiên quyết yếu nhất của Dijkstra: wp(S,R)
- Vienna Development Method (VDM)
- Z và B
- Lô-gic thời gian (temporal logic): □ ◊
- Xử lý đại số (process algebras): CSP, CCS, mu-calculus, pi-calculus, v.v.
- truy vết sự kiện (event traces)
- **Kiểm tra mô hình (model checking)**
  - kiểm thử tự động toàn diện (exhaustive auto. testing)
  - 10^{60} (~2^{200}) trạng thái ~ 200 bit ~ 6 biến
  - trừu tượng hoá, chứng minh (không tự động)

Lý thuyết trong khoá học này

- Đơn giản hơn
  - chỉ là các biểu thức nhị phân (binary expressions)

Lý thuyết trong khoá học này

- Đơn giản hơn
  - chỉ là các biểu thức nhị phân (binary expressions)
- Tổng quát hơn, bao gồm:
  - tính toán giới hạn (terminating) và vô hạn (non-terminating)

Lý thuyết trong khoá học này

- Đơn giản hơn
  - chỉ là các biểu thức nhị phân (binary expressions)
- Tổng quát hơn, bao gồm:
  - tính toán giới hạn (terminating) và vô hạn (non-terminating)
  - tính toán tuần tự (sequential) và song song (parallel)

Lý thuyết trong khoá học này

- Đơn giản hơn
  - chỉ là các biểu thức nhị phân (binary expressions)
- Tổng quát hơn, bao gồm:
  - tính toán giới hạn (terminating) và vô hạn (non-terminating)
  - tính toán tuần tự (sequential) và song song (parallel)
  - tính toán độc lập (stand-alone) và tương tác (interactive)

Lý thuyết trong khoá học này

- Đơn giản hơn
  - chỉ là các biểu thức nhị phân (binary expressions)
- Tổng quát hơn, bao gồm:
  - tính toán giới hạn (terminating) và vô hạn (non-terminating)
  - tính toán tuần tự (sequential) và song song (parallel)
  - tính toán độc lập (stand-alone) và tương tác (interactive)
  - có giới thời gian, không gian (time and space bounds); và thời gian thực (real-time)

Lý thuyết trong khoá học này

- Đơn giản hơn
  - chỉ là các biểu thức nhị phân (binary expressions)
- Tổng quát hơn, bao gồm:
  - tính toán giới hạn (terminating) và vô hạn (non-terminating)
  - tính toán tuần tự (sequential) và song song (parallel)
  - tính toán độc lập (stand-alone) và tương tác (interactive)
  - có giới thời gian, không gian (time and space bounds); và thời gian thực (real-time)
  - tính toán xác suất (probabilistic)

Lý thuyết trong khoá học này

- Đơn giản hơn
  - chỉ là các biểu thức nhị phân (binary expressions)
- Tổng quát hơn, bao gồm:
  - tính toán giới hạn (terminating) và vô hạn (non-terminating)
  - tính toán tuần tự (sequential) và song song (parallel)
  - tính toán độc lập (stand-alone) và tương tác (interactive)
  - có giới thời gian, không gian (time and space bounds); và thời gian thực (real-time)
  - tính toán xác suất (probabilistic)
- **Yêu cầu môn học**
  - biết lập trình cơ bản: gán, rẽ nhánh

Lý thuyết trong khoá học này

- Đơn giản hơn
  - chỉ là các biểu thức nhị phân (binary expressions)
- Tổng quát hơn, bao gồm:
  - tính toán giới hạn (terminating) và vô hạn (non-terminating)
  - tính toán tuần tự (sequential) và song song (parallel)
  - tính toán độc lập (stand-alone) và tương tác (interactive)
  - có giới thời gian, không gian (time and space bounds); và thời gian thực (real-time)
  - tính toán xác suất (probabilistic)
- **Yêu cầu môn học**
  - biết lập trình cơ bản: gán, rẽ nhánh

Lý thuyết trong khoá học này

- Đơn giản hơn
  - chỉ là các biểu thức nhị phân (binary expressions)
- Tổng quát hơn, bao gồm:
  - tính toán giới hạn (terminating) và vô hạn (non-terminating)
  - tính toán tuần tự (sequential) và song song (parallel)
  - tính toán độc lập (stand-alone) và tương tác (interactive)
  - có giới thời gian, không gian (time and space bounds); và thời gian thực (real-time)
  - tính toán xác suất (probabilistic)
- **Yêu cầu môn học**
  - biết lập trình cơ bản: gán, rẽ nhánh

Bài tập 1

Cho 3 phát biểu sau:

1. Có chính xác một phát biểu sai
2. Có chính xác hai phát biểu sai
3. Cả ba phát biểu đều sai

Phát biểu nào là đúng và phát biểu nào là sai?

Bài tập 2

Có 4 lá bài đặt trên bàn mặt trên mỗi lá là các ký tự tương ứng: J, K, 5, 3.

Mỗi lá bài có một mặt số và một mặt chữ.

Phải lật tối thiểu những lá bài nào để chứng minh khẳng định rằng:

- **Mọi lá bài J sẽ có số 3 ở mặt còn lại**

Vì sao?

Bài tập 2 – gợi ý

$(t0=\text{“J”} \Rightarrow b0=\text{“3”}) \land (b0=\text{“J”} \Rightarrow t0=\text{“3”})$

$\land (t1=\text{“J”} \Rightarrow b1=\text{“3”}) \land (b1=\text{“J”} \Rightarrow t1=\text{“3”})$

$\land (t2=\text{“J”} \Rightarrow b2=\text{“3”}) \land (b2=\text{“J”} \Rightarrow t2=\text{“3”})$

$\land (t3=\text{“J”} \Rightarrow b3=\text{“3”}) \land (b3=\text{“J”} \Rightarrow t3=\text{“3”})$

$= (\top \Rightarrow b0=\text{“3”}) \land (\bot \Rightarrow \bot)$ **($\bot$: False, $\top$: True)**

$\land (\bot \Rightarrow b1=\text{“3”}) \land (\bot \Rightarrow \bot)$

$\land (\bot \Rightarrow \bot) \land (b2=\text{“J”} \Rightarrow \bot)$

$\land (\bot \Rightarrow \bot) \land (b3=\text{“J”} \Rightarrow \top)$

Bài tập 2 – gợi ý

Sử dụng các luật sau

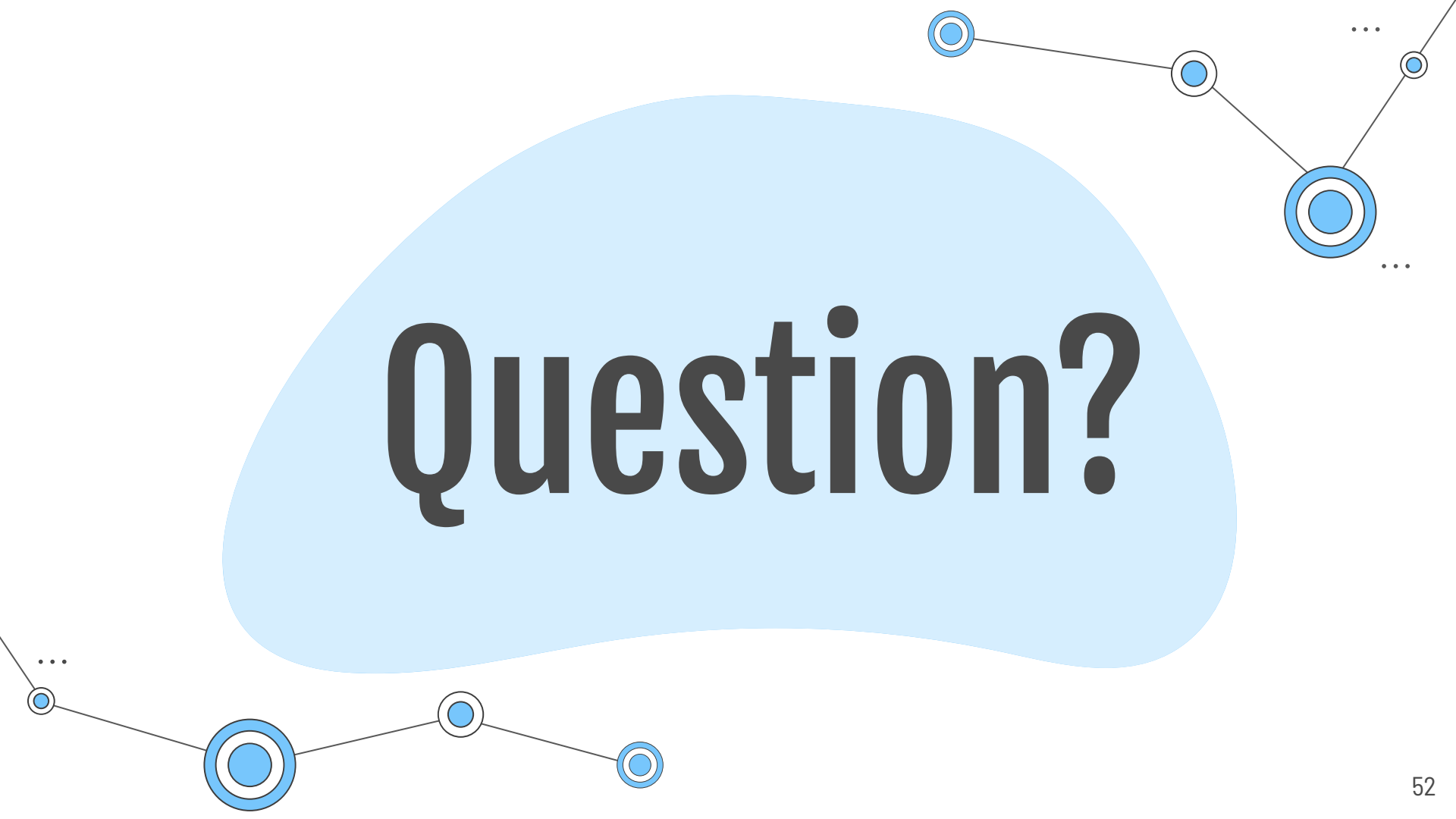
$(\top \Rightarrow a) = a, (a \Rightarrow \bot) = \neg a, (\bot \Rightarrow a) = \top$

và

$(\top \wedge a) = a, (a \wedge \top) = a$

BTVN

- **Tìm hiểu và lấy ví dụ sử dụng các lý thuyết sau:**
  - Bộ ba Hoare (Hoare triples): {P}S{R}
  - Điều kiện tiên quyết yếu nhất của Dijkstra: wp(S,R)
  - Vienna Development Method (VDM)
  - Z và B
  - Lô-gic thời gian (temporal logic): □ ◊
  - Xử lý đại số (process algebras): CSP, CCS, mu-calculus, pi-calculus, v.v.
  - truy vết sự kiện (event traces)
  - Kiểm tra mô hình (model checking)


Question?