Các kỹ thuật đặc tả (4)

Nguyễn Thanh Bình
Khoa Công nghệ Thông tin
Trường Đại học Bách khoa
Đại học Đà Nẵng

Nội dung

- Khái niệm đặc tả
- Tại sao phải đặc tả ?
- Phân loại các kỹ thuật đặc tả
- Các kỹ thuật đặc tả

Khái niệm đặc tả

- Đặc tả (specification)
  - định nghĩa một hệ thống, mô-đun hay một sản phẩm cần phải **làm cái gì**
  - không mô tả nó phải làm như thế nào
  - mô tả những **tính chất của vấn đề** đặt ra
  - không mô tả những tính chất của giải pháp cho vấn đề đó

3

Khái niệm đặc tả

- Đặc tả là hoạt động được tiến hành trong các giai đoạn khác nhau của tiến trình phần mềm:
  - Đặc tả yêu cầu (requirement specification)
    - sự thống nhất giữa những người sử dụng tương lai và những người thiết kế
  - Đặc tả kiến trúc hệ thống (system architect specification)
    - sự thống nhất giữa những người thiết kế và những người cài đặt
  - Đặc tả môđun (module specification)
    - sự thống nhất giữa những người lập trình cài đặt mô-đun và những người lập trình sử dụng mô-đun

4

Tại sao phải đặc tả ?

- Hợp đồng
  - sự thống nhất giữa người sử dụng và người phát triển sản phẩm
- Hợp thức hóa
  - sản phẩm làm ra phải thực hiện chính xác những gì mong muốn
- Trao đổi
  - giữa người sử dụng và người phát triển
  - giữa những người phát triển
- Tái sử dụng

5

Phân loại các kỹ thuật đặc tả

- Đặc tả phi hình thức (informal)
  - ngôn ngữ tự nhiên tự do
  - ngôn ngữ tự nhiên có cấu trúc
  - các kí hiệu đồ họa
- Đặc tả nữa hình thức (semi-informal)
  - trộn lẫn cả ngôn ngữ tự nhiên, các kí hiệu toán học và các kí hiệu đồ họa
- Đặc tả hình thức (formal)
  - kí hiệu toán học
    - ngôn ngữ đặc tả
    - ngôn ngữ lập trình

6

Đặc tả hình thức hay không hình thức ?

- Đặc tả hình thức
  - chính xác (toán học)
  - hợp thức hóa hình thức (công cụ hóa)
  - công cụ trao đổi: khó đọc, khó hiểu
  - khó sử dụng
- Đặc tả không hình thức
  - dễ hiểu, dễ sử dụng
  - mềm dẻo
  - thiếu sự chính xác
  - nhập nhằng

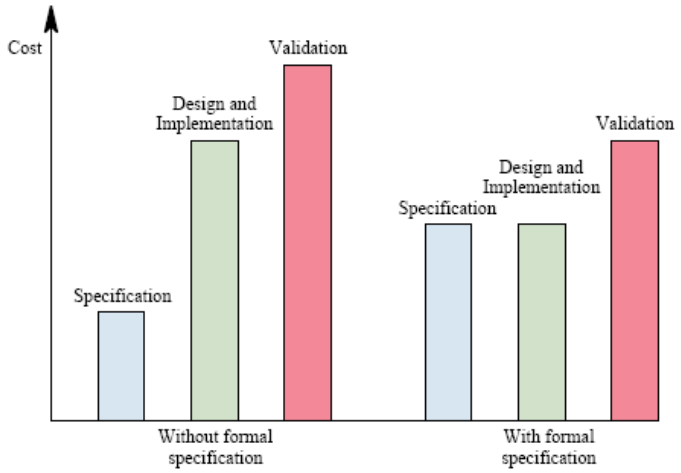
7

Ứng dụng đặc tả hình thức

- ứng dụng trong các giai đoạn sớm của tiến trình phát triển
- hạn chế lỗi trong phát triển phần mềm
- ứng dụng chủ yếu trong phát triển các hệ thống “quan trọng” (critical systems)
  - hệ thống điều khiển
  - hệ thống nhúng
  - hệ thống thời gian thực

8

Chi phí phát triển khi sử dụng đặc tả hình thức



Các kỹ thuật đặc tả

- Trình bày một số kỹ thuật
  - Máy trạng thái hữu hạn
  - Mạng Petri
  - Điều kiện trước và sau
  - Kiểu trừu tượng
  - Đặc tả Z

10

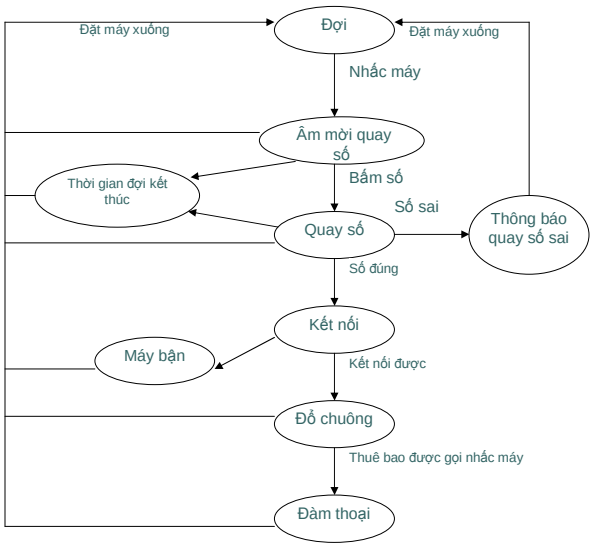
Máy trạng thái hữu hạn (state machine)

- mô tả các luồng điều khiển
- biểu diễn dạng đồ thị
- bao gồm
  - tập hợp các trạng thái S (các nút của đồ thị)
  - tập hợp các dữ liệu vào I (các nhãn của các cung)
  - tập hợp các chuyển tiếp T : S x I $\rightarrow$ S (các cung có hướng của đồ thị)
    - khi có một dữ liệu vào, một trạng thái chuyển sang một trạng thái khác

11

Máy trạng thái hữu hạn

- Ví dụ 1



12

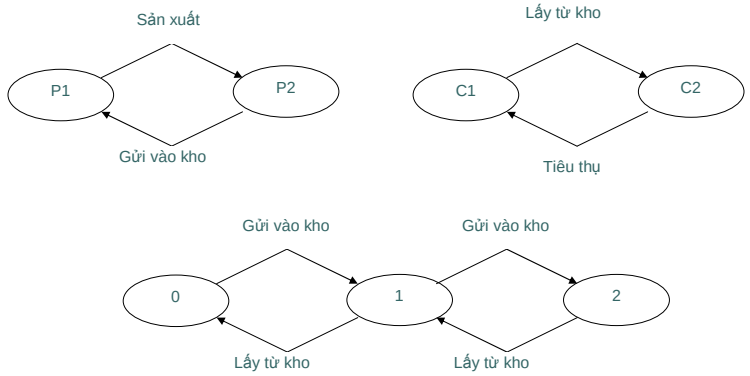
Máy trạng thái hữu hạn

- Ví dụ 2
  - Hệ thống cần mô tả bao gồm một nhà sản xuất, một nhà tiêu thụ và một kho hàng chỉ chứa được nhiều nhất 2 sản phẩm
  - Nhà sản xuất có 2 trạng thái
    - P1: không sản xuất
    - P2: đang sản xuất
  - Nhà tiêu thụ có 2 trạng thái
    - C1: có sản phẩm để tiêu thụ
    - C2: không có sản phẩm để tiêu thụ
  - Nhà kho có 3 trạng thái
    - chứa 0 sản phẩm
    - chứa 1 sản phẩm
    - chứa 2 sản phẩm

13

Máy trạng thái hữu hạn

- Giải pháp 1: mô tả tách rời các thành phần



14

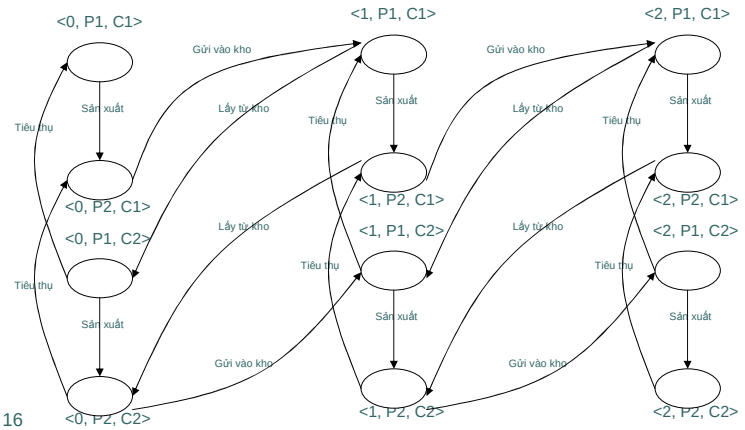
Máy trạng thái hữu hạn

- Giải pháp 1
  - không mô tả được sự hoạt động hệ thống
  - cần mô tả sự hoạt động kết hợp các thành phần của hệ thống

15

Máy trạng thái hữu hạn

- Giải pháp 2: mô tả kết hợp các thành phần



16

Máy trạng thái hữu hạn

- Giải pháp 2
  - mô tả được hoạt động của hệ thống
  - số trạng thái lớn
  - biểu diễn hệ thống phức tạp
  - hạn chế khi đặc tả những hệ thống không đồng bộ
    - các thành phần của hệ thống hoạt động song song hoặc cạnh tranh

17

Mạng Petri
(Petri nets)

- thích hợp để mô tả các hệ thống không đồng bộ với những hoạt động đồng thời
- mô tả luồng điều khiển của hệ thống
- đề xuất từ năm 1962 bởi Carl Adam
- Có hai loại
  - mạng Petri (cổ điển)
  - mạng Petri mở rộng

18

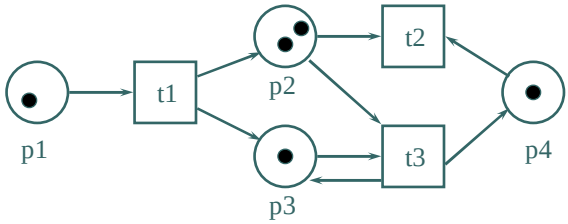
Mạng Petri

- Gồm các phần tử
  - một tập hợp hữu hạn các *nút* (○)
  - một tập hợp hữu hạn các *chuyển tiếp* (□)
  - một tập hợp hữu hạn các *cung* (→)
    - các cung nối các nút với các chuyển tiếp hoặc ngược lại
  - mỗi nút có thể chứa một hoặc nhiều *thẻ* (●)

19

Mạng Petri

- Ví dụ



20

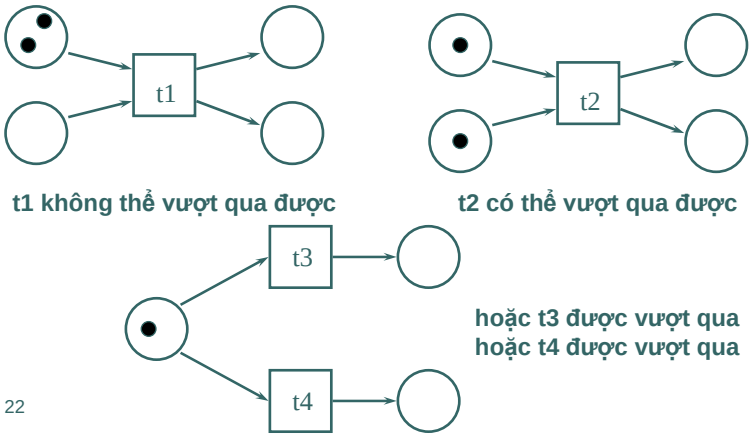
Mạng Petri

- Mạng Petri được định nghĩa bởi sự đánh dấu các nút của nó
- Việc đánh dấu các nút được tiến hành theo nguyên tắc sau:
  - mỗi chuyển tiếp có các nút vào và các nút ra
  - nếu tất cả các nút vào của một chuyển tiếp có ít nhất một thẻ, thì chuyển tiếp này là có thể *vượt qua* được,
  - nếu chuyển tiếp này được thực hiện thì tất cả các nút vào của chuyển tiếp sẽ bị lấy đi một thẻ, và một thẻ sẽ được thêm vào tất cả các nút ra của chuyển tiếp
  - nếu nhiều chuyển tiếp là có thể vượt qua thì chọn chuyển tiếp nào cũng được

21

Mạng Petri

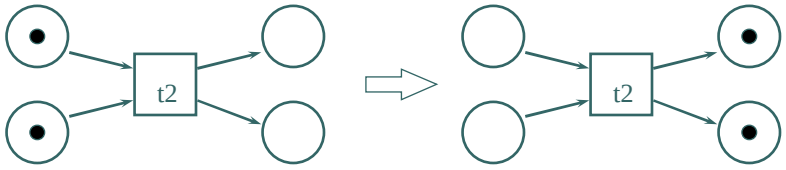
- Ví dụ



22

Mạng Petri

- Ví dụ

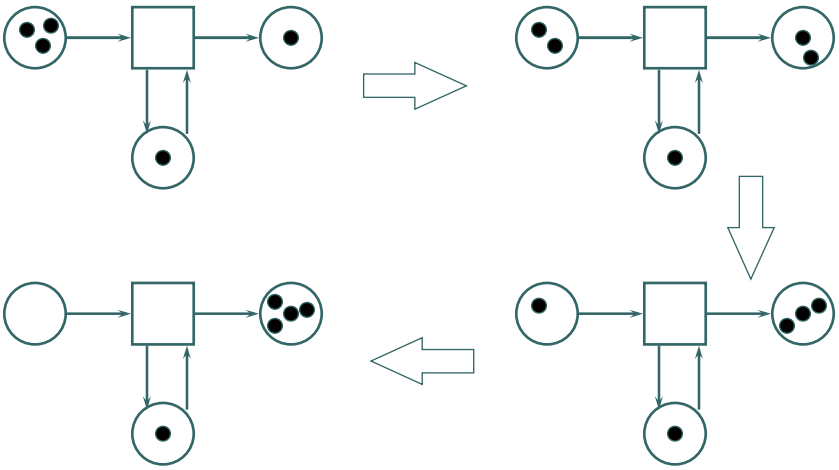


khi t2 được vượt qua

23

Mạng Petri

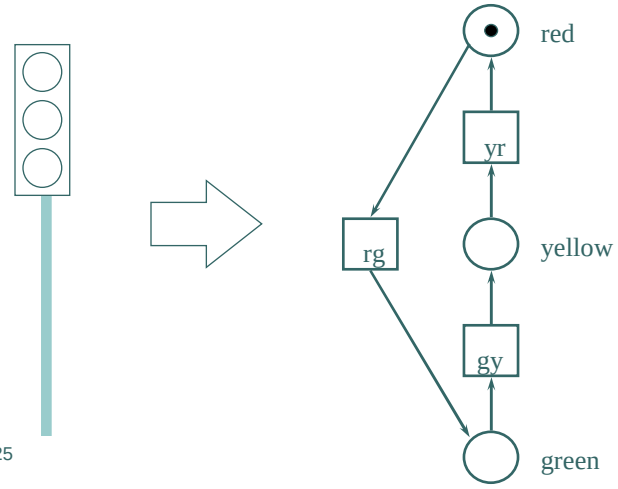
Ví dụ



24

Mạng Petri

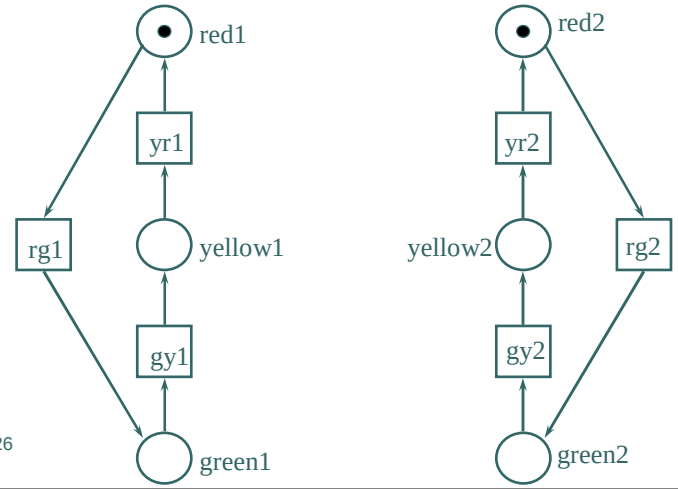
- Ví dụ 1: mô tả hoạt động của đèn giao thông



25

Mạng Petri

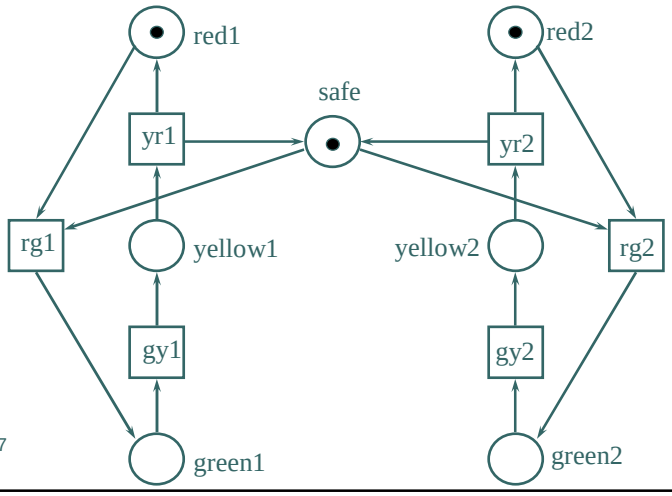
- Ví dụ 1: mô tả hoạt động của 2 đèn giao thông



26

Mạng Petri

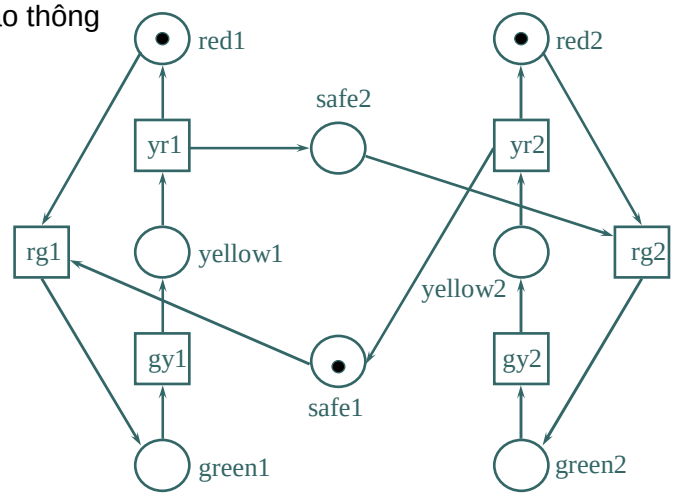
- Ví dụ 1: mô tả hoạt động an toàn của 2 đèn giao thông



27

Mạng Petri

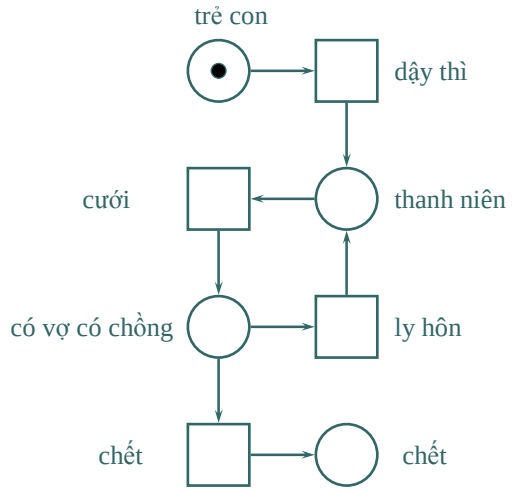
- Ví dụ 1: mô tả hoạt động an toàn và hợp lý của 2 đèn giao thông



28

Mạng Petri

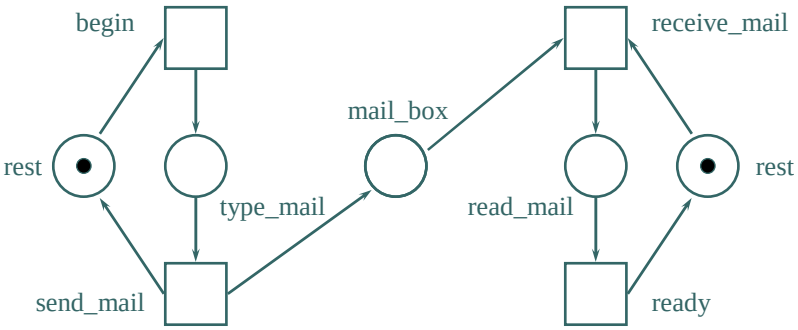
- Ví dụ 2: mô tả chu kỳ sống của một người



29

Mạng Petri

- Ví dụ 3: viết thư và đọc thư

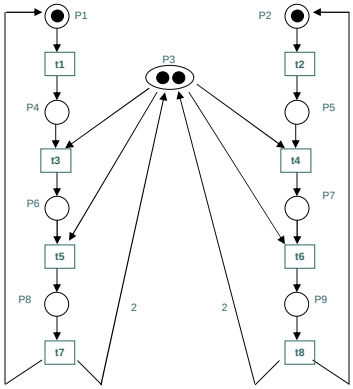


Mô tả trường hợp 1 người viết và 2 người đọc ?
Mô tả trường hợp hộp thư nhận chỉ chứa nhiều nhất 3 thư ?

30

Mạng Petri

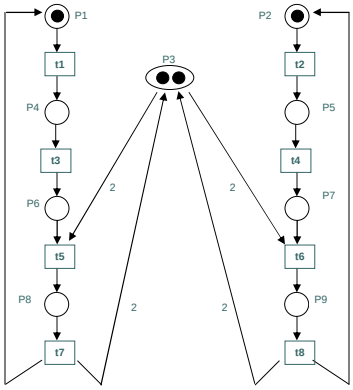
- Ví dụ 4: tình huống nghẽn (dead-lock)



31

Mạng Petri

- Ví dụ 4: giải pháp chống nghẽn



32

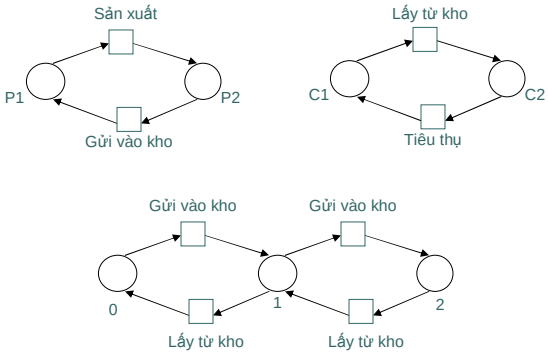
Mạng Petri

- Ví dụ 5
  - Hệ thống cần mô tả bao gồm một nhà sản xuất, một nhà tiêu thụ và một kho hàng chỉ chứa được nhiều nhất 2 sản phẩm
  - Nhà sản xuất có 2 trạng thái
    - P1: không sản xuất
    - P2: đang sản xuất
  - Nhà tiêu thụ có 2 trạng thái
    - C1: có sản phẩm để tiêu thụ
    - C2: không có sản phẩm để tiêu thụ
  - Nhà kho có 3 trạng thái
    - chứa 0 sản phẩm
    - chứa 1 sản phẩm
    - chứa 2 sản phẩm

33

Mạng Petri

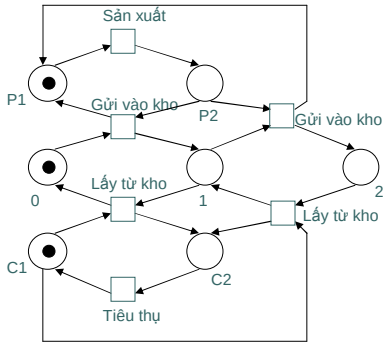
- Ví dụ 5: mô tả tách rời mỗi thành phần



34

Mạng Petri

- Ví dụ 5: mô tả kết hợp các thành phần



35

Điều kiện trước và sau (pre/post condition)

- được dùng để đặc tả các hàm hoặc mô-đun
- đặc tả các tính chất của dữ liệu trước và sau khi thực hiện hàm
  - **pre-condiition**: đặc tả các ràng buộc trên các tham số **trước** khi hàm được thực thi
  - **post-condition**: đặc tả các ràng buộc trên các tham số **sau** khi hàm được thực thi
- có thể sử dụng ngôn ngữ phi hình thức, hình thức hoặc ngôn ngữ lập trình để đặc tả các điều kiện

36

Điều kiện trước và sau

- Ví dụ: đặc tả hàm tìm kiếm

function search (a : danh sách phần tử kiểu K,
size : số phân tử của dánh sách,
e : phần tử kiểu K,
result : Boolean)

pre $\forall i, 1 \le i \le n, a[i] \le a[i+1]$
post result = $(\exists i, 1 \le i \le n, a[i] = e)$

37

Điều kiện trước và sau

- Bài tập: đặc tả các hàm
  1. Sắp xếp một danh sách các số nguyên
  2. Đảo ngược các phần tử của một danh sách
  3. Đếm số phần tử có giá trị *e* trong một danh sách các số nguyên

38

Kiểu trừu tượng
(abstract types)

- Mô tả dữ liệu và các thao tác trên dữ liệu đó ở một mức trừu tượng độc lập với cách cài đặt dữ liệu bởi ngôn ngữ lập trình
- Đặc tả một kiểu trừu tượng gồm:
  - tên của kiểu trừu tượng
    - dùng từ khóa sort
  - khai báo các kiểu trừu tượng đã tồn tại được sử dụng
    - dùng từ khóa imports
  - các thao tác trên trên kiểu trừu tượng
    - dùng từ khóa operations

39

Kiểu trừu tượng

- Ví dụ 1: đặc tả kiểu trừu tượng Boolean

sort Boolean
operations

true	**: → Boolean**
false	**: → Boolean**
¬ _	**: Boolean → Boolean**
_ ∧ _	**: Boolean x Boolean → Boolean**
_ ∨ _	**: Boolean x Boolean → Boolean**

một thao tác không có tham số là một hằng số
một giá trị của kiểu trừu tượng định nghĩa được biểu diễn bởi kí tự "_"

40

Kiểu trừu tượng

- Ví dụ 2: đặc tả kiểu trừu tượng Vector

sort Boolean
operations

true	: → Boolean
false	: → Boolean
¬ _	: Boolean → Boolean
_ ∧ _	: Boolean x Boolean → Boolean
_ ∨ _	: Boolean x Boolean → Boolean

một thao tác không có tham số là một hằng số
một giá trị của kiểu trừu tượng định nghĩa được biểu diễn bởi kí tự "_"

41

Kiểu trừu tượng

- Ví dụ 2: đặc tả kiểu trừu tượng Vector

sort Vector
imports Integer, Element, Boolean
operations

vect	: Integer x Integer → Vector
init	: Vector x Integer → Boolean
ith	: Vector x Integer → Element
change-ith	: Vector x Integer x Element → Vector
supborder	: Vector → Integer
infborder	: Vector → Integer

42

Kiểu trừu tượng

- Ví dụ 2: đặc tả kiểu trừu tượng Vector
  - các thao tác trên kiểu chỉ được định nghĩa mà không chỉ ra ngữ nghĩa của nó
    - tức là ý nghĩa của thao tác
  - sử dụng các *tiên đề* để định nghĩa ngữ nghĩa của các thao tác
    - dùng từ khóa axioms
  - định nghĩa các ràng buộc mà một thao tác được định nghĩa
    - dùng từ khóa precondition

43

Kiểu trừu tượng

- Ví dụ 2: đặc tả kiểu trừu tượng Vector

precondition
ith(v, i) **is-defined-ifonlyif**
infborder(v) ≤ i ≤ *supborder*(v) & *init*(v,i) = true
axioms
infborder(v) ≤ i ≤ *supborder*(v) ⇒ *ith*(*change-ith*(v, i, e), i) = e
infborder(v) ≤ i ≤ *supborder*(v) & *infborder*(v) ≤ j ≤ *supborder*(v) & i ≠ j ⇒
ith(*change-ith*(v, i, e), j) = *ith*(v, j)
init(*vect*(i, j), k) = false
infborder(v) ≤ i ≤ *supborder*(v) ⇒ *init*(*change-ith*(v, i, e), i) = true
infborder(v) ≤ i ≤ *supborder*(v) & i ≠ j ⇒ *init*(*change-ith*(v, i, e), j) = *init*(v, j)
infborder(*vect*(i, j)) = i
infborder(*change-ith*(v, i, e)) = *infborder*(v)
supborder(*vect*(i, j)) = j
supborder(*change-ith*(v, i, e)) = *supborder*(v)

with
v: Vector; i, j, k: Integer; e: Element

44

Kiểu trừu tượng

- Bài tập
  - Đặc tả kiểu trừu tượng **cây nhị phân**
  - Đặc tả kiểu trừu tượng **tập hợp**