Chương 9

Thời gian thực

Với thuộc tính liveness, chúng ta có thể chỉ định rằng hệ thống cuối cùng phải đáp ứng yêu cầu. Chúng tôi không thể xác định rằng nó phải đáp ứng trong vòng 100 năm tới.
Để chỉ định phản hồi kịp thời, chúng ta phải sử dụng thuộc tính thời gian thực.

Một hệ thống không phản hồi trong vòng đời của chúng ta sẽ không hữu ích lắm, vì vậy chúng ta có thể mong đợi các thông số kỹ thuật thời gian thực sẽ phổ biến. Họ không. Thông số kỹ thuật chính thức thường được sử dụng để mô tả những gì hệ thống thực hiện hơn là thời gian thực hiện nó. Tuy nhiên, một ngày nào đó bạn có thể muốn chỉ định các thuộc tính thời gian thực của hệ thống. Chương này sẽ hướng dẫn bạn cách thực hiện.

9.1 Xem lại đồng hồ giờ

Hãy quay lại đặc tả của chúng ta về đồng hồ giờ đơn giản trong Chương 2, trong đó khẳng định rằng biến hr chu kỳ qua các giá trị từ 1 đến 12. Bây giờ chúng ta thêm yêu cầu rằng đồng hồ phải đúng giờ. Trong nhiều thế kỷ, các nhà khoa học đã biểu diễn hành vi thời gian thực của một hệ thống bằng cách đưa vào một biến, theo truyền thống là t, có giá trị là một số thực biểu thị thời gian. Trạng thái trong đó Hãy nhớ rằng trạng thái được gán- t = 17,51 biểu thị trạng thái của hệ thống tại thời điểm 17,51, có lẽ giá trị tính bằng giây trôi qua kể từ 00:00 UT ngày 1 tháng 1 năm 2000. Trong thông số kỹ thuật TLA+ , được đo bằng các thích sử dụng biến bây giờ hơn là t. Để thuận tiện về mặt ngôn ngữ, tôi thường giả sử rằng đơn vị Tôi tất cả các biến. thời gian là giây, mặc dù chúng ta cũng có thể chọn bất kỳ đơn vị nào khác.

Không giống như các ngành khoa học như vật lý và hóa học, khoa học máy tính nghiên cứu các hệ thống mà hành vi của chúng có thể được mô tả bằng một chuỗi các trạng thái riêng biệt, chứ không phải bằng các trạng thái thay đổi liên tục theo thời gian. Chúng tôi coi việc hiển thị của đồng hồ giờ sẽ thay đổi trực tiếp từ số đọc 12 sang số đọc 1 và bỏ qua tính liên tục của các trạng thái trung gian xảy ra trong màn hình vật lý. Điều này có nghĩa là chúng ta giả vờ rằng sự thay đổi diễn ra tức thời (xảy ra sau 0 giây). Vì vậy, một

đặc điểm kỹ thuật thời gian thực của đồng hồ có thể cho phép bước

giờ = 12 giờ = 1
bây giờ = √ 2,47 bây giờ = √ 2,47

Giá trị của now tăng dần giữa các thay đổi đối với hr . Nếu chúng ta muốn chỉ định mất bao lâu để màn hình thay đổi từ 12 thành 1, thì chúng ta sẽ phải đưa ra một trạng thái trung gian biểu thị một màn hình đang thay đổi—có lẽ bằng cách để hr giả sử một số giá trị trung gian chẳng hạn như 12,5 hoặc bằng cách thêm một Boolean- biến có giá trị chg có giá trị cho biết màn hình có thay đổi hay không. Chúng tôi sẽ không làm điều này nhưng sẽ bằng lòng chỉ định đồng hồ giờ mà chúng tôi coi màn hình sẽ thay đổi ngay lập tức.

Giá trị của now thay đổi giữa các thay đổi đối với hr . Giống như việc chúng ta biểu diễn một đồng hồ hiển thị thay đổi liên tục bằng một biến có giá trị thay đổi theo các bước riêng biệt, chúng ta để giá trị hiện tại thay đổi theo các bước riêng biệt. Hành vi hiện tăng theo gia số femto giây sẽ là mô tả đủ chính xác về thời gian thay đổi liên tục đối với thông số kỹ thuật của đồng hồ giờ của chúng tôi. Trên thực tế, không cần phải chọn bất kỳ mức độ chi tiết cụ thể nào về thời gian; bây giờ chúng ta có thể cho phép tiến lên theo số lượng tùy ý giữa các tích tắc đồng hồ. (Vì giá trị của giờ không thay đổi theo các bước thay đổi hiện tại, yêu cầu đồng hồ giữ đúng thời gian sẽ loại trừ các hành vi hiện thay đổi quá nhiều chỉ trong một bước.)

Đồng hồ giờ của chúng ta phải đáp ứng điều kiện thời gian thực nào? Chúng tôi có thể yêu cầu nó luôn hiển thị thời gian chính xác trong vòng ρ giây, đối với một số thực ρ. Tuy nhiên, đây không phải là yêu cầu điển hình về thời gian thực phát sinh trong các hệ thống thực tế. Thay vào đó, chúng tôi yêu cầu đồng hồ tích tắc khoảng một lần mỗi giờ. Chính xác hơn, chúng tôi yêu cầu khoảng thời gian giữa các tích tắc là một giờ cộng hoặc trừ ρ giây, đối với một số dương ρ nào đó. Tất nhiên, yêu cầu này cho phép thời gian hiển thị trên đồng hồ cuối cùng sẽ khác với thời gian thực tế. Nhưng đó là điều mà đồng hồ thật sẽ làm nếu chúng không được cài đặt lại.

Chúng ta có thể bắt đầu đặc tả đồng hồ thời gian thực từ đầu. Tuy nhiên, chúng tôi vẫn muốn đồng hồ giờ thỏa mãn thông số kỹ thuật HC của mô-đun HourClock (Hình 2.1 trên trang 20). Chúng tôi chỉ muốn thêm một yêu cầu bổ sung về thời gian thực. Vì vậy, chúng ta sẽ viết đặc tả dưới dạng kết hợp của HC và một công thức yêu cầu đồng hồ tích tắc mỗi giờ, cộng hoặc trừ ρ giây. Yêu cầu này là sự kết hợp của hai điều kiện riêng biệt: đồng hồ tích tắc nhiều nhất sau mỗi 3600 ρ giây và ít nhất cứ sau 3600 + ρ giây.

Để chỉ định những yêu cầu này, chúng tôi giới thiệu một biến ghi lại lượng thời gian đã trôi qua kể từ tiếng tích tắc cuối cùng của đồng hồ. Hãy gọi nó là t cho bộ đếm thời gian. Giá trị của t được đặt thành 0 theo một bước đại diện cho tiếng tích tắc của đồng hồ—cụ thể là theo bước HCnxt. Bất kỳ bước nào đại diện cho thời gian trôi qua của s giây sẽ tăng lên t bằng s. Một bước biểu thị sự trôi qua của thời gian nếu nó thay đổi bây giờ, và một bước như vậy biểu thị sự trôi qua của giây bây giờ - bây giờ. Vì vậy, sự thay đổi của bộ đếm thời gian t được mô tả bằng hành động

```
TNext  = t = if HCnxt then 0 else t + (now    now)
```

Chúng ta đặt t ban đầu bằng 0, vì vậy chúng ta coi trạng thái ban đầu là trạng thái mà đồng hồ vừa đi êm tích tắc. Khi đó, đặc tả cách t thay đổi là một công thức khẳng định rằng t ban đầu bằng 0 và mỗi bước là bước TNext, nếu không thì tất cả các biến liên quan sẽ không thay đổi—cụ thể là t, hr và now. Công thức này là ,

```
Bộ định thời = (t = 0) [TNext]t, giờ, bây giờ
```

Yêu cầu đồng hồ tích tắc ít nhất một lần sau mỗi 3600 + ρ giây có nghĩa là luôn xảy ra trường hợp tối đa 3600 + ρ giây đã trôi qua kể từ bước HCnxt cuối cùng. Vì t luôn bằng thời gian trôi qua kể từ bước HCnxt cuối cùng nên yêu cầu này được biểu thị bằng công thức

```
Thời gian tối đa = (t 3600 + ρ)
```

(Vì chúng ta không thể đo thời gian với độ chính xác hoàn hảo nên việc chúng ta sử dụng < hay ≤ trong công thức này không thành vấn đề. Khi chúng ta khái quát hóa từ ví dụ này, sẽ thuận tiện hơn một chút khi sử dụng ≤.)

Yêu cầu đồng hồ tích tắc nhiều nhất là 3600 ρ giây một lần có nghĩa là, bất cứ khi nào bước HCnxt xảy ra, ít nhất 3600 ρ giây đã trôi qua kể từ bước HCnxt trước đó. Điều này gợi ý tình trạng

Trong tổng quát hóa, ≥ sẽ thuận tiện hơn >.

(9.1) (HCnxt (t ≥ 3600 ρ))

Tuy nhiên, (9.1) không phải là công thức TLA hợp pháp vì HCnxt . . . là một hành động (một công thức chứa số nguyên tố) và công thức TLA xác nhận rằng một hành động luôn đúng phải có dạng [A]v . Chúng tôi không quan tâm đến các bước không thay đổi giờ, vì vậy chúng tôi có thể thay thế (9.1) bằng công thức TLA

```
Thời gian tối thiểu = [HCnxt (t ≥ 3600 ρ)]hr
```

Ràng buộc thời gian thực mong muốn trên đồng hồ được thể hiện bằng sự kết hợp của ba công thức sau:

```
HCTime = Bộ đếm thời gian Thời gian tối đa Thời gian tối thiểu
```

Công thức HCTime chứa biến t và thông số kỹ thuật của đồng hồ thời gian thực chỉ mô tả những thay đổi đối với giờ (hiển thị đồng hồ) và bây giờ (thời gian). Vì vậy, chúng ta phải giấu t. Việc ẩn được thể hiện trong TLA+ bằng bộ định lượng tồn tại tạm thời được giới thiệu ở Mục 4.3 (trang 41). Tuy nhiên, như đã giải thích trong phần đó, chúng ta không thể viết đơn giản t : HCTime. Chúng ta phải xác định HCTime trong mô-đun khai báo t và sau đó sử dụng phiên bản tham số hóa của mô-đun đó. Việc này được thực hiện trong Hình 9.1 trên trang 121. Thay vì định nghĩa HCTime trong một mô-đun hoàn toàn riêng biệt, tôi đã định nghĩa nó trong một mô-đun con có tên Bên trong của mô-đun RealTimeHourClock chứa thông số kỹ thuật của đồng hồ giờ thời gian thực. Lưu ý rằng tất cả các ký hiệu được khai báo và xác định trong mô-đun chính

cho đến thời điểm đó có thể được sử dụng trong mô-đun con. Mô-đun con bên trong được khởi tạo trong mô-đun chính bằng câu lệnh

```
I(t) = thể hiện Bên trong
```

Khi đó, t trong HCTime có thể được ẩn bằng cách viết t : I(t)!HCTime.

Công thức HC (t : I(t)!HCTime) mô tả những thay đổi có thể có đối với và liên giá trị của giờ, hệ những thay đổi đó với giá trị hiện tại. Nhưng nó nói rất ít về việc giá trị của hiện tại có thể thay đổi như thế nào. Ví dụ: nó cho phép hành vi sau:

giờ = 11	giờ = 11	giờ = 11	giờ = 11	. . .
bây giờ = 23,5	bây giờ = 23,4	bây giờ = 23,5	bây giờ = 23,4	

Bởi vì thời gian không thể quay ngược lại nên hành vi như vậy không thể hiện khả năng vật lý. Mọi người đều biết rằng thời gian chỉ tăng lên, do đó không cần phải cấm hành vi này nếu mục đích duy nhất trong đặc tả của chúng tôi là mô tả đồng hồ giờ.
Tuy nhiên, một đặc tả cũng phải cho phép chúng ta suy luận về một hệ thống. Nếu đồng hồ tích tắc khoảng một giờ một lần thì nó không thể dừng lại. Tuy nhiên, như hành vi trên cho thấy, bản thân công thức HC (t : I(t)!HCTime) cho phép đồng hồ dừng lại. Để suy ra rằng nó không thể, chúng ta cũng cần nêu rõ hiện tại thay đổi như thế nào.

Chúng tôi xác định một công thức RTnow chỉ định những thay đổi có thể có đối với hiện tại. Công thức này không chỉ định mức độ chi tiết của các thay đổi hiện tại; nó cho phép một bước tiến thêm một phần triệu giây hoặc một thế kỷ. Tuy nhiên, chúng tôi đã quyết định rằng bước thay đổi giờ sẽ không thay đổi, điều này ngụ ý rằng bước thay đổi giờ sẽ không thay đổi giờ. Do đó, các bước thay đổi bây giờ được mô tả bằng hành động sau, trong đó Real là tập hợp tất cả các số thực:

```
NowNext = now [r Real : r > now] now có thể bằng bất kỳ số thực nào > now.
              giờ không đổi
```

Công thức RTnow cũng sẽ cho phép các bước không thay đổi. Giá trị ban đầu của now là một số thực tùy ý (chúng ta có thể khởi động hệ thống bất cứ lúc nào), vì vậy phần an toàn của RTnow là

```
(bây giờ Real) [NowNext]bây giờ
```

Điều kiện sống động mà chúng ta mong muốn là bây giờ phải tăng không giới hạn.

Tính công bằng yếu đơn giản của hành động NowNext là không đủ tốt, bởi vì nó cho phép Tính là các hành vi “Zeno” như công bằng yếu được thảo luận ở Chương 8.

```
[bây giờ = .9] [bây giờ = .99] [bây giờ = .999] [bây giờ = .9999] · · ·
```

trong đó giá trị của bây giờ vẫn bị giới hạn. Tính công bằng yếu của hành động NowNext (now > r) ngụ ý rằng cuối cùng một bước NowNext sẽ xảy ra trong đó giá trị mới của now lớn hơn r . (Hành động này luôn được kích hoạt, do đó tính công bằng yếu hàm ý rằng có vô số hành động như vậy phải xảy ra.) Khẳng định

mô-đ un RealTimeHourClock

```
mở rộng Reals, HourClock
biến now Thời gian hiện tại, đ ược tính bằng giây.
hằng số Rho Một số thực dương. giả sử (Rho
  Thực)   (Rho > 0)
```

mô-đ un bên trong

```
  biến t
  TNext   = t = if HCnxt then 0 else t + (now   now)
  hẹn giờ       = (t = 0)     [TNext]t, giờ, bây giờ      t là thời gian trôi qua kể từ bước HCnxt cuối cùng.
  Thời gian tối đ a   =   (t 3600 + Rho)                  t luôn lớn nhất là 3600 + Rho.
  Thời gian tối thiểu   =   [HCnxt   t ≥ 3600   Rho]hr    Bước HCnxt chỉ có thể xảy ra nếu t ≥ 3600   Rho.
  HCTime   = Bộ đ ếm thời gian   Thời gian tối đ a   Thời gian tối thiểu
```

```
I(t)   = thể hiện Bên trong
NowNext   =   now   {r   Real : r > now}      Bước NowNext có thể tiến tới ngay bây giờ với số lượng bất
                  giờ không đ ổi              kỳ  trong khi không thay đ ổi giờ.
RTnow   =   now   Real                        RTnow chỉ đ ịnh thời gian có thể thay đ ổi như thế nào.
              [NowNext]now
             r   Real : WFnow (NowNext   (now > r ))
RTHC   = HC   RTnow   (   t : I(t)!HCTime) Thông số kỹ thuật đ ầy đ ủ.
```

Hình 9.1: Thông số thời gian thực của đ ồng hồ giờ tích tắc mỗi giờ, cộng hoặc trừ giây Rho.

đ iều này với mọi số thực r ngụ ý rằng bây giờ tăng không giới hạn, nên chúng ta lấy đ iều kiện công bằng1

```
          r   Thực : WFnow (NowNext   (bây giờ > r ))
```

Thông số kỹ thuật RTHC hoàn chỉnh của đ ồng hồ giờ thời gian thực, với đ ịnh nghĩ a về công thức RTnow, nằm trong mô-đ un RealTimeHourClock của Hình 9.1 trên trang này. Mô-đ un đ ó mở rộng mô-đ un Reals tiêu chuẩn, xác đ ịnh tập Real của các số thực.

1Một đ iều kiện tương đ ương là r Real : (now > r), nhưng tôi muốn thể hiện sự công bằng bằng Công thức WF và SF.

9.2 Thông số kỹ thuật thời gian thực nói chung

Trong Phần 8.4 (trang 96), chúng ta đ ã thấy rằng sự khái quát thích hợp về yêu cầu về tính sống đ ộng mà đ ồng hồ giờ tích tắc thường xuyên là tính công bằng yếu của hành đ ộng tích tắc đ ồng hồ. Có một cách khái quát tương tự cho các thông số kỹ thuật thời gian thực. Tính công bằng yếu của hành đ ộng A khẳng đ ịnh rằng nếu A đ ược bật liên tục thì bước A cuối cùng phải xảy ra. Tương tự thời gian thực là nếu A đ ược bật liên tục trong vài giây thì bước A phải xảy ra. Vì một hành đ ộng HCnxt luôn đ ược kích hoạt nên yêu cầu đ ồng hồ tích tắc ít nhất một lần trong mỗi 3600 + ρ giây có thể đ ược biểu thị theo cách này bằng cách đ ặt A là HCnxt và là 3600 + ρ.

Yêu cầu một hành đ ộng HCnxt xảy ra nhiều nhất một lần trong mỗi 3600 - ρ giây có thể đ ược khái quát tương tự với đ iều kiện là một hành đ ộng A phải đ ược kích hoạt liên tục trong ít nhất δ giây trước khi bước A có thể xảy ra.

Điều kiện đ ầu tiên, giới hạn trên về khoảng thời gian A có thể đ ược kích hoạt mà không xảy ra bước A, đ ược thỏa mãn một cách rõ ràng nếu bằng Vô cực—một giá trị đ ược xác đ ịnh trong mô-đ un Reals lớn hơn bất kỳ số thực nào. Điều kiện thứ hai, giới hạn dưới δ về khoảng thời gian A phải đ ược kích hoạt trước khi bước A có thể xảy ra, đ ược thỏa mãn hoàn toàn nếu δ bằng 0. Vì vậy, không có gì bị mất khi kết hợp cả hai đ iều kiện này thành một công thức duy nhất chứa δ và làm tham số . Bây giờ tôi đ ịnh nghĩ a một công thức như vậy mà tôi gọi là đ iều kiện ràng buộc thời gian thực.

Công thức công bằng yếu WFv (A) thực tế khẳng đ ịnh tính công bằng yếu của hành đ ộng Av bằng A , (v = v). Chỉ số v là cần thiết đ ể loại trừ các bước nói lắp. Vì tính đ úng đ ắn của một công thức có ý nghĩ a không thể phụ thuộc vào việc có các bước nói lắp hay không nên sẽ vô nghĩ a khi nói rằng bước A đ ã xảy ra hay không xảy ra nếu bước đ ó có thể là một bước nói lắp. Vì lý do này, đ iều kiện thời gian thực tương ứng cũng phải là đ iều kiện cho một hành đ ộng Av hành đ ộng tùy ý A. Trong hầu hết các trường hợp , không phải trên một cần quan tâm, v là bộ tất cả các biến xuất hiện trong A. Do đ ó tôi đ ịnh nghĩ a thời gian thực công thức ràng buộc RTBound(A, v, δ,) đ ể khẳng đ ịnh rằng

- Bước Av không thể xảy ra cho đ ến khi Av đ ược bật liên tục trong ít nhất δ đ ơn vị thời gian kể từ bước Av cuối cùng —hoặc kể từ khi bắt đ ầu hành vi.

- Av có thể đ ược bật liên tục trong hầu hết các đ ơn vị thời gian trước Av bước xảy ra.

RTBound(A, v, δ,) tổng quát hóa công thức t : I(t)!HCTime của đ ặc tả đ ồng hồ giờ thời gian thực và nó có thể đ ược đ ịnh nghĩ a theo cách tương tự, bằng cách sử dụng mô-đ un con. Tuy nhiên, đ ịnh nghĩ a có thể đ ược cấu trúc gọn hơn một chút như sau:

Đối với thông số kỹ thuật TLA+ , tôi đ ã thay thế δ và bằng D và E.

RTBound(A, v, D, E) = đ ặt Bộ đ ịnh thời (t) = . . .

. . .

trong t : Bộ hẹn giờ (t) . . .

Trước tiên, chúng tôi đ ịnh nghĩ a Bộ hẹn giờ (t) là một công thức tạm thời khẳng đ ịnh rằng t luôn bằng khoảng thời gian mà Av đ ược bật liên tục kể từ bước Av cuối cùng . Giá trị của t phải đ ược đ ặt thành 0 theo bước Av hoặc bước tắt Av . Một bước tiến lên bây giờ sẽ tăng t bây giờ nếu Av đ ược bật. Do đ ó, những thay đ ổi đ ối với t đ ược mô tả bằng hành đ ộng

TNext(t) = t = if Av и (đ ã bật Av) rồi 0 nữa

t + (bây giờ bây giờ)

Chúng ta chỉ quan tâm đ ến ý nghĩ a của Bộ đ ịnh thời (t) khi v là một bộ có các thành phần bao gồm tất cả các biến có thể xuất hiện trong A. Trong trường hợp này, một bước mà v không thay đ ổi thì không thể bật hoặc tắt Av . Vì vậy, công thức Bộ đ ếm thời gian (t) sẽ cho phép các bước giữ nguyên t, v và bây giờ không thay đ ổi. Đặt giá trị ban đ ầu của t là 0, chúng tôi xác đ ịnh

Bộ đ ịnh thời (t) = (t = 0) [TNext(t)]t, v, bây giờ

Các công thức MaxTime và MinTime của thông số kỹ thuật của đ ồng hồ giờ thời gian thực có những khái quát rõ ràng:

- MaxTime(t) khẳng đ ịnh rằng t luôn nhỏ hơn hoặc bằng E:

  Thời gian tối đ a(t) = (t ≤ E)

- MinTime(t) khẳng đ ịnh rằng bước Av chỉ có thể xảy ra nếu t ≥ D:

  Thời gian tối thiểu(t) = [A (t ≥ D)]v

  (Một đ ịnh nghĩ a hợp lý không kém của MinTime(t) là [Av (t ≥ D)]v , nhưng trên thực tế, cả hai đ ều tương đ ương nhau.)

Sau đ ó, chúng tôi xác đ ịnh RTBound(A, v, D, E) bằng

t : Bộ hẹn giờ (t) MaxTime(t) MinTime(t)

Chúng ta cũng phải khái quát công thức RTnow của thông số kỹ thuật của đ ồng hồ giờ thời gian thực. Công thức đ ó mô tả hiện tại thay đ ổi như thế nào và nó khẳng đ ịnh rằng giờ không thay đ ổi khi bây giờ thay đ ổi. Tổng quát hóa là công thức RTnow(v), thay thế hr bằng một hàm trạng thái tùy ý v thường là bộ của tất cả các biến, ngoại trừ biến bây giờ, xuất hiện trong đ ặc tả. Sử dụng các đ ịnh nghĩ a này, đ ặc tả RTHC của đ ồng hồ giờ thời gian thực có thể đ ược viết

HC RTnow(hr) RTBound(HCnxt, hr , 3600 Rô, 3600 + Rô)

Mô-đ un RealTime, với các đ ịnh nghĩ a về RTBound và RTnow, xuất hiện trong Hình 9.2 trên trang 125.

Tính công bằng mạnh sẽ củng cố tính công bằng yếu bằng cách yêu cầu bước A xảy ra không chỉ nếu hành đ ộng A đ ược kích hoạt liên tục mà còn nếu nó đ ược kích hoạt nhiều lần. Hiện tại

được kích hoạt nhiều lần bao gồm khả năng nó cũng bị vô hiệu hóa nhiều lần. Tương tự, chúng ta có thể tăng cường các điều kiện ràng buộc thời gian thực bằng cách xác định công thức mạnh hơn SRTBound(A, v, δ,) để khẳng định rằng

- Bước Av không thể xảy ra cho đến khi Av được bật trong tổng số ít nhất δ đơn vị thời gian kể từ bước Av cuối cùng —hoặc kể từ khi bắt đầu hành vi.

- Có thể bật Av cho tổng số đơn vị thời gian trước Av bước xảy ra.

Nếu < Vô cực thì RTBound(A, v, δ,) ngụ ý rằng bước Av phải xảy ra nếu Av được bật liên tục trong vài giây. Do đó, nếu Av được bật vĩ nh viễn thì phải xảy ra vô số bước Av . Do đó, RTBound(A, v, δ,) hàm ý tính công bằng yếu của A. Chính xác hơn, RTBound(A, v, δ,) và RTnow(v) cùng hàm ý WFv (A). Tuy nhiên, SRTBound(A, v, δ,) không hàm ý tính công bằng mạnh mẽ của A. Nó cho phép các hành vi trong đó Av được bật vô hạn nhưng không bao giờ được thực thi—ví dụ: A có thể được bật trong /2 giây, sau đó cho /4 giây, sau đó là /8 giây, v.v. Vì lý do này, SRTBound dường như không có nhiều ứng dụng thực tế nên tôi sẽ không bận tâm đến việc định nghĩa nó một cách chính thức.

9.3 Bộ nhớ đệm thời gian thực

Bây giờ chúng ta hãy sử dụng mô-đun RealTime để viết các phiên bản thời gian thực của đặc tả bộ nhớ có khả năng tuyến tính hóa của Phần 5.3 (trang 51) và đặc tả bộ nhớ đệm ghi của Phần 5.6 (trang 54). Chúng tôi thu được thông số kỹ thuật của bộ nhớ thời gian thực bằng cách tăng cường thông số kỹ thuật trong mô-đun Bộ nhớ (Hình 5.3 trên trang 53) để yêu cầu bộ nhớ phản hồi các yêu cầu của bộ xử lý trong vòng vài giây Rho. Thông số kỹ thuật bộ nhớ hoàn chỉnh của mô-đun Bộ nhớ có được bằng cách ẩn các biến mem, ctl và buf trong ISpec đặc tả bên trong của mô-đun InternalMemory. Nói chung, việc thêm ràng buộc thời gian thực vào đặc tả nội bộ sẽ dễ dàng hơn, trong đó các ràng buộc có thể đề cập đến các biến nội bộ (ẩn). Vì vậy, trước tiên chúng tôi thêm ràng buộc về thời gian vào ISpec và sau đó ẩn các biến nội bộ.

Để chỉ định rằng hệ thống phải phản hồi yêu cầu của bộ xử lý trong vòng Rho giây, chúng tôi thêm một ràng buộc về thời gian giới hạn trên cho một hành động sẽ được kích hoạt khi một yêu cầu được đưa ra và hành động đó chỉ bị vô hiệu hóa (có thể do đang được thực thi) khi bộ xử lý đáp ứng yêu cầu. Trong đặc tả ISpec, việc phản hồi một yêu cầu cần có hai hành động—Do(p) để thực hiện thao tác nội bộ và Rsp(p) để đưa ra phản hồi. Cả hai hành động này đều không phải là hành động chúng ta mong muốn; chúng ta phải xác định một hành động mới cho mục đích này. Có một yêu cầu đang chờ xử lý đối với bộ xử lý p iff ctl[p] bằng “rdy”. Vì vậy, chúng tôi khẳng định rằng

mô-đ un thời gian thực

Mô-đ un này khai báo biến ngay bây giờ, biểu thị thời gian thực và xác đ ịnh các toán tử đ ể viết thông số kỹ thuật thời gian thực. Các ràng buộc thời gian thực đ ược thêm vào một đ ặc tả bằng cách kết hợp nó với RTnow(v) và for-mulas có dạng RTBound(A, v, δ,) cho các hành đ ộng A, trong đ ó v là bộ của tất cả các biến đ ặc tả và 0 ≤ δ Vô cực.

mở rộng số thực

biến now Giá trị của now là một số thực biểu thị thời gian hiện tại, theo đ ơn vị không xác đ ịnh.

RTBound(A, v, δ,) xác nhận rằng bước Av chỉ có thể xảy ra sau khi Av đ ược bật liên tục trong δ đ ơn vị thời gian kể từ bước Av cuối cùng (hoặc bắt đ ầu hành vi) và nó phải xảy ra trước khi Av đ ược bật liên tục đ ược bật trong nhiều đ ơn vị thời gian kể từ bước Av cuối cùng (hoặc bắt đ ầu hành vi).

```
RTBound(A, v, D, E)   = let
      TNext(t)   = t = if Av    и (enabled Av )
                                  thì 0
                                  else t + (bây giờ    bây giờ)
            Bộ đ ịnh thời (t)   = (t = 0)     [TNext(t)]t, v,
            bây giờ MaxTime(t)   =   (t ≤ E)
            MinTime(t)   =   [A    (t ≥ D)]v
      trong    t : Bộ đ ịnh thời (t)    MaxTime(t)    MinTime(t)
```

Bộ đ ịnh thời(t) xác nhận rằng t là khoảng thời gian Av đ ược bật liên tục mà không xảy ra vòng Av .

Khẳng đ ịnh rằng t luôn luôn ≤ E.

Khẳng đ ịnh rằng bước Av chỉ có thể xảy ra nếu t ≥ D.

RTnow(v) khẳng đ ịnh rằng bây giờ là một số thực đ ược tăng không giới hạn, theo các bước tăng tùy ý, theo các bước mà v không thay đ ổi.

```
RTnow(v)   =       hãy đ ể NowNext   =    bây giờ    {r    Real : r > now}
                                          không thay đ ổi v
                   trong    bây giờ    Real
                              [NowNext]now
                            r    Real : WFnow (NowNext    (now > r ))
```

Hình 9.2: Mô-đ un RealTime đ ể viết thông số kỹ thuật thời gian thực.

hành đ ộng sau không thể đ ược kích hoạt lâu hơn Rho giây mà không đ ược thực thi:

```
        Trả lời(p)   = (ctl[p] = “rdy”)    (ctl [p] = “rdy”)
```

Thông số kỹ thuật hoàn chỉnh là công thức RTSpec của mô-đ un RTMemory trong Hình 9.3 ở trang tiếp theo. Để cho phép ẩn các biến mem, ctl và buf, mô-đ un RTMemory chứa mô-đ un con Inner mở rộng mô-đ un InternalMemory.

Sau khi đ ã thêm ràng buộc thời gian thực vào đ ặc tả của bộ nhớ có khả năng tuyến tính hóa, hãy tăng cường đ ặc tả của bộ nhớ đ ệm ghi đ ể nó đ áp ứng

```
mô-đ un RTMemory
Thông số kỹ thuật củng cố đ ặc tả bộ nhớ có thể tuyến tính hóa của Phần 5.3 bằng cách yêu cầu gửi phản hồi tới mọi yêu cầu của bộ xử lý trong vòng
Rho giây.

mở rộng MemoryInterface, RealTime

hằng số Rho

giả sử (Rho    Thực)    (Rho > 0)
    mô-đ un bên trong
    Chúng tôi giới thiệu một mô hình con đ ể có thể ẩn các biến mem, ctl và buf .

    mở rộng bộ nhớ trong

    Trả lời(p)   =                                  Reply(p) đ ược kích hoạt khi nhận đ ược yêu cầu từ p; nó bị vô hiệu hóa khi bước
         (ctl[p] = “rdy”)    (ctl [p] = “rdy”)      Phản hồi(p) đ ưa ra phản hồi.

    RTISpec   =    ISpec                                                    Chúng tôi khẳng đ ịnh đ ộ trễ giới hạn
                         p    Proc : RTBound(Phản hồi(p), ctl, 0, Rho)     trên của Rho trên Reply(p), đ ối với tất cả
                    RTnow(memInt, mem, ctl, buf )                          bộ xử lý p.

Bên trong (mem, ctl, buf )   = instance Inner

RTSpec   =    mem, ctl, buf : Bên trong (mem, ctl, buf )!RTISpec
```

Hình 9.3: Phiên bản thời gian thực của đ ặc tả bộ nhớ tuyến tính hóa.

khắc phục hạn chế đ ó. Đối tượng không chỉ là thêm bất kỳ ràng buộc thời gian thực nào đ ể thực hiện công việc—đ iều đ ó có thể dễ dàng thực hiện bằng cách sử dụng cùng một ràng buộc mà chúng tôi đ ã thêm vào đ ặc tả bộ nhớ. Chúng tôi muốn viết đ ặc tả của thuật toán thời gian thực—một đ ặc tả cho người triển khai biết cách đ áp ứng các ràng buộc thời gian thực. Điều này thường đ ược thực hiện bằng cách đ ặt giới hạn thời gian thực cho các hành đ ộng ban đ ầu của đ ặc tả không tính thời gian, chứ không phải bằng cách thêm giới hạn thời gian cho một hành đ ộng mới, như chúng tôi đ ã làm đ ối với đ ặc tả bộ nhớ. Phải đ ạt đ ược giới hạn trên về thời gian phản hồi bằng cách thực thi các ràng buộc giới hạn trên đ ối với các hành đ ộng của hệ thống.

Nếu chúng tôi cố gắng đ ạt đ ược giới hạn về thời gian phản hồi bằng cách thêm giới hạn thời gian thực vào các hành đ ộng của đ ặc tả bộ nhớ đ ệm ghi qua, chúng tôi sẽ gặp phải vấn đ ề sau. Hoạt đ ộng của các bộ xử lý khác nhau “cạnh tranh” với nhau đ ể xếp các hoạt đ ộng vào hàng đ ợi hữu hạn memQ. Ví dụ: khi thực hiện yêu cầu ghi lại cho bộ xử lý p, hệ thống phải thực thi hành đ ộng DoWr (p) đ ể xếp hoạt đ ộng vào hàng đ ợi ở đ uôi memQ. Hành đ ộng đ ó không đ ược kích hoạt nếu memQ đ ã đ ầy. Hành đ ộng DoWr (p) có thể bị vô hiệu hóa liên tục bởi hệ thống thực hiện các hành đ ộng DoWr hoặc RdMiss cho các bộ xử lý khác. Đó là lý do tại sao, đ ể đ ảm bảo tính thực tế—rằng mỗi yêu cầu cuối cùng đ ều nhận đ ược phản hồi—trong Phần 8.7 (trang 107), chúng tôi phải khẳng đ ịnh tính công bằng mạnh mẽ của các hành đ ộng DoWr và RdMiss. Cách duy nhất đ ể đ ảm bảo

rằng một hành động DoWr (p) được thực thi trong một khoảng thời gian nào đó là sử dụng các ràng buộc giới hạn dưới đối với các hành động của các bộ xử lý khác để đảm bảo rằng chúng không thể thực hiện các hành động DoWr hoặc RdMiss quá thường xuyên. Mặc dù có thể thực hiện được đặc tả như vậy nhưng nó không phải là cách tiếp cận mà bất kỳ ai cũng có thể áp dụng trong thực tế.

Phương pháp thông thường để thực thi các giới hạn thời gian thực đối với quyền truy cập vào tài nguyên dùng chung là lên lịch sử dụng tài nguyên cho các bộ xử lý khác nhau. Vì vậy, hãy sửa đổi bộ đệm ghi để thêm kỷ luật lập lịch cho các hành động xếp hàng các hoạt động trên memQ. Chúng tôi sử dụng lập kế hoạch luân phiên, đây có lẽ là cách dễ thực hiện nhất. Giả sử các bộ xử lý được đánh số từ 0 đến N 1. Lập kế hoạch quay vòng có nghĩa là một thao tác cho bộ xử lý p là thao tác tiếp theo được xếp hàng sau một thao tác cho bộ xử lý q nếu không có thao tác nào cho bất kỳ bộ xử lý nào (q + 1)) %N , (q + 2) %N , . . . , (p 1) % N đang chờ được đưa vào memQ.

Để thể hiện điều này một cách chính thức, trước tiên chúng ta đặt tập Proc của bộ xử lý bằng tập 0 . . (N 1) số nguyên. Chúng ta thường làm điều này bằng cách định nghĩa Proc bằng 0 . . (N 1). Tuy nhiên, chúng tôi muốn sử dụng lại các tham số và định nghĩa từ đặc tả bộ nhớ đệm ghi qua và điều đó dễ thực hiện nhất bằng cách mở rộng mô-đun WriteThroughCache. Vì Proc là tham số của module đó nên chúng ta không thể định nghĩa nó. Do đó, chúng ta đặt N là một tham số không đổi mới và đặt Proc = 0. . (N 1) là một giả định.2 Để triển khai lập kế hoạch quay vòng, chúng tôi sử

dụng biến LastP bằng với bộ xử lý cuối cùng có hoạt động được xếp vào hàng đợi memQ. Chúng tôi xác định vị trí toán tử sao cho p là bộ xử lý vị trí (p) sau LastP theo thứ tự vòng tròn:

vị trí(p) = chọn i 1 . . N : p = (lastP + i) %N

(Do đó, vị trí (lastP) bằng N .) Một thao tác cho bộ xử lý p có thể là thao tác tiếp theo truy cập memQ nếu không có thao tác nào cho bộ xử lý q có vị trí (q) < vị trí (p) sẵn sàng truy cập vào nó—nghĩa là, nếu canGoNext(p) là đúng, ở đâu

canGoNext(p) = q Proc : (position(q) < location(p)) và đã bật
(RdMiss(q) DoWr (q))

Sau đó, chúng tôi xác định RTRdMiss(p) và RTDoWr (p) giống với RdMiss(p) và DoWr (p), ngoại trừ việc chúng có điều kiện kích hoạt bổ sung canGoNext(p) và chúng đặt LastP thành p. Các hành động phụ khác của hành động ở trạng thái tiếp theo cũng giống như trước, ngoại trừ việc chúng cũng phải giữ nguyên LastP.

Để đơn giản, chúng tôi giả sử một giới hạn trên duy nhất của Epsilon về khoảng thời gian bất kỳ hành động nào của bộ xử lý p có thể vẫn được kích hoạt mà không được thực thi—ngoại trừ hành động Evict(p, a) mà chúng tôi không bao giờ yêu cầu xảy ra.

trong số đó. Nói chung, giả sử A1, . . Ak, là những hành động sao cho (i) không có hai hành động nào

2Chúng ta cũng có thể khởi tạo mô-đun WriteThroughCache bằng 0 . . (N 1) thay thế cho Proc; nhưng điều đó sẽ yêu cầu khai báo các tham số khác của WriteThroughCache, bao gồm cả các tham số từ mô-đun MemoryInterface.

được kích hoạt đồng thời và (ii) khi bất kỳ Ai nào được kích hoạt, nó phải được thực thi trước khi một Aj khác có thể được kích hoạt. Trong trường hợp này, một ràng buộc RTBound duy nhất trên A1 . . . Ak tương đương với các ràng buộc riêng biệt trên tất cả Ai . Do đó, chúng tôi có thể đặt một ràng buộc duy nhất đối với việc phân tách tất cả các hành động của bộ xử lý p, ngoại trừ việc chúng tôi không thể sử dụng cùng một ràng buộc cho cả DoRd(p) và RTRdMiss(p) vì bước Evict(p, a) có thể vô hiệu hóa DoRd(p) và kích hoạt RTRdMiss(p). Do đó, chúng tôi sử dụng một ràng buộc riêng cho RTRdMiss(p).

Chúng tôi giả định giới hạn trên của Delta tại thời điểm MemQWr hoặc MemQRd có thể được bật mà không cần loại bỏ một thao tác khỏi memQ. Biến memQ biểu thị một hàng đợi vật lý giữa bus và bộ nhớ chính và Delta phải đủ lớn để một thao tác được chèn vào hàng đợi trống sẽ đến bộ nhớ và bị loại bỏ trong vòng vài giây Delta.

Chúng tôi muốn bộ nhớ đệm ghi theo thời gian thực triển khai đặc tả bộ nhớ thời gian thực. Điều này đòi hỏi một giả định liên quan đến Delta, Epsilon và Rho để đảm bảo rằng ràng buộc về thời gian của đặc tả bộ nhớ được thỏa mãn—cụ thể là độ trễ giữa thời điểm bộ nhớ nhận được yêu cầu từ bộ xử lý p và khi nó phản hồi nhiều nhất là Rho. Việc xác định giả định này đòi hỏi phải tính toán giới hạn trên của độ trễ đó. Tìm giới hạn trên nhỏ nhất thật khó; nó dễ dàng hơn để chỉ ra điều đó

2 (N + 1) Epsilon + (N + QLen) Delta

là giới hạn trên. Vì vậy, chúng tôi giả định rằng giá trị này nhỏ hơn hoặc bằng Rho.

Thông số kỹ thuật đầy đủ xuất hiện trong Hình 9.4 ở hai trang sau. Mô-đun này cũng khẳng định như một định lý rằng đặc tả RTSpec của bộ nhớ đệm ghi qua thời gian thực thực hiện (ngụ ý) đặc tả bộ nhớ thời gian thực, công thức RTSpec của mô-đun RTMemory.

9.4 Thông số kỹ thuật Zeno

Tôi đã mô tả công thức RTBound(HCnxt, hr , δ,) như khẳng định rằng bước HCnxt phải xảy ra trong vòng vài giây của bước HCnxt trước đó. Tuy nhiên, ẩn chứa trong mô tả này là một khái niệm về quan hệ nhân quả không có trong công thức. Sẽ chính xác hơn nếu mô tả công thức như khẳng định rằng bây giờ không thể tiến thêm quá vài giây trước khi bước HCnxt tiếp theo xảy ra.

Công thức không cho chúng ta biết liệu điều kiện này có được đáp ứng bằng cách làm cho đồng hồ kêu tích tắc hay bằng cách ngăn thời gian trôi đi hay không. Thật vậy, công thức được thỏa mãn bởi hành vi “Zeno”:3

giờ = 11	giờ = 11	giờ = 11	giờ = 11	. . .
bây giờ = 0	bây giờ = /2	bây giờ = 3/4	bây giờ = 7/8	

3Nhà triết học Hy Lạp Zeno đã đặt ra một nghịch lý rằng trước tiên một mũi tên phải đi được một nửa quãng đường tới mục tiêu, sau đó là một phần tư khoảng cách tiếp theo, rồi đến một phần tám tiếp theo, v.v.; do đó nó sẽ không thể hạ cánh trong một khoảng thời gian hữu hạn.

```
mô-đun RTWriteThroughCache mở rộng
WriteThroughCache, RealTime
hằng số N
giả sử (N ∈ Nat) ∧ (Proc = 0 . . N − 1) các
hằng số Delta, Epsilon, Rho Một số giới hạn thời gian thực đối với
các hành động. giả sử (Delta ∈ Real)
         (Delta > 0) ∧ (Epsilon ∈ Real)
         (Epsilon > 0) ∧ (Rho ∈ Real)
           (Rho > 0) ∧ 2 ∗ (N + 1) ∗ Epsilon + (N + QLen) ∗ Delta < Rho
```

Chúng tôi giả sử rằng tập Proc của bộ xử lý bằng 0 . . N-1.

Chúng tôi sửa đổi đặc tả bộ đệm ghi qua để yêu cầu các hoạt động cho các bộ xử lý khác nhau được xếp vào hàng đợi trên memQ theo thứ tự vòng tròn.

```
biến LastP Bộ xử lý cuối cùng sắp xếp một thao tác trên memQ.
RTInit = Init ∧ (lastP ∈ Proc) Ban đầu, LastP có thể bằng bất kỳ bộ xử lý nào.

vị trí(p) = p là vị trí(p) bộ xử lý thứ sau P cuối cùng theo thứ tự vòng tròn.
  chọn i ∈ 1 . . N : p = (lastP + i) %N

canGoNext(p) = Đúng nếu bộ xử lý p có thể là bộ xử lý tiếp theo xếp hàng một thao tác
  trên memQ. ∀ q ∈ Proc : (vị trí(q) < vị trí(p)) ∧ được bật (RdMiss(q) ∨ DoWr (q))

RTRdMiss(p) = canGoNext(p)
                RdMiss(p)
                LastP = p

RTDoWr (p) = canGoNext(p) ∧ DoWr
               (p) ∧ LastP
               = p

RTNext = ∨ p ∈ Proc : RTRdMiss(p) ∨ RTDoWr (p)
         ∨ p ∈ Proc : Req(p) ∨ Rsp(p) ∨ DoRd(p) ∨ a
                        ∈ Adr : Evict(p , Một)
             ∨ MemQWr ∨ MemQRd
         không thay đổi lần cuối

vars = memInt, wmem, buf , ctl, cache, memQ, LastP
```

Các hành động RTRdMiss(p) và RTDoWr(p) giống như RdMiss(p) và DoWr(p) ngoại trừ việc chúng không được kích hoạt trừ khi p là bộ xử lý tiếp theo theo thứ tự vòng tròn sẵn sàng xếp hàng một thao tác trên memQ, và họ đặt LastP thành p.

Hành động trạng thái tiếp theo RTNext giống như Next ngoại trừ RTRdMiss(p) và RTDoWr(p) được thay thế bằng RdMiss(p) và DoWr(p) và các hành động khác được sửa đổi để giữ nguyên LastP.

Hình 9.4a: Phiên bản thời gian thực của bộ nhớ đệm ghi (bắt đầu).

```
RTSpec  =
   RTInit     [RTNext]vars
   RTBound(MemQWr    MemQRd, vars, 0, Delta)      p    Proc :
      RTBound(RTDoWr (p)    DoRd(p)    Rsp(p), vars, 0, Epsilon)
                                      RTBound(RTRdMiss(p),
                    vars, 0, Epsilon)    RTnow(vars)
```

Chúng tôi đặt độ trễ giới hạn trên của Delta cho các hành động MemQWr và MemQRd (loại bỏ các hoạt động khỏi memQ) và độ trễ giới hạn trên của Epsilon đối với các hành động khác.

```
RTM  = thể hiện Định lý RTMemory RTSpec
   RTM !RTSpec
```

Hình 9.4b: Phiên bản thời gian thực của bộ nhớ đệm ghi (cuối).

trong đó từng giây không bao giờ trôi qua. Chúng tôi loại trừ các hành vi Zeno như vậy bằng cách kết hợp với đặc tả của chúng tôi công thức RTnow(hr)—chính xác hơn bằng cách kết hợp liên từ tính sống động của nó

```
          r    Thực : WFnow (Tiếp theo    (bây giờ > r ))
```

ngụ ý rằng thời gian tiến triển không giới hạn. Hãy gọi công thức này là NZ (đối với Non-Zeno).

Hành vi Zeno không gây ra vấn đề gì; chúng bị cấm một cách tầm thường bằng cách kết hợp với New Zealand. Một vấn đề tồn tại nếu đặc tả chỉ cho phép các hành vi Zeno. Ví dụ: giả sử chúng ta gắn với đặc tả của đồng hồ giờ không tính thời gian điều kiện RTBound(HCnxt, hr , δ,) với một số δ và với δ > . Điều này khẳng định rằng đồng hồ phải đợi ít nhất δ giây trước khi tích tắc, nhưng phải tích tắc trong khoảng thời gian ngắn hơn. Nói cách khác, đồng hồ không bao giờ có thể tích tắc. Chỉ có hành vi Zeno, trong đó số giây không bao giờ trôi qua, mới có thể đáp ứng thông số kỹ thuật này.
Việc kết hợp NZ với đặc điểm kỹ thuật này mang lại một công thức cho phép không có hành vi nào— tức là một công thức tương đương với sai.

Ví dụ này là một trường hợp cực đoan của cái được gọi là đặc tả Zeno. Đặc tả Zeno là đặc tả tồn tại hành vi hữu hạn σ thỏa mãn phần an toàn nhưng không thể mở rộng thành hành vi vô hạn thỏa mãn cả phần an toàn và NZ.

[4] Nói cách khác, các hành vi hoàn chỉnh duy nhất thỏa mãn phần an toàn mở rộng σ là các hành vi Zeno. Một thông số kỹ thuật không phải là Zeno đương nhiên được cho là không phải Zeno. Theo định nghĩa về việc đóng máy (trong Phần 8.9.2 trên trang 111), một thông số kỹ thuật sẽ không phải là Zeno nếu nó là máy đóng. Chính xác hơn, nó không phải là Zeno nếu cặp thuộc tính bao gồm phần an toàn của đặc tả (sự kết hợp của đặc tả không xác định thời gian, điều kiện ràng buộc thời gian thực và phần an toàn của công thức RTnow) và NZ là máy đóng .

4Hãy nhớ lại rằng, ở trang 112, một hành vi hữu hạn σ được xác định để thỏa mãn tính chất an toàn P nếu việc thêm vô số bước lắp bắp vào cuối σ sẽ tạo ra một hành vi thỏa mãn P.

Đặc tả Zeno là đặc tả trong đó yêu cầu về thời gian tăng lên không giới hạn sẽ loại trừ một số hành vi hữu hạn lẽ ra được phép.

Thông số kỹ thuật như vậy có thể không chính xác vì các điều kiện ràng buộc thời gian thực có thể đang hạn chế hệ thống theo những cách không lường trước được. Về mặt này, thông số kỹ thuật của Zeno cũng giống như các thông số kỹ thuật không đóng máy khác.

Mục 8.9.2 đề cập rằng sự kết hợp của các điều kiện công bằng trên các hành động phụ của quan hệ trạng thái tiếp theo sẽ tạo ra một đặc tả máy đóng. Có một kết quả tương tự cho các điều kiện RTBound và các thông số kỹ thuật không phải Zeno. Một đặc tả không phải là Zeno nếu nó là sự kết hợp của (i) một công thức có dạng Init [Next]vars , (ii) công thức RTnow(vars) và (iii) một số hữu hạn các công thức có dạng RTBound(Ai , vars, δi , i), ở đâu cho mỗi i

- 0 i i Vô cực
- Ai là một phần phụ của hành động trạng thái tiếp theo Tiếp theo.
- Không có bước nào vừa là A… và bước Aj , với mọi Aj có j = i.

Định nghĩa của một hành động phụ xuất hiện ở trang 111.

Đặc biệt, điều này ngụ ý rằng đặc tả RTSpec của bộ nhớ đệm ghi theo thời gian thực trong mô-đun RTWriteThroughCache không phải là Zeno.

Kết quả này không áp dụng cho đặc tả của bộ nhớ thời gian thực trong mô-đun RTMemory (Hình 9.3 trên trang 126) vì hành động Reply(p) không phải là một hành động phụ của hành động trạng thái tiếp theo INext của công thức ISpec. Tuy nhiên, cation được xác định INext vẫn là phi Zeno, bởi vì mọi hành vi hữu hạn σ thỏa mãn đặc tả đều có thể được mở rộng đến một hành vi trong đó thời gian tiến triển không giới hạn. Ví dụ: trước tiên chúng ta có thể mở rộng σ để đáp ứng ngay lập tức tất cả các yêu cầu đang chờ xử lý (trong thời gian 0), sau đó mở rộng nó thành hành vi vô hạn bằng cách thêm các bước vừa tăng lên ngay bây giờ.

ở trang 53

Thật dễ dàng để xây dựng một ví dụ trong đó việc kết hợp công thức RTBound cho một hành động không phải là một phần phụ của hành động trạng thái tiếp theo sẽ tạo ra đặc tả Zeno. Ví dụ, hãy xem xét công thức

(9.2) HC RTBound(hr , 0, 3600) RTnow(hr) = hr 1, hr

trong đó HC là thông số kỹ thuật của đồng hồ giờ. Hành động trạng thái tiếp theo HCnxt của HC xác nhận rằng hr được tăng thêm 1 hoặc thay đổi từ 12 thành 1. Công thức RTBound xác nhận rằng bây giờ không thể tiến lên trong 3600 giây trở lên nếu không xảy ra bước hr = hr 1. Vì HC khẳng định rằng mỗi bước thay đổi giờ là một bước HCnxt, nên phần an toàn của (9.2) chỉ được thỏa mãn bởi các hành vi trong đó hiện tăng ít hơn 3600 giây. Do đặc tả đầy đủ (9.2) chứa NZ liên hợp, xác nhận rằng bây giờ tăng không giới hạn, nên nó tương đương với sai và do đó là đặc tả Zeno.

Khi một đặc tả mô tả cách hệ thống được triển khai, các ràng buộc thời gian thực có thể được biểu thị dưới dạng công thức RTBound cho các hành động phụ của hành động trạng thái tiếp theo. Đây là những loại công thức tương ứng khá trực tiếp với việc triển khai. Ví dụ: mô-đun RTWriteThroughCache

mô tả một thuật toán để triển khai bộ nhớ và nó có giới hạn thời gian thực đối với các hoạt động phụ của hành động trạng thái tiếp theo. Mặt khác, các thông số kỹ thuật cấp cao hơn, trừu tượng hơn—những thông số mô tả những gì hệ thống phải làm hơn là cách thực hiện nó—ít có khả năng có các ràng buộc thời gian thực được thể hiện theo cách này. Do đó, thông số kỹ thuật cấp cao của bộ nhớ thời gian thực trong mô-đun RTMemory chứa công thức RTBound cho một hành động không phải là hành động phụ của hành động trạng thái tiếp theo.

9.5 Thông số kỹ thuật hệ thống hybrid

Hệ thống được mô tả bằng đặc tả TLA+ là một thực thể vật lý. Các biến số của đặc tính thể hiện một phần nào đó của trạng thái vật lý - sự hiển thị của đồng hồ hoặc sự phân bố điện tích trong một miếng silicon thực hiện một ô nhớ.
Trong đặc tả thời gian thực, biến bây giờ khác với các biến khác vì chúng ta không loại bỏ bản chất liên tục của thời gian. Đặc điểm kỹ thuật bây giờ cho phép giả định bất kỳ giá trị liên tục nào. Các trạng thái rời rạc trong một hành vi có nghĩa là chúng ta đang quan sát trạng thái của hệ thống và do đó giá trị của hiện tại tại một chuỗi các thời điểm rời rạc.

Có thể có những đại lượng vật lý khác ngoài thời gian mà chúng ta muốn biểu thị tính chất liên tục của chúng trong một đặc tả. Đối với hệ thống kiểm soát không lưu, chúng ta có thể muốn biểu diễn vị trí và vận tốc của máy bay. Đối với một hệ thống điều khiển lò phản ứng hạt nhân, chúng ta có thể muốn biểu diễn các thông số vật lý của chính lò phản ứng đó. Thông số kỹ thuật đại diện cho các đại lượng thay đổi liên tục như vậy được gọi là thông số kỹ thuật của hệ thống lai.

Ví dụ, hãy xem xét một hệ thống, trong số những thứ khác, điều khiển một công tắc ảnh hưởng đến chuyển động một chiều của một số vật thể. Giả sử vị trí p của vật tuân theo một trong các định luật sau, tùy thuộc vào công tắc tắt hay bật:

(9.3) d 2p/dt2 + c dp/dt + f [t] = 0 d
2p/dt2 + c dp/dt + f [t] + k p = 0

trong đó c và k là các hằng số, f là một hàm nào đó và t biểu thị thời gian. Tại bất kỳ thời điểm nào, vị trí tương lai của vật thể được xác định bởi vị trí và vận tốc hiện tại của vật thể. Vì vậy, trạng thái của vật thể được mô tả bởi hai biến số—đó là vị trí p và vận tốc w của nó. Các biến này có liên quan bởi w = dp/dt.

Chúng tôi mô tả hệ thống này bằng đặc tả TLA+ trong đó các biến p và w chỉ được thay đổi theo các bước thay đổi ngay bây giờ—nghĩa là các bước biểu thị thời gian trôi qua. Chúng tôi chỉ định các thay đổi đối với trạng thái hệ thống rời rạc và mọi ràng buộc thời gian thực như trước đây. Tuy nhiên, chúng tôi thay thế RTnow(v) bằng một công thức có hành động ở trạng thái tiếp theo sau, trong đó Tích hợp và D được giải thích

bên dưới và v là bộ của tất cả các biến rời rạc:

```
  now    {r    Real : r > now} w =
thay    Integrate(D, now, now ,p, w )    p ,    không
đổi v Các biến rời rạc thay đổi tức thời.
```

Liên từ thứ hai khẳng định rằng p và w bằng các biểu thức thu được bằng cách giải phương trình vi phân thích hợp cho vị trí và vận tốc của vật thể tại thời điểm hiện tại , giả sử rằng giá trị của chúng tại thời điểm hiện tại là p và w. Phương trình vi phân được chỉ định bởi D, trong khi Tích phân là toán tử tổng quát để giải (tích phân) một phương trình vi phân tùy ý.

Để xác định phương trình vi phân mà đối tượng thỏa mãn, giả sử rằng switchOn là một biến trạng thái có giá trị Boolean mô tả vị trí của công tắc. Sau đó, chúng ta có thể viết lại cặp phương trình (9.3) thành 2 p/dt2 + c dp/dt + f [t] + (nếu

d switchOn thì k p else 0) = 0

Sau đó, chúng ta xác định hàm D để phương trình này có thể được viết là

```
D[t, p, dp/dt, d   2 p/dt2 ] = 0
```

Sử dụng ký hiệu TLA+ để xác định hàm của nhiều đối số, được giải thích trong Phần 16.1.7 trên trang 301, định nghĩa là

```
D[t, p0, p1, p2    Real]   = p2
     + c    p1 + f [t] + (nếu switchOn thì k    p0 else 0)
```

Chúng ta thu được đặc tả mong muốn nếu toán tử Tích hợp được xác định sao cho Tích phân(D, t0, t1, x 0, . . . , x n 1) là giá trị tại thời điểm t1 của n-tuple

x , dx/dt, . . . , d n 1 /dtn 1

trong đó x là nghiệm của phương trình vi phân

```
D[t, x , dx/dt, . . . , d n x/ctn ] = 0
```

có đạo hàm bậc 0 đến (n 1) tại thời điểm t0 là x 0, . . . của , x n 1. Định nghĩa a-Tích phân xuất hiện trong mô-đun Phương trình vi phân của Phần 11.1.3 (trang 174).

Nhìn chung, đặc tả hệ thống lai tương tự như đặc tả thời gian thực, ngoại trừ công thức RTnow(v) được thay thế bằng công thức mô tả những thay đổi đối với tất cả các biến thể hiện các đại lượng vật lý thay đổi liên tục. Toán tử Tích hợp sẽ cho phép bạn chỉ định những thay đổi đó cho nhiều hệ thống kết hợp. Một số hệ thống sẽ yêu cầu các toán tử khác nhau. Ví dụ, việc mô tả sự tiến triển của một số đại lượng vật lý có thể yêu cầu một toán tử mô tả nghiệm của phương trình vi phân từng phần. Tuy nhiên, nếu bạn có thể mô tả sự tiến hóa bằng toán học thì nó có thể được chỉ định trong TLA+.

Các thông số kỹ thuật của hệ thống hybrid dường như vẫn chỉ được giới học thuật quan tâm nên tôi sẽ không nói thêm gì về chúng. Nếu bạn có dịp viết một bài, cuộc thảo luận ngắn gọn này sẽ chỉ ra cách bạn có thể thực hiện điều đó.

9.6 Nhận xét về thời gian thực

Các ràng buộc thời gian thực được sử dụng thường xuyên nhất để đặt giới hạn trên về thời gian hệ thống có thể thực hiện một việc gì đó. Với khả năng này, chúng có thể được coi là một dạng sống động mạnh mẽ, không chỉ xác định rằng điều gì đó cuối cùng phải xảy ra mà còn xác định thời điểm nó phải xảy ra. Trong các thông số kỹ thuật rất đơn giản, chẳng hạn như đồng hồ giờ và bộ đệm ghi, các ràng buộc thời gian thực thường thay thế các điều kiện tồn tại. Các thông số kỹ thuật phức tạp hơn có thể khẳng định cả các ràng buộc thời gian thực và các thuộc tính sống động.

Các thông số kỹ thuật thời gian thực mà tôi thấy không yêu cầu các ràng buộc về thời gian quá phức tạp. Chúng là thông số kỹ thuật của các thuật toán khá đơn giản trong đó các ràng buộc về thời gian là rất quan trọng đối với tính chính xác hoặc của các hệ thống phức tạp hơn trong đó thời gian thực chỉ xuất hiện thông qua việc sử dụng thời gian chờ đơn giản để đảm bảo tính sống động. Tôi nghi ngờ rằng mọi người không xây dựng các hệ thống với các ràng buộc thời gian thực phức tạp vì quá khó để làm cho chúng đúng.

Tôi đã mô tả cách viết một đặc tả thời gian thực bằng cách kết hợp các công thức RTnow và RTBound với một đặc tả không tính thời gian. Người ta có thể chứng minh rằng tất cả các thông số kỹ thuật thời gian thực có thể được viết dưới dạng này. Trên thực tế, chỉ cần sử dụng công thức RTBound cho các hành động phụ của hành động ở trạng thái tiếp theo là đủ. Tuy nhiên, kết quả này chỉ được quan tâm về mặt lý thuyết vì đặc điểm kỹ thuật thu được có thể cực kỳ phức tạp. Toán tử RTnow và RTBound giải quyết tất cả các vấn đề về đặc tả thời gian thực mà tôi gặp phải; nhưng tôi chưa gặp đủ để tự tin nói rằng đó là tất cả những gì bạn cần. Tuy nhiên, tôi khá tự tin rằng, bất kể thuộc tính thời gian thực nào bạn phải chỉ định, sẽ không khó để thể hiện chúng trong TLA+.