Chương 7

Viết một đặc tả: Vài lời khuyên

Bây giờ bạn đã tìm hiểu tất cả những điều cần biết về TLA+ để viết các thông số kỹ thuật của riêng mình. Dưới đây là một số gợi ý bổ sung để giúp bạn bắt đầu.

7.1 Tại sao chỉ định

Viết một đặc tả đòi hỏi nỗ lực; lợi ích mà nó mang lại phải biện minh cho nỗ lực đó. Mục đích của việc viết đặc tả là giúp tránh lỗi. Dưới đây là một số cách nó có thể làm điều đó.

- Viết đặc tả TLA+ có thể giúp ích cho quá trình thiết kế. Việc phải mô tả một cách chính xác một thiết kế thường bộc lộ các vấn đề—các tương tác tinh tế và các "trường hợp góc" dễ bị bỏ qua. Những vấn đề này dễ khắc phục hơn khi được phát hiện trong giai đoạn thiết kế thay vì sau khi bắt đầu triển khai.

- Đặc tả TLA+ có thể cung cấp một cách rõ ràng, ngắn gọn để truyền đạt một thiết kế. Nó giúp đảm bảo rằng các nhà thiết kế đồng ý về những gì họ đã thiết kế và nó cung cấp hướng dẫn có giá trị cho các kỹ sư triển khai và thử nghiệm hệ thống. Nó cũng có thể giúp người dùng hiểu hệ thống.

- Đặc tả TLA+ là một mô tả chính thức về những công cụ có thể được áp dụng để giúp tìm ra lỗi trong thiết kế và giúp kiểm tra hệ thống. Công cụ hữu ích nhất được viết ra cho mục đích này là trình kiểm tra mô hình TLC, được mô tả trong Chương 14.

Liệu lợi ích có biện minh cho nỗ lực viết đặc tả hay không phụ thuộc vào bản chất của dự án. Bản thân sự đặc tả không phải là mục đích cuối cùng; nó chỉ là một công cụ mà kỹ sư có thể sử dụng khi thích hợp.

7.2 Những gì cần chỉ định

Mặc dù chúng ta nói về việc xác định một hệ thống nhưng đó không phải là điều chúng ta làm. Đặc tả là một mô hình toán học về một cách nhìn cụ thể về một phần nào đó của hệ thống. Khi viết một đặc tả, điều đầu tiên bạn phải chọn chính xác là phần nào của hệ thống mà bạn muốn mô hình hóa. Đôi khi sự lựa chọn là hiển nhiên; thường thì không. Giao thức kết hợp bộ đệm của một máy tính đa bộ xử lý thực sự có thể được kết nối mật thiết với cách bộ xử lý thực hiện các lệnh. Việc tìm kiếm một bản tóm tắt mô tả giao thức kết hợp trong khi loại bỏ các chi tiết thực hiện lệnh có thể khó khăn. Nó có thể yêu cầu xác định giao diện giữa bộ xử lý và bộ nhớ không tồn tại trong thiết kế hệ thống thực tế.

Mục đích chính của đặc tả là giúp tránh lỗi. Bạn nên chỉ rõ những phần của hệ thống mà thông số kỹ thuật có nhiều khả năng phát hiện lỗi nhất. TLA+ đặc biệt hiệu quả trong việc phát hiện các lỗi tương tranh—những lỗi phát sinh thông qua sự tương tác của các thành phần không đồng bộ. Vì vậy, khi viết đặc tả TLA+ , bạn có thể sẽ tập trung nỗ lực vào các phần của hệ thống có nhiều khả năng xảy ra lỗi như vậy nhất. Nếu đó không phải là nơi bạn nên tập trung nỗ lực thì có lẽ bạn không nên sử dụng TLA+.

7.3 Hạt nguyên tử

Sau khi chọn phần nào của hệ thống để chỉ định, bạn phải chọn mức độ trừu tượng của đặc tả. Khía cạnh quan trọng nhất của mức độ trừu tượng là tính nguyên tử, sự lựa chọn những thay đổi nào của hệ thống được thể hiện dưới dạng một bước duy nhất của hành vi. Gửi tin nhắn trong một hệ thống thực tế bao gồm nhiều thao tác phụ, nhưng chúng tôi thường biểu diễn nó dưới dạng một bước duy nhất. Mặt khác, việc gửi và nhận tin nhắn thường được biểu diễn dưới dạng các bước riêng biệt khi chỉ định hệ thống phân tán.

Trình tự tương tự của các hoạt động hệ thống được thể hiện bằng một trình tự các bước ngắn hơn trong biểu diễn chi tiết thô hơn so với trình tự chi tiết hơn. Điều này hầu như luôn làm cho đặc tả chi tiết thô đơn giản hơn chi tiết chi tiết hơn. Tuy nhiên, đặc tả chi tiết hơn mô tả chính xác hơn hành vi của hệ thống thực tế. Một đặc tả chi tiết hơn có thể không tiết lộ được các chi tiết quan trọng của hệ thống.

Không có quy tắc đơn giản nào để quyết định mức độ nguyên tử. Tuy nhiên, có một cách nghĩ về mức độ chi tiết có thể hữu ích. Để mô tả nó, chúng tôi

cần toán tử thành phần hành động TLA+ "·". Nếu A và B là các hành động thì hành động A·B được thực hiện bằng cách thực hiện A trước rồi đến B dưới dạng một bước duy nhất. Chính xác hơn, A · B là hành động được xác định bằng cách cho $s \rightarrow t$ là bước A · B nếu tồn tại trạng thái u sao cho $s \rightarrow u$ là bước A và $u \rightarrow t$ là bước B.

Khi xác định mức độ nguyên tử, chúng ta phải quyết định xem nên biểu diễn việc thực hiện một thao tác dưới dạng một bước hay một chuỗi các bước, mỗi bước tương ứng với việc thực hiện một thao tác con. Hãy xem xét trường hợp đơn giản của một thao tác bao gồm hai thao tác phụ được thực hiện tuần tự, trong đó các thao tác phụ đó được mô tả bằng hai hành động R và L. (Việc thực thi R sẽ kích hoạt L và vô hiệu hóa R.) Khi việc thực thi thao tác đó được biểu thị bằng hai bước, mỗi bước đó là bước R hoặc bước L. Sau đó, thao tác được mô tả bằng hành động $R \lor L$. Khi việc thực hiện nó được thể hiện bằng một bước duy nhất, thao tác được mô tả bằng hành động R·L.[1] Đặt S2 là thông số kỹ thuật chi tiết hơn trong đó thao tác được thực hiện theo hai bước và đặt S1 là thông số kỹ thuật chi tiết hơn trong đó thao tác được thực hiện dưới dạng một bước R·L duy nhất. Để chọn mức độ nguyên tử, chúng ta phải chọn lấy S1 hay S2 làm thông số kỹ thuật. Hãy xem xét mối quan hệ giữa hai thông số kỹ thuật.

Chúng ta có thể chuyển đổi bất kỳ hành vi nào σ thỏa mãn S1 thành hành vi σ thỏa mãn S2 bằng cách thay thế mỗi bước $s \xrightarrow{R \cdot L} t$ bằng cặp bước $s \xrightarrow{R} u_{bạn} \xrightarrow{L} t$, đối với một số trạng thái u. Nếu chúng ta coi σ tương đương với σ thì chúng ta có thể coi S1 là một phiên bản được tăng cường của S2—một phiên bản cho phép ít hành vi hơn. Đặc tả S1 yêu cầu mỗi bước R phải được theo sau bởi bước L ngay lập tức, trong khi S2 cho phép các hành vi trong đó các bước khác nằm giữa bước R và L. Để chọn mức độ nguyên tử thích hợp, chúng ta phải quyết định xem những hành vi bổ sung được S2 cho phép có quan trọng hay không.

Các hành vi bổ sung được S2 cho phép sẽ không quan trọng nếu việc thực thi hệ thống thực tế mà chúng mô tả cũng được mô tả bởi các hành vi được S1 cho phép. Vì vậy, chúng ta có thể hỏi liệu mỗi hành vi τ thỏa mãn S2 có một hành vi tương ứng τ thỏa mãn S1, tức là, theo một nghĩa nào đó, tương đương với τ hay không. Một cách để xây dựng τ từ τ là biến đổi một chuỗi các bước

$$s \xrightarrow{R} u_1 \xrightarrow{A_1} u_2 \xrightarrow{A_2} u_3 \ldots u_n \xrightarrow{A_n} u_{n+1} \xrightarrow{L} t \qquad (7.1)$$

vào trình tự

$$s \xrightarrow{A_1} v_1 \ldots \xrightarrow{A_k} v_k \xrightarrow{R L} v_{k+1} \xrightarrow{A_{k+1}} v_{k+2} \ldots v_{n+1} \xrightarrow{A_n} t \qquad (7.2)$$

trong đó Ai là các hành động hệ thống khác có thể được thực thi giữa các bước R và L. Cả hai chuỗi đều bắt đầu ở trạng thái s và kết thúc ở trạng thái t, nhưng các trạng thái trung gian có thể khác nhau.

1Trên thực tế, chúng tôi mô tả thao tác bằng một hành động thông thường, giống như những hành động chúng tôi vừa viết, tương đương với R·L. Toán tử "·" hiếm khi xuất hiện trong thông số kỹ thuật thực tế. Nếu bạn muốn sử dụng nó, hãy tìm cách viết đặc tả tốt hơn; bạn có thể có thể tìm thấy một.

Khi nào có thể thực hiện được sự chuyển đổi như vậy? Câu trả lời có thể được đưa ra dưới dạng quan hệ giao hoán. Chúng ta nói rằng hành động A và B chuyển đổi nếu thực hiện chúng theo một trong hai thứ tự sẽ tạo ra kết quả như nhau. Về hình thức, A và B giao hoán nếu A · B tương đương với B · A. Một điều kiện đủ đơn giản cho tính giao hoán là hai hành động giao hoán nếu (i) mỗi hành động không thay đổi bất kỳ biến nào mà giá trị của nó có thể bị thay đổi bởi biến kia, và (ii) không bật hoặc tắt cái kia.
Không khó để thấy rằng chúng ta có thể biến đổi (7.1) thành (7.2) theo hai phương trình sau các trường hợp:

- R đi lại với mỗi Ai . (Trong trường hợp này, k = n.)
- L đi lại với mỗi Ai . (Trong trường hợp này, k = 0.)

Nói chung, nếu một phép toán bao gồm một chuỗi m hành động con, chúng ta phải quyết định có nên chọn biểu diễn chi tiết hơn O1 O2 hay không. . . Om hoặc loại thô hơn O1 · O2 · · · Om. Tổng quát hóa phép biến đổi từ (7.1) sang (7.2) là phép biến đổi một hành vi tùy ý thỏa mãn đặc tả chi tiết hơn thành một hành vi trong đó chuỗi O1, O2, . . . , Om bước tới nối tiếp bước kia. Sự chuyển đổi như vậy có thể thực hiện được nếu tất cả trừ một trong các hành động Oi giao hoán với mọi hành động khác của hệ thống. Tính giao hoán có thể được thay thế bằng các điều kiện yếu hơn, nhưng đó là trường hợp phổ biến nhất.

Bằng cách chuyển đổi các hành động và thay thế một chuỗi s O1 · · · Om t bước theo một bước O1 · · · Om duy nhất , bạn có thể chuyển đổi bất kỳ hành vi nào của đặc tả chi tiết hơn thành hành vi tương ứng của đặc tả chi tiết hơn.
Nhưng điều đó không có nghĩa là thông số kỹ thuật chi tiết thô hơn cũng tốt như thông số chi tiết mịn hơn. Trình tự (7.1) và (7.2) không giống nhau và trình tự bước Oi không giống với một bước O1 · · · Om . Việc bạn có thể coi hành vi được chuyển đổi tương đương với hành vi ban đầu và sử dụng đặc tả chi tiết hơn hay không, tùy thuộc vào hệ thống cụ thể mà bạn đang chỉ định và mục đích của đặc tả. Hiểu mối quan hệ giữa các thông số kỹ thuật chi tiết hơn và chi tiết hơn có thể giúp bạn lựa chọn giữa chúng; nó sẽ không đưa ra lựa chọn cho bạn.

7.4 Cấu trúc dữ liệu

Một khía cạnh khác của mức độ trừu tượng của đặc tả là độ chính xác mà nó mô tả cấu trúc dữ liệu của hệ thống. Ví dụ: đặc tả của giao diện chương trình có nên mô tả bố cục thực tế của các đối số của thủ tục trong bộ nhớ hay các đối số nên được trình bày một cách trừu tượng hơn?

Để trả lời câu hỏi như vậy, bạn phải nhớ rằng mục đích của thông số kỹ thuật là giúp phát hiện lỗi. Một mô tả chính xác về cách bố trí các đối số của thủ tục sẽ giúp ngăn ngừa các lỗi gây ra do hiểu lầm về cách bố trí đó, nhưng lại làm phức tạp thêm đặc tả của giao diện chương trình. Các

chi phí chỉ hợp lý nếu những lỗi đó có thể là một vấn đề thực sự và đặc tả TLA+ cung cấp cách tốt nhất để tránh chúng.

Nếu mục đích của đặc tả là phát hiện các lỗi gây ra bởi sự tương tác không đồng bộ của các thành phần thực thi đồng thời thì việc mô tả chi tiết về cấu trúc dữ liệu sẽ là một sự phức tạp không cần thiết. Vì vậy, có thể bạn sẽ muốn sử dụng các mô tả trừu tượng, cấp cao về cấu trúc dữ liệu của hệ thống trong đặc tả. Ví dụ, để chỉ định một giao diện chương trình, bạn có thể đưa ra các tham số không đổi để thể hiện các hành động gọi và quay lại từ một thủ tục—các tham số tương tự như Gửi và Trả lời của giao diện bộ nhớ được mô tả trong Phần 5.1 (trang 45).

7.5 Viết thông số kỹ thuật

Khi bạn đã chọn phần hệ thống để chỉ định và mức độ trừu tượng, bạn đã sẵn sàng bắt đầu viết đặc tả TLA+. Chúng ta đã thấy việc này được thực hiện như thế nào; Hãy xem lại các bước.

Đầu tiên, chọn các biến và xác định loại bất biến và vị từ ban đầu.
Trong quá trình thực hiện việc này, bạn sẽ xác định các thông số và giả định không đổi về chúng mà bạn cần. Bạn cũng có thể phải xác định một số hằng số bổ sung.

Tiếp theo, viết hành động ở trạng thái tiếp theo, tạo nên phần lớn thông số kỹ thuật. Phác thảo một vài hành vi mẫu có thể giúp bạn bắt đầu. Trước tiên, bạn phải quyết định cách phân rã hành động trạng thái tiếp theo thành sự tách rời của các hành động mô tả các loại hoạt động hệ thống khác nhau. Sau đó bạn xác định những hành động đó. Mục tiêu là làm cho các định nghĩa hành động trở nên nhỏ gọn và dễ đọc nhất có thể, điều này đòi hỏi phải cấu trúc chúng một cách cẩn thận. Một cách để giảm kích thước của một đặc tả là xác định các vị từ trạng thái và các hàm trạng thái được sử dụng trong một số định nghĩa hành động khác nhau. Khi viết các định nghĩa hành động, bạn sẽ xác định mô-đun tiêu chuẩn nào bạn cần và sẽ thêm câu lệnh mở rộng thích hợp. Bạn cũng có thể phải xác định một số toán tử hằng số cho cấu trúc dữ liệu mà bạn đang sử dụng.

Bây giờ bạn phải viết phần tạm thời của đặc tả. Nếu bạn muốn chỉ định các đặc tính sống động, bạn phải chọn các điều kiện công bằng, như được mô tả bên dưới trong Chương 8. Sau đó, bạn kết hợp vị từ ban đầu, hành động trạng thái tiếp theo và bất kỳ điều kiện công bằng nào mà bạn đã chọn vào định nghĩa của một thời gian duy nhất. công thức đó là đặc điểm kỹ thuật.

Cuối cùng, bạn có thể khẳng định các định lý về đặc tả. Nếu không có gì khác, bạn có lẽ muốn thêm một định lý về tính đúng kiểu.

7.6 Một số gợi ý thêm

Dưới đây là một số gợi ý linh tinh có thể giúp bạn viết thông số kỹ thuật tốt hơn.

Đừng quá thông minh.

Sự thông minh có thể làm cho thông số kỹ thuật khó đọc—và thậm chí sai. Công thức q = h
q có thể trông giống như một cách viết ngắn gọn, hay

(7.3) (h = Đầu(q)) (q = Đuôi(q))

Nhưng q = h q không chỉ khó hiểu hơn (7.3) mà nó còn sai.
Chúng ta không biết a b bằng bao nhiêu nếu a và b không phải là cả hai dãy, vì vậy chúng ta không biết liệu h = Head(q) và q = Tail(q) có phải là giá trị duy nhất của h và q thỏa mãn q hay không = h q . Có thể có các giá trị khác của h và q , không phải là các chuỗi, thỏa mãn công thức.

Nói chung, cách tốt nhất để xác định giá trị mới của biến v là dùng liên kết có dạng v = exp hoặc v exp, trong đó exp là hàm trạng thái—một biểu thức không có số nguyên tố.

Một loại bất biến không phải là một giả định.

Kiểu bất biến là một thuộc tính của một đặc tả, không phải là một giả định. Khi viết một đặc tả, chúng ta thường định nghĩa một kiểu bất biến. Nhưng đó chỉ là một định nghĩa; một định nghĩa không phải là một giả định. Giả sử bạn xác định một bất biến kiểu xác nhận rằng biến n thuộc loại Nat. Khi đó bạn có thể nghĩ rằng liên kết n > 7 trong một hành động khẳng định rằng n là một số tự nhiên lớn hơn 7. Thực tế không phải vậy. Công thức n > 7 chỉ khẳng định rằng n > 7. Sẽ hài lòng nếu n = √ 96 cũng như nếu n = 8. Vì chúng ta không biết liệu “abc” > 7 có đúng hay không nên thậm chí có thể thỏa mãn nếu n = “abc”. Ý nghĩa của công thức không thay đổi chỉ vì bạn đã xác định một bất biến kiểu xác nhận n
Nat.

Nói chung, bạn có thể muốn mô tả giá trị mới của biến x bằng cách xác nhận một số thuộc tính của x. Tuy nhiên, hành động ở trạng thái tiếp theo sẽ ngụ ý rằng x là một phần tử của một tập hợp phù hợp nào đó. Ví dụ: một đặc tả có thể xác định2

```
Hành động1  = (n > 7)   . . .
Hành động2  = (n 6)   . . .
Kế tiếp     = (n   Nat)   (Hành động1   Hành động2)
```

2Một cách tiếp cận khác là xác định Tiếp theo bằng Hành động1 Hành động2 và đặt thông số kỹ thuật là Ban đầu [Tiếp theo]... (n Nat). Nhưng tốt hơn hết bạn nên sử dụng biểu mẫu đơn giản Init [Next]... để biết thông số kỹ thuật.

Đừng quá trừu tượng.

Giả sử người dùng tương tác với hệ thống bằng cách gõ trên bàn phím. Chúng ta có thể mô tả sự tương tác một cách trừu tượng bằng một biến typ và một tham số toán tử KeyStroke, trong đó hành động KeyStroke(“a”, typ, typ) đại diện cho người dùng gõ một “a”. Đây là phương pháp chúng tôi đã sử dụng khi mô tả giao tiếp giữa bộ xử lý và bộ nhớ trong mô-đun MemoryInterface trên trang 48.

Một mô tả cụ thể hơn là để kbd biểu thị trạng thái của bàn phím, có thể đặt kbd = {} có nghĩa là không có phím nào được nhấn và kbd = {“a”} có nghĩa là phím a được nhấn. Việc gõ chữ a được thể hiện bằng hai bước, bước [kbd = {}] [kbd = {“a”}] thể hiện việc nhấn phím a và bước [kbd = {“a”}] [bước kbd = {}] thể hiện sự phát hành của nó. Đây là cách tiếp cận mà chúng tôi đã thực hiện trong phần đặc tả giao diện không đồng bộ của Chương 3.

Giao diện trừu tượng đơn giản hơn; việc gõ chữ a được thể hiện bằng một bước KeyStroke(“a”, typ, typ) thay vì một cặp bước. Tuy nhiên, việc sử dụng cách biểu diễn cụ thể khiến chúng ta tự nhiên đặt ra câu hỏi: điều gì sẽ xảy ra nếu người dùng nhấn phím a và trước khi thả nó ra, nhấn phím b? Điều đó thật dễ dàng để mô tả bằng biểu diễn cụ thể. Trạng thái nhấn cả hai phím là kbd = {“a”, “b”}.
Việc nhấn và nhả một phím được thể hiện đơn giản bằng hai hành động

Nhấn(k) = kbd = kbd {k} Phát hành(k) = kbd = kbd \ {k}

Khả năng nhấn hai phím không thể được thể hiện bằng giao diện trừu tượng đơn giản. Để diễn đạt nó một cách trừu tượng, chúng ta sẽ phải thay thế tham số KeyStroke bằng hai tham số PressKey và ReleaseKey, đồng thời chúng ta sẽ phải thể hiện rõ ràng thuộc tính là một khóa không thể được giải phóng cho đến khi nó được ấn xuống và ngược lại. Việc trình bày cụ thể hơn thì đơn giản hơn.

Chúng ta có thể quyết định rằng chúng ta không muốn xem xét khả năng hai phím bị ấn xuống và chúng ta thích cách biểu diễn trừu tượng hơn. Nhưng đó phải là một quyết định có ý thức. Sự trừu tượng của chúng ta không nên làm chúng ta mù quáng trước những gì có thể xảy ra trong hệ thống thực tế. Khi nghi ngờ, sẽ an toàn hơn khi sử dụng một biểu diễn cụ thể mô tả chính xác hơn hệ thống thực. Bằng cách đó, bạn sẽ ít có khả năng bỏ qua các vấn đề thực sự hơn.

Đừng cho rằng các giá trị trông khác nhau là không bằng nhau.

Các quy tắc của TLA+ không ngụ ý rằng 1 = “a”. Nếu hệ thống có thể gửi tin nhắn dưới dạng chuỗi hoặc số, hãy biểu thị tin nhắn dưới dạng bản ghi có trường loại và giá trị— ví dụ:

[loại “Chuỗi”, giá trị “a”] hoặc [loại “Nat”, giá trị 1]

Chúng tôi biết rằng hai giá trị này khác nhau vì chúng có các trường loại khác nhau.

Di chuyển định lượng ra bên ngoài.

Thông số kỹ thuật thường dễ đọc hơn nếu được di chuyển ra ngoài các liên kết và được di chuyển ra ngoài các liên từ. Ví dụ, thay vì

```
Hướng lên          =    e    Thang máy :  . . .
Xuống   =    e    Thang máy : . . .
Di chuyển   = Lên    Xuống
```

thường tốt hơn là viết

```
Lên (e)          =  . . .
Xuống(e)   =       . . .
Di chuyển          =    e    Thang máy : Lên(e)    Xuống(e)
```

Chỉ ưu tiên những gì bạn muốn nói đến.

Khi viết một hành động, hãy cẩn thận khi đặt các số nguyên tố. Biểu thức f [e] bằng f [e]; nó chỉ bằng f [e] nếu e = e, điều này không cần đúng nếu biểu thức e chứa các biến. Hãy đặc biệt cẩn thận khi bắt đầu một toán tử có định nghĩa chứa một biến. Ví dụ: giả sử x là một biến và op được xác định bởi

```
op(a)   =     x + một
```

Khi đó op(y) bằng (x+y) , bằng x +y , trong khi op(y) bằng x+y . Không có cách nào để sử dụng op và viết biểu thức x +y. (Viết op (y) không được vì nó không hợp lệ—bạn chỉ có thể viết nguyên tố một biểu thức chứ không phải toán tử.)

Viết bình luận dưới dạng bình luận.

Đừng đưa ý kiến vào chính đặc điểm kỹ thuật. Tôi đã thấy mọi người viết những điều như định nghĩa hành động sau:

```
A   =      x ≥ 0
              . . .
              x < 0
               sai
```

Phân biệt thứ hai có nghĩa là người viết dự định không kích hoạt A khi x < 0. Nhưng phân biệt đó hoàn toàn dư thừa, vì F false bằng sai và F false bằng F với mọi công thức F. Vì vậy, phân biệt thứ hai sự tách biệt của định nghĩa chỉ phục vụ như một hình thức bình luận. Tốt hơn là nên viết

```
A   =    x ≥ 0          A không được kích hoạt nếu x < 0
            . . .
```

7.7 Thời điểm và cách thức chỉ định

Thông số kỹ thuật thường được viết muộn hơn mức cần thiết. Các kỹ sư thường bị hạn chế nghiêm trọng về thời gian và họ có thể cảm thấy rằng việc viết một đặc tả sẽ làm họ chậm lại. Chỉ sau khi một thiết kế trở nên phức tạp đến mức họ cần trợ giúp để hiểu nó thì hầu hết các kỹ sư mới nghĩ đến việc viết một đặc tả chính xác.

Viết một đặc tả giúp bạn suy nghĩ rõ ràng. Suy nghĩ rõ ràng là khó; chúng ta có thể sử dụng tất cả sự giúp đỡ mà chúng ta có thể nhận được. Việc đưa thông số kỹ thuật trở thành một phần của quá trình thiết kế có thể cải thiện thiết kế.

Tôi đã mô tả cách viết một đặc tả giả định rằng thiết kế lại hệ thống đã tồn tại. Nhưng tốt hơn hết bạn nên viết đặc tả khi hệ thống đang được thiết kế. Thông số kỹ thuật ban đầu sẽ không đầy đủ và có thể không chính xác. Ví dụ: đặc tả ban đầu của bộ nhớ đệm ghi ở Mục 5.6 (trang 54) có thể bao gồm định nghĩa

```
RdMiss(p)   = Xếp hàng yêu cầu ghi giá trị từ bộ nhớ vào bộ đệm của p.
  Một số điều kiện kích hoạt phải được gắn liền
  ở đây.    memQ = Nối (memQ, buf [p])          Nối yêu cầu vào memQ.
  ctl = [ctl ngoại trừ ![p] = “?”]               Đặt ctl[p] thành giá trị sẽ được xác định sau.
  không thay đổi memInt, wmem, buf , cache
```

Một số chức năng hệ thống lúc đầu sẽ bị bỏ qua; nó có thể được đưa vào sau bằng cách thêm các phân cách mới vào hành động trạng thái tiếp theo. Các công cụ có thể được áp dụng cho các thông số kỹ thuật sơ bộ này để giúp tìm ra lỗi thiết kế.