Kiểu dữ liệu - Đặc tả trừu tượng

Chủ biên/Biên tập Gunther Schadow

gunther@aurora.rg.iupui.edu
Viện chăm sóc sức khỏe Regenstrief

Biên tập Paul Biron

paul.v.biron@kp.org
Kaiser Permanente, Nam California

Biên tập Lloyd McKenzie

lmckenzi@ca.ibm.com
Dịch vụ toàn cầu IBM

Biên tập Grahame Grieve

grahame@kestral.com.au
Công ty Máy Tính Kestral

Biên tập Doug Pratt

Douglas.Pratt@siemens.com
Siemens

Xuất bản lần cuối: 26/09/2005 4:12 chiều

MỤC LỤC

Lời nói đầu

i. Lời nói đầu

Tài liệu này đặc tả các Kiểu dữ liệu HL7 Phiên bản 3 trên một lớp trừu tượng, không phụ thuộc vào việc biểu diễn. Cụm từ "không phụ thuộc vào việc biểu diễn" có nghĩa là không phụ thuộc cả về cú pháp trừu tượng và triển khai trong bất kỳ công nghệ triển khai cụ thể nào.

Tài liệu này đi kèm với Đặc tả Công nghệ triển khai (ITS). Các tài liệu ITS có thể coi như một bản tóm tắt ngắn gọn nhưng đầy đủ các kiểu dữ liệu mà trên thực tế hướng nhiều hơn tới việc triển khai trong một công nghệ triển khai cụ thể.

Các bảng từ vựng có trong đặc tả này liệt kê các nội dung hiện thời của các miền từ vựng để người đọc dễ tham khảo. Tuy nhiên, tại bất kỳ thời điểm nào, nguồn quy phạm của các miền từ vựng này là các bảng từ vựng trong cơ sở dữ liệu của RIM. Với một số miền từ vựng lớn, chỉ có thể trình bày một phần các giá trị. Có thể tham khảo miền đầy đủ trong các bảng từ vựng bằng cách tìm kiếm tên miền gắn với bảng đó trong các bảng từ vựng của RIM.

ii. Lời cảm ơn

Đặc tả này là kết quả của nhiều năm làm việc căng thẳng qua e-mail, các hội nghị, điện thoại và các cuộc thảo luận trực tiếp, và cả những lần hòa giải thống nhất bằng cách bỏ phiếu thông qua. Chúng tôi xin gửi lời cảm ơn tới rất nhiều cá nhân đã tham gia vào các thời điểm khác nhau trong quá trình thiết kế, thảo luận và nhận xét. Gunther Schadow (Viện Y Tế Regenstrief) là người đứng đầu nhóm làm việc đặc biệt này, và là tác giả chính của tài liệu này. Paul V. Biron (Kaiser Permanente), Doug Pratt (Siemens), Lloyd McKenzie (IBM), và Grahame Grieve (Công ty Máy tính Kestral) với vai trò đồng biên tập vào những thời điểm khác nhau. Những người có đóng góp to lớn về ý tưởng và hỗ trợ là Mark Tucker (Viện Regenstrief), George Beeler, Stan Huff (Hệ thống y tế Intermountain), cũng như Mike Henderson (Kaiser Permanente), Anthony Julian (Mayo), Joann Larson (Kaiser Permanente), Mark Shafarman (Hệ thống y tế Oacis), Wes Rishel (Tập đoàn Gartner), và Robin Zimmerman (Kaiser Permanente). Chúng tôi cũng xin cảm ơn Bob Dolin (Kaiser Permanente), Clem McDonald (Viện Regenstrief), Kai Heitmann (HL7 Đức), Rob Seliger (Sentillion), và Harold Solbrig (Mayo Clinic) vì những nhận xét có tính phê bình và những ý tưởng mới. Chúng tôi đã nhận được sự hỗ trợ vô cùng quan trọng từ các thành viên của nhóm làm việc đặc biệt này, Laticia Fitzpatrick (Kaiser Permanente), Matt Huges, Randy Marbach (Kaiser Permanente), Larry Reis (Hệ thống Wizdom), Carlos Sanroman (Kaiser Permanente), Greg Thomas (Kaiser Permanente). Lời cảm ơn còn xin được gửi tới James Case (Đại học California, Davis), Norman Daoust (Hệ thống y tế Partners), Irma Jongeneel (HL7 Hà Lan), Michio Kimura (HL7 Nhật Bản), John Molina (SMS), Richard Ohlmann (McKessonHBOC), David Rowed (HL7 Úc), và Klaus Veil (Macquarie Health Corp., HL7 Úc), vì đã chia sẻ chuyên môn của họ qua những câu hỏi quan trọng. Công trình này có được là nhờ Viện Y Tế Regenstrief.

iii. Vấn đề nổi bật

Khuôn khổ hợp chuẩn để ràng buộc các kiểu dữ liệu

1. Giới thiệu

1.1 Kiểu dữ liệu là gì?

Mọi phần tử dữ liệu đều có một kiểu dữ liệu. Các kiểu dữ liệu xác định ý nghĩa (ngữ nghĩa) của các giá trị dữ liệu mà có thể được gán cho một phần tử dữ liệu. Trao đổi dữ liệu có nghĩa đòi hỏi rằng chúng ta biết định nghĩa của các giá trị được trao đổi. Điều này đúng đối với các "giá trị" phức tạp như bản tin nghiệp vụ cũng như đối với các giá trị đơn giản hơn chẳng hạn như các chuỗi ký tự hay các số nguyên.

Theo tiêu chuẩn ISO 11404, một kiểu dữ liệu là "một tập hợp các giá trị riêng biệt, được thiết lập tính đặc trưng bởi đặc tính của các giá trị đó và bởi các thao tác trên các giá trị đó." Một kiểu dữ liệu có phần khái niệm và mở rộng. Về phần khái niệm, kiểu dữ liệu định nghĩa các đặc tính mà mỗi giá trị dữ liệu thuộc kiểu đó thể hiện. Về phần mở rộng, các kiểu dữ liệu có một tập hợp các giá trị dữ liệu thuộc kiểu dữ liệu đó ("tập hợp giá trị" của kiểu).

Các đặc tính ngữ nghĩa của các kiểu dữ liệu được ISO 11404 gọi là "các đặc tính của những giá trị và [...] các thao tác trên các giá trị đó." Một đặc tính ngữ nghĩa của một kiểu dữ liệu được gọi bằng một cái tên và có một giá trị cho mỗi giá trị dữ liệu. Giá trị của đặc tính của một giá trị dữ liệu bản thân nó phải là một giá trị do một kiểu dữ liệu định nghĩa - không có giá trị dữ liệu nào tồn tại mà lại không được định nghĩa bởi một kiểu dữ liệu.

Do đó, các kiểu dữ liệu là các khối xây dựng cơ bản được dùng để tạo nên bất kỳ một ý nghĩa nào ở cấp cao hơn: các bản tin, các tài liệu hồ sơ bệnh án trên máy tính, hay các đối tượng nghiệp vụ và các giao dịch của chúng. Vậy thì, đâu là sự khác biệt giữa một kiểu dữ liệu và một bản tin, tài liệu, hay đối tượng nghiệp vụ? **Các giá trị kiểu dữ liệu đại diện cho chính nó, giá trị là trên hết, không phải định danh hay trạng thái hay thay đổi trạng thái được định nghĩa cho một giá trị dữ liệu.** Ngược lại trong các đối tượng nghiệp vụ, chúng ta dò theo trạng thái và định danh; các đặc tính của một đối tượng giống hệt có thể thay đổi theo thời gian. Các giá trị dữ liệu thì không như vậy: một giá trị dữ liệu và các đặc tính của nó là không đổi. Ví dụ, số 5 luôn luôn là số 5, không có sự khác biệt nào giữa số 5 này và số 5 kia (không có định danh được phân biệt từ giá trị), số 5 không bao giờ đổi thành số 6 (không thay đổi trạng thái). Có thể coi giá trị dữ liệu là các đối tượng bất biến trong đó định danh là không quan trọng (định danh và bình đẳng là như nhau.)[1]

1.2 Biểu diễn các giá trị dữ liệu

Có thể biểu diễn giá trị dữ liệu thông qua nhiều ký hiệu khác nhau nhưng ý nghĩa của giá trị dữ liệu thì không bị ràng buộc với bất kỳ biểu diễn cụ thể nào.

Ví dụ, các số đếm (các số nguyên không âm) được định nghĩa - về mặt khái niệm - là một kiểu dữ liệu trong đó mỗi giá trị có một giá trị kế tiếp, trong đó 0 không

là giá trị kế tiếp của một giá trị số đếm nào khác. Dựa trên định nghĩa này, có thể định nghĩa phép cộng, nhân, và các phép toán khác. Bất kỳ biểu diễn nào thể hiện được các quy tắc được nêu trong phần định nghĩa khái niệm của kiểu dữ liệu số đếm đều là biểu diễn hợp lệ của các số đếm. Ví dụ cho các biểu diễn số đếm hợp lệ là chuỗi số thập phân, các túi bi ve, hay các vết vạch trên tường. Số năm được biểu diễn bởi từ "năm", bởi chữ số Ả Rập "5" hay chữ số La Mã "V". Cách biểu diễn không quan trọng miễn là nó tuân thủ định nghĩa ngữ nghĩa của kiểu dữ liệu.

Một ví dụ khác, kiểu dữ liệu Boolean được định nghĩa bởi phần mở rộng của nó, hai giá trị riêng biệt *true (đúng)* và *false (sai)* và các quy tắc phủ định và kết hợp các giá trị này trong các phép hội và tuyển. Biểu diễn các giá trị Boolean có thể là các từ "đúng" và "sai," "có" và "không," các số 0 và 1, hai ký hiệu bất kỳ riêng biệt với nhau. Cách biểu diễn kiểu dữ liệu không quan trọng miễn là nó tuân thủ định nghĩa ngữ nghĩa của kiểu dữ liệu.

Đặc tả này định nghĩa ngữ nghĩa, ý nghĩa của các kiểu dữ liệu HL7. **Đặc tả này chỉ bàn về ngữ nghĩa, độc lập với các vấn đề về biểu diễn và thao tác hay các công nghệ triển khai cụ thể.** Các tiêu chuẩn bổ sung để biểu diễn các giá trị dữ liệu được định nghĩa ở đây đang được sử dụng để định nghĩa cho các phương thức tiếp cận công nghệ khác nhau. Các tiêu chuẩn này được gọi là "Đặc tả công nghệ triển khai" (ITS.) Các ITS này xác định cách các giá trị được biểu diễn tuân theo các định nghĩa ngữ nghĩa của đặc tả này, đặc tả này có thể bao gồm cú pháp cho các ký tự hoặc các phương thức biểu diễn nhị phân, và các quy trình máy tính để điều chỉnh việc biểu diễn các giá trị dữ liệu. Ý nghĩa của các biểu diễn ITS được truyền thông, tạo ra, và xử lý trong các chương trình máy tính được xác định dựa trên tiêu chuẩn này, đặc tả kiểu dữ liệu về mặt ngữ nghĩa.

1.3 Đặc tính của giá trị dữ liệu

Các giá trị dữ liệu có các đặc tính được định nghĩa bởi kiểu dữ liệu của chúng. Các "trường dữ liệu" "kiểu dữ liệu tổng hợp" là ví dụ phổ biến nhất cho các đặc tính này. Tuy nhiên, nhìn chung nên coi đặc tính của giá trị dữ liệu như các vị từ logic hoặc các hàm toán học; hay nói một cách đơn giản hơn nhưng vẫn chính xác, các đặc tính là các câu hỏi một người có thể nêu ra về một giá trị dữ liệu để nhận về câu trả lời là một giá trị dữ liệu khác.

Một đặc tính được gọi bằng tên của nó. Ví dụ, kiểu dữ liệu *integer (số nguyên)* có thể có một đặc tính được đặt tên là "ký hiệu." Một đặc tính có một miền, vốn là tập hợp các giá trị "câu trả lời" có thể. Tập hợp các giá trị "câu trả lời" có thể được xác định bởi kiểu dữ liệu của đặc tính, nhưng miền của một đặc tính có thể là tập con của một tập hợp giá trị của kiểu dữ liệu.

Một đặc tính cũng có thể có các tham số, là thông tin bổ sung phải cung cấp cùng với một câu hỏi để có được câu trả lời chính xác. Ví dụ, một đặc tính quan trọng

của một số nguyên là một số nguyên *cộng* một số nguyên khác cho ra một số nguyên khác, vì vậy đặc tính cộng của một số nguyên cần một tham số: số nguyên khác.

Cho dù các đặc tính ngữ nghĩa có tham số hay không không phải là một sự phân biệt thích hợp về cơ bản. Đặc tính ngữ nghĩa không có đối số của một kiểu dữ liệu không nhất thiết là một "trường dữ liệu" của một kiểu dữ liệu "kết hợp". Ví dụ, đối với các giá trị số nguyên, chúng ta có thể xác định đặc tính *is-zero (là-0)* có giá trị Boolean là true (*đúng)* khi số đó là 0 và false (*sai)* khi số đó không phải là 0. Điều này không có nghĩa là *is-zero (là-0)* phải là một thành phần rõ ràng của một biểu diễn số nguyên bất kỳ.

Đặc tính ngữ nghĩa của kiểu dữ liệu có tham số không có khái niệm hoạt động cụ thể như "gọi thủ tục," "truyền tham số," "trả giá trị," "trả về trường hợp ngoại lệ (hoặc là trường hợp phát sinh lỗi)," v.v. Đây đều là các khái niệm của việc triển khai hệ thống máy tính của các kiểu dữ liệu – nhưng các khái niệm hoạt động này không phù hợp cho ngữ nghĩa của các kiểu dữ liệu.

Đặc tả này chỉ thảo luận về ngữ nghĩa của các kiểu dữ liệu. Đặc tả này không đề cập về cú pháp biểu diễn giá trị (dù chỉ là một cú pháp trừu tượng), cũng không đề cập về giao diện hoạt động cho các giá trị dữ liệu.

1.4 Sự cần thiết của tính trừu tượng

Vì sao đặc tả này coi trọng việc có tính trừu tượng với cú pháp biểu diễn cũng như triển khai hoạt động?

HL7 cần loại đặc tả kiểu dữ liệu ngữ nghĩa trừu tượng này vì một mục đích hết sức thực tế. Một đặc điểm thiết kế quan trọng của HL7 phiên bản 3 là tính mở của nó đối với các cách biểu diễn và công nghệ triển khai. Tất cả các đặc tả HL7 phiên bản 3 có nhiệm vụ phải được thể hiện dưới hình thức độc lập với cách biểu diễn và công nghệ triển khai cụ thể. HL7 hiểu rằng, mặc dù đôi khi một vài cách biểu diễn và công nghệ triển khai có thể phổ biến hơn các cách biểu diễn và công nghệ triển khai khác, nhưng công nghệ luôn thay đổi - và với công nghệ luôn thay đổi, việc biểu diễn các giá trị dữ liệu cũng sẽ thay đổi. Các tiêu chuẩn HL7 chủ yếu hướng tới thông tin trong lĩnh vực y tế, không phụ thuộc vào công nghệ hỗ trợ thông tin này. HL7 kỳ vọng rằng đặc tả được định nghĩa độc lập với công nghệ ngày nay sẽ tiếp tục tỏ ra hữu dụng, ngay cả sau "chuyển đổi mô hình" công nghệ tiếp theo.

Vấn đề kiểu dữ liệu có mối liên hệ mật thiết với công nghệ triển khai hơn so với hầu hết các tiêu chuẩn thông tin HL7 khác - và theo đó tiềm ẩn nguy cơ rằng chúng ta sẽ định nghĩa các kiểu dữ liệu theo cách quá phụ thuộc vào các công nghệ triển khai hiện nay.

Đại đa số các tiêu chuẩn HL7 là về các đối tượng nghiệp vụ phức tạp. Các đối tượng nghiệp vụ phức tạp với rất nhiều thuộc tính thông tin có thể được xác định như

cú pháp trừu tượng, trong đó các thành phần cuối cùng được định nghĩa dựa trên kiểu dữ liệu. Ngược lại, định nghĩa kiểu dữ liệu dựa trên cú pháp trừu tượng là không mấy hữu ích do các thành phần của các cấu trúc cú pháp trừu tượng này vẫn sẽ phải có kiểu dữ liệu.[2]

Vì sao đặc tả này lại vòng vo như vậy? Vì sao kiểu dữ liệu "ANY" được định nghĩa dựa trên các chuyên biệt hóa của chính nó?

Đặc tả này cần độc lập với bất kỳ triển khai cụ thể nào, và do đó nó mang tính trừu tượng, và không hướng đến việc có thể triển khai được. Trong trường hợp này, tính vòng vo không phải là vấn đề, do nó không tạo ra bất kỳ sự không chắc chắn nào về nội dung của đặc tả này.

Vì sao đặc tả này không định nghĩa một tập hợp các kiểu dữ liệu nguyên thủy dựa trên đó có thể định nghĩa các kiểu dữ liệu kết hợp đơn giản là các cú pháp trừu tượng?

Bất kỳ triển khai cụ thể nào đối với các tiêu chuẩn HL7 cuối cùng cũng phải sử dụng các kiểu dữ liệu được xây dựng cho công nghệ triển khai đó. Vì vậy, chúng ta cần một sự liên hệ hết sức linh hoạt giữa các kiểu dữ liệu trừu tượng của HL7 và các kiểu dữ liệu được xây dựng cho một công nghệ triển khai cụ thể bất kỳ. Với một đặc tả ngữ nghĩa, một Đặc tả công nghệ triển khai (ITS) có thể tuân thủ bằng cách đơn giản là chỉ ra mối liên hệ giữa cấu trúc của công nghệ nó đặc tả và ngữ nghĩa kiểu dữ liệu HL7 phiên bản 3. Việc xét một kiểu dữ liệu là nguyên thủy hay tổng hợp là không phù hợp từ góc độ ngữ nghĩa, và câu trả lời có thể khác biệt đối với các công nghệ triển khai khác nhau.

Ví dụ, tiêu chuẩn này chỉ rõ một chuỗi ký tự là một kiểu dữ liệu với nhiều đặc tính (ví dụ, tập hợp ký tự, ngôn ngữ, v.v.) Tuy nhiên, trong nhiều Công nghệ triển khai, các chuỗi ký tự là các kiểu dữ liệu lớp đầu tiên nguyên thủy. Chúng tôi khuyến khích sử dụng các kiểu dữ liệu bản địa này thay vì một cấu trúc mà biểu diễn thiếu hiệu quả tất cả các đặc tính ngữ nghĩa là "thành phần." Đặc tả này chỉ yêu cầu rằng các đặc tính được xác định cho các giá trị dữ liệu bằng cách nào đó có thể được suy ra từ bất cứ biểu diễn nào được chọn, không quan trọng việc các giá trị này được biểu diễn như thế nào. Dù là "nguyên thủy" hay "kết hợp", với ít hay nhiều "thành phần", là "trường dữ liệu" hay "phương pháp" - tất cả đều không phù hợp.

Một ví dụ khác, một biểu diễn thập phân, một thanh ghi dấu phẩy động và một số nguyên có chia tỷ lệ đều là các cách biểu diễn bản địa có thể đối với các số thực của các công nghệ triển khai khác nhau. Một vài cách biểu diễn có những đặc tính mà các cách khác không có. Ví dụ, số nguyên chia tỷ lệ có độ chính xác cố định và một khoảng tương đối hẹp. Giá trị dấu phẩy động có độ chính xác biến thiên và một khoảng rộng, nhưng các giá trị dấu phẩy động mất bất kỳ thông tin nào về độ chính xác. Biểu diễn thập phân có độ chính xác biến thiên và duy trì thông tin về độ chính

xác (nhưng lại chậm xử lý.) Ngữ nghĩa kiểu giá trị phải độc lập với tất cả các đặc tính ngẫu nhiên này của các cách biểu diễn khác nhau, và phải định nghĩa được các đặc tính cần thiết mà bất kỳ công nghệ nào cũng có thể biểu diễn được.

1.5 Sự cần thiết của Tiêu chuẩn kiểu dữ liệu HL7

Vì sao HL7 cần tiêu chuẩn kiểu dữ liệu của riêng nó? Vì sao HL7 không thể chỉ đơn giản chấp nhận một tiêu chuẩn do người khác định nghĩa?

Như đã nói ở phần trước, tất cả các công nghệ triển khai HL7 đều có một hệ thống kiểu dữ liệu nào đó, nhưng có sự khác biệt giữa các hệ thống kiểu dữ liệu của các công nghệ triển khai. Thêm vào đó, nhiều hệ thống kiểu dữ liệu của các công nghệ triển khai không đủ mạnh để thể hiện các khái niệm quan trọng đối với lớp ứng dụng HL7.

Ví dụ, có ít công nghệ triển khai cung cấp các khái niệm đại lượng vật lý, độ chính xác, khoảng, thông tin còn thiếu, và tính không chắc chắn, là những khái niệm rất phù hợp trong tính toán khoa học và y tế.

Mặt khác, các công nghệ triển khai có những phân biệt không phù hợp theo quan điểm ngữ nghĩa trừu tượng, ví dụ, các số thực dấu phẩy tĩnh với dấu phẩy động; các số nguyên 8, 16, 32, hay 64 bit; ngày tháng với dấu ký hiệu thời gian.

Một số hệ thống kiểu dữ liệu đã được sử dụng làm đầu vào của đặc tả này. Các hệ thống này bao gồm các hệ thống kiểu của rất nhiều ngôn ngữ lập trình chính, bao gồm BASIC, Pascal, MODULA-2, C, C++, JAVA, ADA, LISP và SCHEME. Trong này cũng bao gồm các hệ thống kiểu của các công nghệ triển khai độc lập về ngôn ngữ, như Ký hiệu Cú pháp Trừu tượng 1 (ASN.1), Ngôn ngữ Định nghĩa Giao diện (IDL) của Nhóm Quản lý Đối tượng (OMG) và Ngôn ngữ Ràng buộc Đối tượng (OCL), SQL 92 và SQL 99, các kiểu dữ liệu độc lập về ngôn ngữ ISO 11404, và các kiểu dữ liệu XML Schema phần 2. Các kiểu dữ liệu có liên quan đến tiêu chuẩn y tế cũng được cân nhắc, trong đó có HL7 phiên bản 2.x, các kiểu được sử dụng bởi các bản tin CEN TC 251 và Kiến trúc Hồ sơ Y tế Điện tử (EHCRA) và DICOM.

1.6 Yêu cầu

Các kiểu dữ liệu được mô tả trong đặc tả này được thiết kế để đáp ứng một số yêu cầu, bao gồm:

- Cân nhắc về mô hình hóa
- Cân nhắc về triển khai
- Tính tương thích với các tiêu chuẩn kiểu dữ liệu khác
- Các yêu cầu về chức năng được xác định trong các tiêu chuẩn HL7 khác có sử dụng kiểu dữ liệu

Trong số này, cân nhắc cuối cùng là quan trọng nhất. Các kiểu dữ liệu này được thiết kế để cung cấp các chức năng được yêu cầu trong suốt các tiêu chuẩn HL7. Các yêu cầu này không phải lúc nào cũng tương thích, và trong suốt đặc tả này có những chỗ trong đó có một vài đặc điểm thiết kế nhất định chưa hẳn là tối ưu cho một trong số 4 cân nhắc liệt kê ở trên. Ở những chỗ này, các yêu cầu dẫn tới đặc điểm thiết kế này được mô tả trong phần Yêu cầu. Các phần yêu cầu này chỉ mang tính thông tin, không mang tính quy phạm.

Yêu cầu

Mô hình Thông tin Tham chiếu xác định một số lớp tham chiếu mà tất cả các mô hình thông tin miền dựa trên. Mỗi lớp tham chiếu này có một loạt các thuộc tính có một kiểu được gán cho. Khi các lớp tham chiếu được sử dụng trong (được sao chép sang) các mô hình miền, các kiểu trong các lớp tham chiếu có thể bị thay thế bởi các kiểu khác để làm rõ và ràng buộc việc sử dụng thuộc tính trong các lớp sao chép.

Đặc tả kiểu dữ liệu này phải xác định các quy tắc theo đó các kiểu dữ liệu có thể được thay thế theo cách này. Đặc tả này chọn sử dụng ẩn dụ chuyên biệt hóa làm cơ sở cho các quy tắc thay thế, vì đây là phương pháp được hiểu và sử dụng rộng rãi trong lý thuyết và thực tiễn, và vì các quy tắc này dễ hiểu và dễ quản lý hơn các lựa chọn thay thế. Việc sử dụng chuyên biệt hóa này có thể dẫn tới những thiết kế có vẻ không quen thuộc với một số người.

1.7 Các hình thức định nghĩa kiểu dữ liệu

Đặc tả này định nghĩa kiểu dữ liệu dưới vài hình thức, sử dụng mô tả bằng văn bản, các sơ đồ UML, bảng biểu, và một định nghĩa chính thức.

1.7.1 Ngôn ngữ định nghĩa kiểu dữ liệu chính thức

Một định nghĩa chính thức về kiểu dữ liệu được sử dụng để chỉ ra rõ ràng nhất ở mức có thể ngữ nghĩa của kiểu dữ liệu được đề xuất. Ngôn ngữ định nghĩa kiểu dữ liệu này được mô tả chi tiết trong Giới thiệu về Ngôn ngữ định nghĩa kiểu dữ liệu chính thức (DTDL) (§ 1.9). Các ngôn ngữ chính thức đưa ra những tuyên bố rõ ràng cần thiết và do đó dễ tiếp cận đối với một số lập luận chính thức về chứng minh hay bác bỏ. Tuy nhiên, tính súc tích của những tuyên bố chính thức này có thể gây khó hiểu cho con người. Vì vậy, tất cả các kết luận quan trọng có được từ các tuyên bố chính thức cũng được diễn giải bằng ngôn ngữ thông thường.

1.7.2 Bảng đặc tính

Để nhanh chóng có được một cái nhìn tổng quan, ngay từ phần đầu của nhiều kiểu dữ liệu đặc tả này đã có các bảng liệt kê các đặc tính "chính". Các đặc tính "chính" là một khái niệm có phần mơ hồ về các đặc tính dễ bị hiểu là các "trường dữ liệu" khi kiểu dữ liệu được triển khai như một hồ sơ, hoặc được kỳ vọng sẽ được sử dụng thường xuyên hơn. Các bảng này được cung cấp để dễ dàng có một cái nhìn tổng

quan về nội dung và mục đích của các kiểu dữ liệu. Không có yêu cầu nào về việc các đặc tính được liệt kê trong các bảng này phải được biểu diễn như các trường dữ liệu, và các bảng này **không** phải là các định nghĩa cú pháp trừu tượng.

Mỗi hàng trong các bảng đặc tính mô tả một đặc tính với các cột sau:

1. **Tên** - tên của đặc tính như đã được nêu trong định nghĩa chính thức. Đối với một số kiểu dữ liệu, trường tên của đặc tính đầu tiên có thể trống. Điều này có thể xảy ra với các kiểu dữ liệu được định nghĩa là phần mở rộng của các kiểu dữ liệu khác và khi nó không hữu dụng cho việc tóm tắt con để thể hiện bất kỳ đặc tính nào của cha.
2. **Kiểu** - kiểu dữ liệu của đặc tính đó.
3. **Định nghĩa** - một đoạn văn bản ngắn mô tả ý nghĩa của đặc tính.

1.7.3 Các sơ đồ ngôn ngữ mô hình hóa thống nhất (UML)

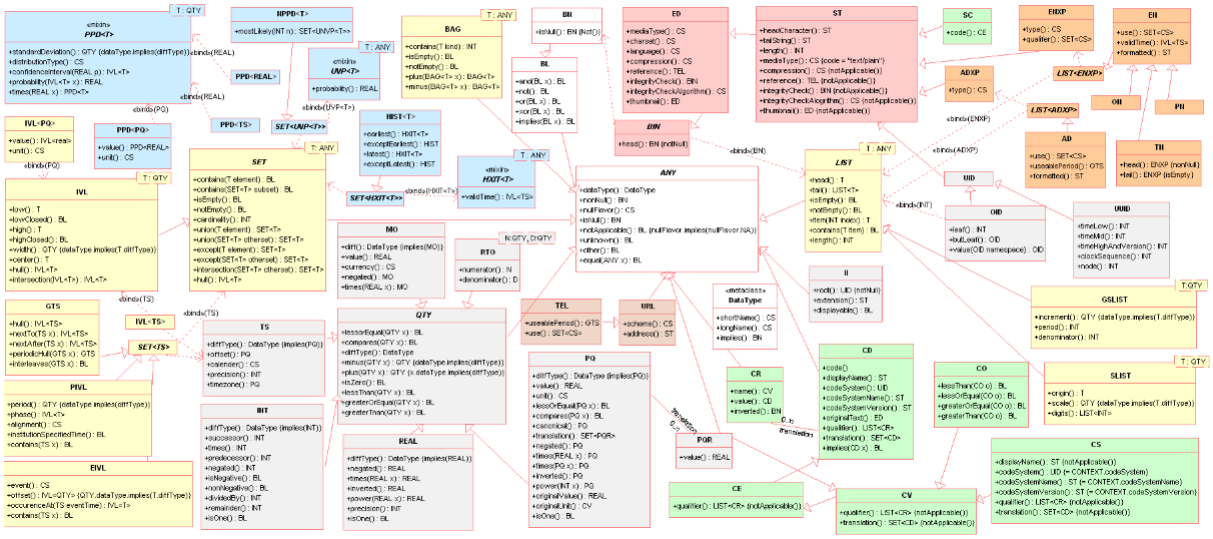
Ngôn ngữ Mô hình hóa Thống nhất (UML) được sử dụng để trình bày bằng đồ họa cách thức các kiểu dữ liệu liên hệ với nhau. Các kiểu dữ liệu được thể hiện như các lớp UML sử dụng tên ngắn cho lớp đó. Các đặc tính của kiểu được thể hiện như các thao tác UML. Các kiểu chung được thể hiện như các lớp tham số UML, với các liên kết thực hiện UML liên hệ với các kiểu cụ thể của chúng.

Không thể biểu diễn phần lớn các chi tiết của các khai báo kiểu dữ liệu trong biểu diễn UML. Vì vậy, nên sử dụng định nghĩa chính thức của các kiểu dữ liệu trong Ngôn ngữ Định nghĩa Kiểu Dữ liệu (DTDL) để đặc tả chi tiết các kiểu dữ liệu.

Một số ràng buộc từ DTDL được biểu diễn như các ràng buộc đối với thao tác. Khi các ràng buộc được thể hiện, chúng là các tuyên bố sẽ là đúng và được lấy từ đặc tả DTDL.

Các sơ đồ UML sử dụng khuôn mẫu "mixin" (trộn). Khuôn mẫu trộn áp dụng cho một lớp tham số, và biểu thị rằng lớp đó chuyên biệt hóa kiểu tham số và thể hiện tất cả các đặc tính của kiểu T ngoài các đặc tính của chính nó.

1.8 Tổng quan về các kiểu dữ liệu



Bảng 1. Tổng quan về các kiểu dữ liệu HL7 phiên bản 3

Tên	Ký hiệu	Mô tả
DataValue (Giá trị dữ liệu)	ANY	Xác định các đặc tính cơ bản của mọi giá trị dữ liệu. Đây là một loại dữ liệu trừu tượng, nghĩa là không có giá trị nào có thể chỉ là giá trị dữ liệu mà không thuộc về bất kỳ một kiểu dữ liệu cụ thể nào. Mọi kiểu dữ liệu cụ thể đều là một chuyên biệt hóa của kiểu Giá trị dữ liệu trừu tượng chung này.
Boolean	BL	*BL* đại diện cho các giá trị của hai giá trị luận lý (logic). Một giá trị *BL* có thể là *đúng* hoặc *sai*, hoặc như bất kỳ giá trị nào khác, có thể có giá trị NULL (rỗng)
BooleanNonNull (Boolean không Null)	BN	*BN* ràng buộc kiểu boolean sao cho giá trị của kiểu dữ liệu không thể là NULL. Kiểu dữ liệu này được tạo ra để sử dụng trong đặc tả kiểu dữ liệu khi việc sử dụng một giá trị null là không phù hợp (trường hợp bắt buộc phải có giá trị khác NULL)
Encapsulated Data (Dữ liệu được đóng	ED	Dữ liệu có mục đích chủ yếu là để cho người đọc hiểu hoặc cho các xử lý máy tính khác bên ngoài phạm vi của HL7. Nó bao gồm ngôn ngữ viết có định dạng

Tên	Ký hiệu	Mô tả
gói)		hoặc không có định dạng, dữ liệu đa phương tiện, hoặc các thông tin có cấu trúc do một tiêu chuẩn khác xác định (ví dụ, XML – chữ ký). Thay vì chứa bản thân dữ liệu, một *ED* cũng có thể chỉ chứa một đường dẫn tham chiếu (xem TEL.) Cần lưu ý rằng kiểu dữ liệu ST là một chuyên biệt hóa của kiểu dữ liệu *ED* khi Kiểu phương tiện được cố định là văn bản/đơn thuần.
Character String (Chuỗi ký tự)	ST	Kiểu dữ liệu chuỗi ký tự đại diện cho dữ liệu văn bản, chủ yếu được dùng cho máy xử lý (ví dụ, sắp xếp, truy vấn, lập chỉ mục, v.v.) Được dùng cho tên, ký hiệu, và các biểu thức chính thống.
Concept Descriptor (Bộ mô tả khái niệm)	CD	Một *CD* biểu diễn loại khái niệm bất kỳ, thông thường bằng cách gán cho một mã đã được xác định trong một hệ thống mã. Một *CD* có thể chứa văn bản hoặc cụm từ gốc dùng làm cơ sở để mã hóa và một hoặc nhiều dạng chuyển ngữ sang các hệ thống mã hóa khác nhau. Một *CD* cũng có thể chứa các từ hạn định để mô tả, ví dụ khái niệm "bàn chân trái" là một thuật ngữ hậu kết hợp được xây dựng từ mã chính là "BÀN CHÂN" và từ hạn định là "TRÁI". Trong các trường hợp có một giá trị ngoại lệ, kiể dữ liệu *CD* không cần chứa mã mà chỉ có văn bản gốc mô tả khái niệm.
Coded Simple Value (Giá trị đơn giản được mã hóa)	CS	Dữ liệu được mã hóa trong dạng hình thức đơn giản nhất, trong đó chỉ có mã không được xác định trước. Hệ thống mã và phiên bản hệ thống mã được cố định bởi ngữ cảnh mà giá trị *CS* xảy ra. *CS* được dùng cho các thuộc tính được mã hóa mà chỉ có một tập hợp giá trị do HL7 định nghĩa.

Tên	Ký hiệu	Mô tả
Coded Ordinal (Số thứ tự được mã hóa)	CO	Dữ liệu được mã hóa, trong đó hệ thống mã hóa mà từ đó mã tới được xếp thứ tự. Kiểu dữ liệu *CO* bổ sung ngữ nghĩa liên quan đến việc xếp thứ tự sao cho các mô hình sử dụng các miền đó có thể đưa vào sử dụng các phần tử mô hình có bao hàm các câu lệnh về thứ tự của các thuật ngữ trong một miền.
Coded With Equivalents (Được mã hóa có tương đương)	CE	Dữ liệu được mã hóa bao gồm một giá trị được mã hóa, không bắt buộc, (các) giá trị được mã hóa từ các hệ thống mã hóa khác mà có cùng định danh khái niệm đó. Kiểu dữ liệu này được sử dụng khi có thể có các mã thay thế cùng tồn tại.
Character String with Code (Chuỗi ký tự có mã)	SC	Một chuỗi ký tự có thể có một mã đi kèm. Phải luôn có đoạn văn bản nếu có mã. Mã thường là một mã địa phương.
Instance Identifier (Định danh thực thể)	II	Một định danh xác định duy nhất một vật thể hoặc đối tượng. Ví dụ như định danh đối tượng cho các đối tượng của sơ đồ HL7 RIM, số bệnh án, ID chỉ định, ID hạng mục catalog dịch vụ, Số định danh phương tiện giao thông (VIN), v.v. Định danh thực thể được xác định dựa trên các định danh đối tượng của ISO.
Telecommunication Address (Địa chỉ viễn thông)	TEL	Một số điện thoại (giọng nói hoặc fax), địa chỉ email, hoặc một địa chỉ định vị khác có thể liên lạc bằng một thiết bị viễn thông cho một tài nguyên. Địa chỉ được xác định như một dữ liệu Định vị tài nguyên toàn cầu (URL) được hạn định bởi đặc tả thời gian và các mã sử dụng, để giúp quyết định sử dụng địa chỉ nào tại một thời điểm và dành cho một mục đích nhất định.
Postal Address	AD	Địa chỉ gửi thư và địa chỉ nhà hay địa chỉ văn phòng. Là một chuỗi các phần của địa chỉ, như tên phố hoặc

Tên	Ký hiệu	Mô tả
(Địa chỉ bưu điện)		hòm thư bưu điện, thành phố, mã bưu điện, quốc gia, v.v.
Entity Name (Tên thực thể)	EN	Tên cho một người, tổ chức, địa điểm hay vật thể. Là một chuỗi các thành phần của tên, như tên hay họ, tiền tố, hậu tố, v.v. Ví dụ cho các giá trị tên thực thể là "Jim Bob Walton, con", "Health Level Seven, Inc.", "Hồ Tahoe", v.v. Một tên thực thể có thể chỉ đơn giản là một chuỗi ký tự hoặc có thể bao gồm một vài thành phần của tên thực thể, như "Jim", "Bob", "Walton", và "con", "Health Level Seven" và "Inc.", "Hồ" và "Tahoe".
Trivial Name (Tên thông thường)	TN	Một giới hạn của tên thực thể, mà thực ra là một chuỗi đơn giản được dùng để cho biết tên đơn giản của sự vật và địa điểm.
Person Name (Tên người)	PN	Kiểu dữ liệu EN được sử dụng khi thực thể được đặt tên cho một Người. Là một chuỗi các thành phần của tên, như tên hay họ, tiền tố, hậu tố, v.v. Một thành phần của tên là một mục giới hạn của thành phần tên thực thể mà chỉ chấp nhận các từ hạn định thành phần tên thực thể được sử dụng cho tên người. Do cấu trúc của tên thực thể chủ yếu được quyết định bởi các yêu cầu về tên người, mục giới hạn này là rất nhỏ.
Organization Name (Tên tổ chức)	ON	Một ON được sử dụng khi thực thể được đặt tên là một Tổ chức. Là một chuỗi các thành phần của tên.
Integer Number (Số nguyên)	INT	Các số nguyên (-1,0,1,2, 100, 3398129, v.v.) là các số chính xác do việc đếm và liệt kê tạo ra. Các số nguyên là rời rạc, tập hợp các số nguyên là vô hạn nhưng có thể đếm được. Không có bất kỳ giới hạn nào được áp đặt cho khoảng số nguyên. Hai trạng thái

Tên	Ký hiệu	Mô tả
		NULL được xác định cho vô cùng dương và vô cùng âm.
Real Number (Số thực)	REAL	Các phân số. Thường được sử dụng khi đo đạc, ước tính, hoặc tính toán các đại lượng từ các số thực khác. Cách biểu diễn tiêu biểu là số thập phân, trong đó số lượng các số thập phân quan trọng được gọi là độ chính xác.
Ratio (Tỷ lệ)	RTO	Một đại lượng được cấu tạo như thương số của một đại lượng tử số chia cho một đại lượng mẫu số. Các thừa số chung trong tử số và mẫu số không được tự động triệt tiêu. Kiểu dữ liệu *RTO* hỗ trợ độ chuẩn (ví dụ "1:128") và các đại lượng khác từ phòng thí nghiệm mà thực sự biểu diễn các tỷ lệ. Tỷ lệ không đơn giản là "các con số có cấu trúc", đặc biệt các giá trị đo huyết áp (ví dụ "120/60") không phải là tỷ lệ. Trong nhiều trường hợp, nên sử dụng kiểu dữ liệu REAL thay cho *RTO*.
Physical Quantity (Đại lượng vật lý)	PQ	Một đại lượng có đơn vị thể hiện một kết quả đo đạc.
Monetary Amount (Số tiền)	MO	Kiểu dữ liệu *MO* là một đại lượng biểu diễn số tiền trong một loại tiền tệ nào đó. Loại tiền tệ là đơn vị trong đó số tiền được thể hiện tại các vùng kinh tế khác nhau. Trong khi số tiền là loại đại lượng duy nhất (tiền), thì tỷ giá hối đoái giữa các đơn vị tiền khác nhau lại biến thiên. Đây là sự khác biệt chính giữa hai kiểu dữ liệu PQ và *MO*, và là lý do tại sao đơn vị tiền tệ lại không phải là đơn vị vật lý.
Point in Time	TS	Một đại lượng xác định một điểm trên trục thời gian tự nhiên. Thời điểm thường được biểu diễn ở dạng

Tên	Ký hiệu	Mô tả
(Thời điểm)		một biểu thức diễn giải lịch.
Set (Tập hợp)	SET	Một giá trị chứa các giá trị riêng biệt khác không theo một thứ tự cụ thể nào.
Sequence (Chuỗi)	LIST	Một giá trị chứa các giá trị rời rạc (nhưng không nhất thiết riêng biệt) khác theo một trật tự đã được xác định.
Bag (Bộ sưu tập)	BAG	Một bộ sưu tập không có thứ tự các giá trị, trong đó mỗi giá trị có thể xuất hiện nhiều hơn một lần trong bộ sưu tập.
Interval (Khoảng)	IVL	Một tập hợp các giá trị liên tiếp của một kiểu dữ liệu cơ bản theo thứ tự.
History (Lịch sử)	HIST	Một tập hợp các giá trị dữ liệu có đặc tính thời gian-hợp lệ và do đó tuân thủ kiểu HXIT. Thông tin lịch sử không bị giới hạn ở quá khứ; các giá trị dự kiến trong tương lai cũng có thể xuất hiện.
Uncertain Value – Probabilistic (Giá trị không chắc chắn – có xác suất)	UVP	Phần mở rộng kiểu dữ liệu chung được sử dụng để cho biết xác suất thể hiện độ tin cậy của thông tin về tính đúng đắn của giá trị được đưa ra.
Periodic Interval of Time (Khoảng thời gian định kỳ)	PIVL	Một khoảng thời gian định kỳ tái diễn. Kiểu dữ liệu *PIVL* có hai đặc tính, pha và chu kỳ. Pha xác định "nguyên mẫu khoảng" được lặp lại sau mỗi chu kỳ.
Event-Related	EIVL	Xác định khoảng thời gian định kỳ trong đó mỗi lần

Tên	Ký hiệu	Mô tả
Periodic Interval of Time (Khoảng thời gian định kỳ liên quan đến sự kiện)		tái diễn dựa trên các hoạt động của cuộc sống hàng ngày hoặc các sự kiện quan trọng khác có liên quan đến thời gian nhưng không được xác định hoàn toàn bởi thời gian.
General Timing Specification (Đặc tả thời gian tổng quát)	GTS	Một <dt-TS>, xác định thời gian của các sự kiện và hoạt động và các mẫu hợp lệ theo chu kỳ có thể tồn tại đối với một số loại thông tin, ví dụ số điện thoại (liên lạc ban đêm, ban ngày), địa chỉ (còn gọi là "chim tuyết", sống gần đường xích đạo vào mùa đông và xa đường xích đạo vào mùa hè) và giờ hành chính.
Parametric Probability Distribution (Phân bổ xác suất theo tham số)	PPD	Phần mở rộng kiểu dữ liệu chung xác định tính không chắc chắn của dữ liệu định lượng bằng cách dùng một hàm phân bổ và các tham số của nó. Ngoài các tham số cụ thể của phân bố, một giá trị trung bình (giá trị mong đợi) và một độ lệch chuẩn luôn được cung cấp để giúp duy trì một mức độ liên kết liên thông tối thiểu nếu các ứng dụng nhận không thể làm việc với một phân bổ xác suất nhất định.

1.9 Giới thiệu về ngôn ngữ Định nghĩa kiểu dữ liệu chính thức (DTDL)

LƯU Ý: Đây không phải là một đặc tả API. Mặc dù ngôn ngữ chính thức này có thể giống với một ngôn ngữ lập trình hay ngôn ngữ định nghĩa giao diện kết nối nào đó, nó không hướng đến định nghĩa chi tiết của các chương trình và các phương tiện triển khai khác. Các định nghĩa chính thức là một phần quy chuẩn của đặc tả này, nhưng ngôn ngữ cụ thể này không cần phải được triển khai hay sử dụng trong các hệ thống quy chuẩn; cũng như không cần tất cả các đặc tính ngữ nghĩa phải được triển khai hay sử dụng bởi các hệ thống quy chuẩn. Hoạt động nội bộ của các hệ thống, cách chúng triển khai các kiểu dữ liệu, **các chức năng và dịch vụ của chúng hoàn toàn nằm ngoài phạm vi của đặc tả này. Định nghĩa chính thức chỉ xác định ý nghĩa** của các giá trị dữ liệu bằng cách đưa ra các tuyên bố về việc nên trông đợi gì, về mặt lý thuyết, cách các giá trị này liên hệ và ứng xử.

Ngôn ngữ định nghĩa kiểu dữ liệu chính thức này[3] xác định:

- tên kiểu và tên ngắn;
- các giá trị được đặt tên của một phần mở rộng được liệt kê đầy đủ;
- đặc tính ngữ nghĩa, đơn phân, nhị phân, và các đặc tính cấp cao hơn;
- các bất biến, nghĩa là các ràng buộc đối với đặc tính;
- các chuyển đổi kiểu được chấp nhận;
- cú pháp của dạng chữ giá trị chuỗi ký tự (nếu có);

Định nghĩa một kiểu dữ liệu diễn ra theo hai bước. Đầu tiên, kiểu dữ liệu được khai báo. Việc khai báo khẳng định tên cho kiểu dữ liệu mới với một danh sách các tên, kiểu, và chữ ký của các đặc tính ngữ nghĩa của kiểu mới. Bước này khai báo, chứ không định nghĩa kiểu. Định nghĩa được thực hiện trong các câu lệnh logic về cái gì luôn đúng đối với các giá trị của kiểu này và các đặc tính của chúng (câu lệnh bất biến).

1.9.1 Khai báo

Mọi kiểu dữ liệu đều được khai báo theo hình thức bắt đầu với từ khóa **type (kiểu)**. Ví dụ, dưới đây là phần đầu của một khai báo cho kiểu dữ liệu Boolean có tên ngắn bí danh BL và chuyên biệt hóa kiểu dữ liệu ANY.[4]

```
type Boolean alias BL specializes ANY
  values(true, false)
{
  BL  not;
  BL  and(BL x);
};
```

Khai báo kiểu dữ liệu Boolean cũng chứa một mệnh đề-**giá trị** khai báo tập hợp giá trị hoàn chỉnh của Boolean (phần mở rộng của nó) như những thực thể được đặt tên. Các giá trị được đặt tên này cũng là dạng chữ chuỗi ký tự hợp lệ. Không có kiểu dữ liệu khác nào được định nghĩa trong đặc tả này có một tập hợp giá trị hữu hạn, đó là lí do vì sao mệnh đề-**giá trị** là duy nhất với Boolean. Trong ngôn ngữ chính thức được đánh dấu, tên giá trị sử dụng phông chữ in nghiêng.

Đoạn trong dấu ngoặc cong tiếp sau đoạn đầu chứa các khai báo về đặc tính ngữ nghĩa đúng với mọi giá trị của kiểu dữ liệu. Một dấu chấm phẩy kết thúc mỗi khai báo đặc tính; và một dấu chấm phẩy khác nằm sau dấu ngoặc cong đóng kết thúc khai báo kiểu dữ liệu.

Một khai báo đặc tính đề cập tới từ trái sang phải: (1) kiểu dữ liệu của miền giá trị của đặc tính, (2) tên đặc tính, và (3) một danh sách tham số tùy chọn. Danh sách tham số của một đặc tính được đặt trong dấu ngoặc đơn chứa một chuỗi các khai báo

tham số. Mỗi tham số được khai báo bằng tên kiểu dữ liệu và tên tham số. Các đặc tính ngữ nghĩa không có tham số không sử dụng danh sách tham số rỗng.[5]

Mệnh đề-**chuyên biệt hóa** có nghĩa là (a) thừa kế đặc tính từ chi tới loài, và (b) khả năng thay thế của các giá trị của kiểu loài cho các biến số của kiểu chi. Chuyên biệt hóa có thể bao gồm định nghĩa cho các đặc tính bổ sung và đặc tả về các ràng buộc với các đặc tính được thừa kế cho kiểu được chuyên biệt hóa.

Một ví dụ về thừa kế là: khi CD có mã đặc tính và CS chuyên biệt hóa CD, thì CS cũng có mã đặc tính này mặc dù isNull (là rỗng) không được liệt kê rõ ràng trong khai báo đặc tính của BL. Một ví dụ về khả năng thay thế là: khi một đặc tính được khai báo là thuộc kiểu dữ liệu CD, và CS chuyên biệt hóa CD, thì một giá trị của đặc tính này có thể thuộc kiểu CS. Nói cách khác, khả năng thay thế giống như gộp tất cả các giá trị của kiểu CS cũng là các giá trị của kiểu CD.[6]

Khai báo-**kiểu** có thể được định tính bằng các từ khóa **abstract (trừu tượng)**, **protected. (được bảo vệ.)**, hoặc **private (cá nhân)**. Kiểu trừu tượng là kiểu trong đó không có giá trị nào có thể chỉ thuộc loại này mà không thuộc về một chuyên biệt hóa cụ thể của kiểu trừu tượng. Kiểu protected là kiểu được sử dụng trong đặc tả này nhưng không đặc tính nào ngoài phạm vi đặc tả này được khai báo là thuộc kiểu được bảo vệ. Kiểu private là một trừu tượng "trợ giúp" nội bộ, được định nghĩa chỉ để phục vụ mục đích định nghĩa một khía cạnh ngữ nghĩa nào đó của các kiểu dữ liệu nhưng không được sử dụng dù là như kiểu đặc tính của một kiểu protected hay public khác.[7] (Chúng tôi cũng một lần sử dụng từ hạn định **cá nhân**. Các kiểu protected (cá nhân) chỉ được xác định để phục vụ định nghĩa chính thức của các kiểu khác và không được sử dụng dưới bất kỳ hình thức nào bên ngoài đặc tả này).

1.9.2 Câu lệnh bất biến

Khai báo đặc tính ngữ nghĩa, tên của chúng, các kiểu dữ liệu, và các đối số chỉ cung cấp những đầu mối cho biết kiểu dữ liệu mới có thể về cái gì. Định nghĩa thực sự nằm ở câu lệnh bất biến. Câu lệnh bất biến là các câu lệnh logic luôn đúng.

Trong suốt đặc tả này, câu lệnh bất biến được cung cấp theo một cú pháp chính thức nhưng cũng được diễn đạt bằng ngôn ngữ thông thường. Lợi ích của cú pháp chính thức đó là nó có thể được giải thích và định kiểu một cách rõ ràng. Lợi ích của câu lệnh được diễn đạt bằng ngôn ngữ thông thường đó làm chúng dễ hiểu hơn, đặc biệt đối với những người không được đào tạo để đọc ngôn ngữ chính thức.

Cú pháp chính thức giúp làm sắc nét thêm tính quyết đoán của đặc tả này. Tuy nhiên, trong một số trường hợp, ngữ nghĩa đầy đủ của một kiểu vượt quá những gì có thể được diễn đạt trong các câu lệnh bất biến. Sự kết hợp giữa ngôn ngữ thông thường và ngôn ngữ chính thức giúp đặc tả này thêm rõ ràng.

Các câu lệnh bất biến được hình thành nhờ sử dụng từ khóa **invariant (bất biến)**, từ này khai báo một hoặc nhiều biến số theo hình thức tương tự như một danh sách tham số của một đặc tính. Câu lệnh bất biến có thể chứa một mệnh đề **trong đó**, mệnh đề này ràng buộc các tham số cho toàn bộ nhóm bất biến. Nhóm bất biến được đặt trong dấu ngoặc cong. Nó chứa một danh sách các khẳng định và tất cả khẳng định này phải đúng.

```
invariant(BL x)
    where x.nonNull {
  x.and(true).equal(x);
};
```

Ngữ nghĩa của câu lệnh bất biến là một vị từ logic với một từ lượng hóa chung ("for all"/đối với tất cả).

Có thể đọc câu lệnh bất biến trên trong ngôn ngữ thông thường là "Đối với tất cả các giá trị Boolean x, trong đó x là non-NULL, x AND *true* bằng x là đúng." Nên đặt tên tất cả các đặc tính sao cho có thể đọc các khẳng định như đọc những câu văn thông thường.[8]

Không cần xác định danh sách tham số của một câu lệnh bất biến nếu không cần tham số.

```
invariant {
  true.not.equal(false);
  false.not.equal(true);
};
```

1.9.2.1 Biểu thức khẳng định

Khẳng định trong các câu lệnh bất biến là các biểu thức được tạo nên từ các đặc tính ngữ nghĩa của kiểu dữ liệu được định nghĩa. Biểu thức khẳng định phải có giá trị Boolean (*đúng* hoặc *sai*.)[9] Không có kiểu dữ liệu nguyên thủy nào, hay thao tác nào, có trước định nghĩa của một kiểu dữ liệu bất kỳ. Đặc điểm duy nhất có từ trước của ngôn ngữ biểu thức khẳng định là:[10]

- chuỗi ký tự biểu diễn lời nói trong ngôn ngữ định nghĩa kiểu dữ liệu;
- khái niệm về một khẳng định thành công (*đúng*) hoặc thất bại (*sai*);
- câu lệnh bất biến: **invariant**(...) **where** ... {...};
- hình thức biểu thức từ lượng hóa chung **forall** (...) **where** ... {...}; đồng nghĩa với câu lệnh bất biến;
- hình thức biểu thức từ lượng hóa tồn tại **exist** (...) **where** ... {...};

- phép hội ẩn (toán tử logic AND) giữa các *khẳng định* được phân cách với nhau bằng dấu chấm phẩy: *khẳng định*$_1$; *khẳng định*$_2$; ... ; *khẳng định*$_n$;
- các biến số và các khai báo trong danh sách tham số bất biến;
- tham chiếu đặc tính sử dụng dấu chấm: *x.property*;
- chuyển đổi kiểu ẩn và hiện: **(***T***)***x*;
- các dấu ngoặc đơn để ghi đè ưu tiên của chuyển đổi và toán tử phân giải đặc tính: (*T*)*x.property* so với ((*T*)*x*).*property*.

1.9.2.2 Biểu thức từ lượng hóa lồng nhau

Trong các biểu thức khẳng định, có thể hình thành các câu lệnh từ lượng hóa lồng nhau tương tự như các câu lệnh bất biến. Trong thực tế, từ lượng hóa chung được tạo nên nhờ sử dụng từ khóa **forall (đối với tất cả)** cũng tương tự như câu lệnh bất biến. Có thể sử dụng từ lượng hóa chung trong một biểu thức lồng nhau khi mức độ phức tạp của vấn đề đòi hỏi điều này, như trong ví dụ dưới đây:

```
invariant(SET<T> x, y)
    where x.nonNull {
  x.subset(y).equal(
    forall(T element) where x.contains(element) {
      y.contains(element);
      });
};
```

Từ lượng hóa tồn tại có nghĩa như trong logic mệnh đề chung. Ví dụ, bất biến sau đây có nghĩa: "TẬP HỢP các giá trị *x* và *y* giao nhau khi và chỉ khi có tồn tại một phần tử *e* được chứa trong cả tập hợp *x* và *y*."

```
invariant(SET x, y)
    where x.nonNull {
  x.intersects(y).equal(
    exists(T e) {
      x.contains(e);
      y.contains(e);
      });
};
```

Từ lượng hóa tồn tại có thể có một mệnh đề-where; tuy nhiên, không có sự khác biệt nào trong việc liệu một khẳng định được đưa ra như một mệnh đề-where hay trong nhóm từ lượng hóa tồn tại. Ngược lại, đối với các từ lượng hóa chung, mệnh đề-where làm suy yếu khẳng định vì nhóm này giờ chỉ áp dụng cho các giá trị đáp ứng được các tiêu chuẩn trong mệnh đề-where.

1.9.3 Chuyển đổi kiểu

Đặc tả này định nghĩa một số chuyển đổi được phép nhất định giữa các kiểu dữ liệu. Ví dụ, có một cặp chuyển đổi giữa Chuỗi ký tự (ST) và Mã hóa dữ liệu (ED). Điều này có nghĩa là nếu một người trông đợi giá trị ED nhưng thực tế lại có giá trị ST, người đó có thể chuyển giá trị ST thành ED.[11]

Có ba loại chuyển đổi kiểu được định nghĩa: tăng, giảm, và dạng chữ chuỗi ký tự. Chuyển đổi kiểu có thể là ẩn hoặc hiện. Chuyển đổi kiểu dạng ẩn xảy ra khi một kiểu nào đó được trông đợi (ví dụ như một tham số cho một câu lệnh) nhưng thực tế một kiểu khác lại được cung cấp. Nếu kiểu dữ liệu được cung cấp có một chuyển đổi sang kiểu dữ liệu được trông đợi, chuyển đổi nên được thực hiện ẩn.

LƯU Ý: Một Đặc tả Công nghệ Triển khai sẽ phải chỉ rõ chuyển đổi kiểu dạng ẩn được hỗ trợ như thế nào. Một số công nghệ hỗ trợ trực tiếp trong khi một số khác lại không; dù trong bất kỳ trường hợp nào, có thể thiết lập các quy tắc xử lý để chỉ rõ các chuyển đổi này được thực hiện như thế nào.

Có thể xác định một chuyển đổi tuyệt đối trong một biểu thức khẳng định bằng cách sử dụng tên kiểu được-chuyển-đổi-thành trong ngoặc đơn trước giá trị được chuyển đổi. Ví dụ, dưới đây là một chuyển đổi tuyệt đối trong mệnh đề-where của một câu lệnh bất biến.

```
invariant(ED x)
  where ((ST)x).nonNull { ... };
```

Chuyển đổi kiểu có ưu tiên thấp hơn so với chu kỳ phân giải đặc tính. Vì vậy "(T)a.b" chuyển đổi giá trị của đặc tính b của biến a thành kiểu dữ liệu T trong khi "((T)a).b " chuyển đổi giá trị của biến a thành T và sau đó tham chiếu đặc tính b của giá trị được chuyển đổi đó.

Chuyển đổi kiểu dạng ẩn trong biểu thức khẳng định được thực hiện khi có thể. Nếu tham số chính thức của một đặc tính được khai báo là thuộc kiểu dữ liệu T; nhưng biểu thức được sử dụng như một tham số thực sự lại thuộc kiểu U; và nếu U không mở rộng T; và nếu U xác định một chuyển đổi thành T, chuyển đổi từ T sang U có hiệu lực.

1.9.3.1 Chuyển đổi giảm

Chuyển đổi giảm là chuyển đổi có sự mất mát thông tin. Nhìn chung, điều này có nghĩa là một kiểu phức tạp hơn được chuyển đổi thành một kiểu đơn giản.

Một ví dụ cho chuyển đổi giảm là chuyển đổi từ Khoảng (IVL) thành một Đại lượng (QTY) đơn giản, ví dụ trung tâm khoảng. Trong ngôn ngữ định nghĩa kiểu dữ liệu, một chuyển đổi giảm được khai báo nhờ sử dụng từ khóa **demotion (giảm)** và tên kiểu dữ liệu giảm về:

```
type Interval alias IVL {
```

```
  ...
  demotion  QTY;
  ...
};
```

Đặc tả chuyển đổi giảm phải chỉ ra thông tin nào bị mất và hậu quả chính của việc mất thông tin đó là gì.

1.9.3.2 Chuyển đổi tăng

Chuyển đổi tăng là chuyển đổi trong đó có thông tin mới được tạo ra. Nhìn chung, điều này có nghĩa là một kiểu đơn giản hơn được chuyển đổi thành một kiểu phức tạp hơn.

Ví dụ, chúng ta cho phép bất kỳ Đại lượng (QTY) nào được chuyển đổi thành một Khoảng (IVL). Tuy nhiên, IVL có nhiều đặc tính ngữ nghĩa hơn so với QTY, biên trên và biên dưới. Vì vậy, chuyển đổi từ QTY thành IVL là một chuyển đổi tăng. Các đặc tính bổ sung của QTY không có trong IVL phải nhận giá trị mới, giá trị mặc định, hoặc giá trị tính toán. Đặc tả chuyển đổi tăng phải chỉ ra những giá trị này là gì hoặc chúng có thể được tạo ra như thế nào.

Một chuyển đổi tăng từ kiểu QTY sang kiểu IVL được định nghĩa là một đặc tính ngữ nghĩa của kiểu dữ liệu QTY nhờ sử dụng từ khóa **promotion (tăng)** và tên kiểu dữ liệu tăng lên thành:

```
type Quantity alias QTY {
  ...
  promotion  IVL;
  ...
};
```

Thông thường, một chuyển đổi tăng được định nghĩa là từ một kiểu đơn giản thành một kiểu phức tạp hơn. Cũng thường như vậy, trong tài liệu này kiểu đơn giản được khai báo sớm hơn so với kiểu phức tạp hơn. Do đó, khai báo tất cả chuyển đổi tăng lên kiểu phức tạp trong kiểu đơn giản sẽ bao gồm các tham chiếu về phía trước và sẽ gây khó hiểu cho người đọc. Vì vậy, một cú pháp thay thế cho phép định nghĩa chuyển đổi tăng trong kiểu phức tạp hơn. Việc này được chỉ ra bằng cách nêu tên của kiểu mà từ đó tiến hành chuyển đổi tăng trong một danh sách tham số đặt sau kiểu mà chuyển đổi tăng biến thành.

```
type Interval alias IVL {
  ...
  promotion  IVL (QTY x);
  ...
};
```

1.9.4 Dạng chữ

Dạng chữ là biểu diễn chuỗi ký tự của một giá trị dữ liệu. Dạng chữ được định nghĩa cho nhiều kiểu. Dạng chữ là một chuyển đổi kiểu thành một Chuỗi ký tự (ST) với một cú pháp được định nghĩa đặc biệt.

Tuy nhiên, không phải mọi chuyển đổi thành một ST đều là chuyển đổi dạng chữ. Một chuyển đổi dạng chữ cho một kiểu dữ liệu nên có khả năng biểu diễn toàn bộ tập hợp giá trị của kiểu dữ liệu trong khi bất kỳ chuyển đổi nào khác thành ST có thể chỉ liên hệ một tập con nhỏ hơn của kiểu dữ liệu được chuyển đổi.

Mục đích của việc có dạng chữ là để một người có thể viết ra các giá trị dưới hình thức ngắn gọn mà con người có thể đọc được. Ví dụ, dạng chữ cho các kiểu số nguyên (INT) và số thực (REAL) là các chuỗi ký hiệu, số, có thể cả dấu thập phân, v.v. Các kiểu khoảng quan trọng hơn (IVL<REAL>, IVL<PQ>, IVL<TS>) lại có biểu diễn dạng chữ cho phép con người sử dụng, ví dụ, "<5" có nghĩa "nhỏ hơn 5", thì dễ đọc hơn rất nhiều so với một hình thức được cấu trúc hoàn toàn của khoảng. Đối với một số kiểu dữ liệu cao cấp hơn như khoảng, đặc tả định thời gian tổng quát, và phân bổ xác suất theo tham số, chúng tôi cho rằng dạng chữ có thể là hình thức duy nhất để biểu diễn các giá trị này cho tới khi người sử dụng đã quen với những khái niệm cơ bản.

Mỗi chuyển đổi dạng chữ có cú pháp (ngữ pháp) của riêng nó, thường được điều chỉnh phù hợp với những gì mọi người cảm thấy trực quan. Cú pháp này vì thế có thể không hoàn toàn dễ hiểu từ quan điểm của một máy tính.[12]

LƯU Ý: Đặc tả Công nghệ Triển khai (ITS) dựa trên chuỗi ký tự của các kiểu dữ liệu trừu tượng này có thể có hoặc không chọn dạng chữ được định nghĩa ở đây để biểu diễn các kiểu dữ liệu này. Chúng tôi hy vọng ITS XML sẽ sử dụng không phải tất cả mà là một vài dạng chữ được định nghĩa ở đây.

1.9.4.1 Khai báo

Trong ngôn ngữ định nghĩa kiểu dữ liệu chúng ta khai báo dạng chữ như một đặc tính của kiểu dữ liệu bằng cách sử dụng từ khóa **literal (dạng chữ)** tiếp đến là tên kiểu dữ liệu ST, do dạng chữ là một chuyển đổi thành kiểu dữ liệu ST.

```
type IntegerNumber alias INT {
  ...
  literal  ST;
  ...
};
```

1.9.4.2 Định nghĩa

Định nghĩa thực sự của dạng chữ được thực hiện bên ngoài nhóm khai báo kiểu dữ liệu sử dụng ngữ pháp thuộc tính. Ngữ pháp thuộc tính là ngữ pháp chỉ ra cả cú

pháp và ngữ nghĩa của cấu trúc ngôn ngữ. Cú pháp được định nghĩa về cơ bản trong Dạng Backus-Naur (BNF).[13]

Ví dụ, cân nhắc định nghĩa đơn giản sau đây về một kiểu dữ liệu cho số đếm (các số nguyên dương). Định nghĩa kiểu này chỉ phụ thuộc vào kiểu dữ liệu Boolean (BL) và có dạng chữ chuỗi ký tự được khai báo:

```
type CardinalNumber alias CARD {
  BL      isZero;
  BL      equal(ANY x);
  CARD    successor;
  CARD    plus(CARD x);
  CARD    timesTen;
  literal  ST;
};
```

Cú pháp và ngữ nghĩa dạng chữ ban đầu được thể hiện đầy đủ và sau đó được mô tả chi tiết.

```
CARD.literal ST {
  CARD : CARD digit  { $.equal($1.timesTen.plus($2); }
       | digit       { $.equal($1); };

  CARD digit : "0"   { $.isZero; }
             | "1"   { $.equal(0.successor); }
             | "2"   { $.equal(1.successor); }
  ...
             | "8"   { $.equal(7.successor); }
             | "9"   { $.equal(8.successor); }
};
```

Mọi quy tắc cú pháp đều chứa tên của ký hiệu, dấu hai chấm và định nghĩa (được gọi là *sản phẩm*) của ký hiệu. Một sản phẩm là một chuỗi ký hiệu. Các ký hiệu khác này cũng được định nghĩa trong ngữ pháp, hay chúng là các ký hiệu kết thúc. Ký hiệu kết thúc là các chuỗi ký tự được viết trong các dấu ngoặc kép hoặc các mẫu chuỗi (được gọi là *biểu thức chính quy*). Vì vậy dạng thức:

```
CARD : CARD digit
     | digit;
```

có nghĩa là bất kỳ ký hiệu số đếm nào đều là ký hiệu số đếm với một chữ số đứng sau hoặc chỉ là một chữ số. Vạch dọc biểu thị phép tuyển (toán tử logic OR). Một quy tắc cú pháp kết thúc bằng một dấu chấm phẩy.

Mọi ký hiệu đều có chính xác một giá trị của một kiểu dữ liệu được định nghĩa. Kiểu dữ liệu của giá trị của ký hiệu được khai báo khi ký hiệu được khai báo:

```
CARD digit : "0"
           | "1"
           | "2"
           | ...
           | "8"
           | "9";
```

có nghĩa là ký hiệu *digits* (chữ số) có một giá trị của kiểu CARD. Ký hiệu-bắt đầu bản thân nó là kiểu dữ liệu và không cần một tên riêng biệt.

Ngữ nghĩa của biểu thức dạng chữ được xác định trong các quy tắc ngữ nghĩa nằm trong dấu ngoặc cong cho mỗi sản phẩm được định nghĩa của một ký hiệu:

ký hiệu **:** $sản\ phẩm_1$ **{** $quy\ tắc_1$ **}** | $sản\ phẩm_2$ **{** $quy\ tắc_2$ **}** | ... | $sản\ phẩm_n$ **{** $quy\ tắc_n$ **};**

Một quy tắc ngữ nghĩa đơn giản là một danh sách được phân cách bằng dấu chấm phẩy các biểu thức khẳng định Boolean của cùng một kiểu như được dùng trong các câu lệnh bất biến. Tuy nhiên, có những biến số đặc biệt được định nghĩa trong quy tắc ngữ nghĩa mà đều bắt đầu bằng ký tự dollar (ví dụ, $, $1, $2, $3, ...) Ký tự $ đơn giản đại diện cho giá trị của ký hiệu hiện được định nghĩa; trong khi $1, $2, $3, v.v. đại diện cho giá trị của các phần của sản phẩm liên quan của quy tắc ngữ nghĩa. Ví dụ, trong

```
CARD : CARD digit  { $.equal($1.timesTen.plus($2); }
     | digit       { $.equal($1); };
```

sản phẩm đầu tiên "CARD digit" có quy tắc ngữ nghĩa cho biết: giá trị $ của ký hiệu được định nghĩa bằng với giá trị $1 của ký hiệu đầu tiên CARD nhân mười cộng giá trị $2 của chữ số ký hiệu thứ hai.[14]

Một ký hiệu kết thúc có thể được xác định là một mẫu chuỗi, được gọi là *biểu thức chính quy*. Cú pháp biểu thức chính quy được sử dụng ở đây là cú pháp kinh điển do Aho phát minh và được sử dụng trong AWK, LEX, GREP, và PERL. Các biểu thức chính quy xuất hiện giữa hai dấu gạch chéo /.../. Trong một mẫu biểu thức chính quy, mọi ký tự ngoại trừ **[] ^ $. / : ( ) \ | ? * + {}** so khớp với chính nó. Các ký tự khác thực sự được sử dụng trong đặc tả này được định nghĩa ở Bảng 2.

Bảng 2. Các ký tự đặc biệt cho biểu thức chính quy

Mẫu	**Định nghĩa**

Mẫu	Định nghĩa
[...]	Xác định một lớp ký tự. Ví dụ, /[A-Za-z]/ so khớp các ký tự của bảng chữ cái viết hoa và viết thường.
[^ ...]	Quy định loại trừ một lớp ký tự. Ví dụ, /[^BCD]/ so khớp bất kỳ ký tự nào ngoại trừ B, C, và D.
...?	Mẫu trước đó là tùy chọn. Ví dụ, /ab?c/ so khớp "ac" và "abc".
...*	Mẫu trước đó có thể không xảy ra hoặc xảy ra nhiều lần. Ví dụ, /ab*c/ so khớp "ac", "abc", "abbc", "abbbc", v.v.
...+	Mẫu trước đó có thể xảy ra một hoặc nhiều lần. Ví dụ, /ab+c/ so khớp "abc", "abbc", "abbbc", nhưng không so khớp "ac".
... {n,m}	Mẫu trước đó có thể xảy ra từ *n* đến *m* lần, trong đó *n* và *m* là các số đếm 0 (*n (m*. Ví dụ, /ab{2,4}c/ so khớp "abbc", "abbbc", và "abbbbc".
... | ...	Mẫu ở hai phía của vạch dọc có thể so khớp. Ví dụ, /ab|cd/ so khớp "abd" và "acd" nhưng không so khớp "abcd".
(...)	Mẫu trong ngoặc đơn được sử dụng như một mẫu cho các toán tử phía trên. Ví dụ, /a(bc)*/ so khớp "a", "abc", "abcbc", "abcbcbc", v.v.
... : ...	Mẫu bên trái so khớp nếu có mẫu bên phải đứng sau, nhưng so khớp không dùng mẫu bên phải. Ví dụ, /ab:c/ so khớp "abc" nhưng không so khớp "ab", tuy nhiên, giá trị của ký hiệu được so khớp là "ab" còn "c" được để lại cho ký hiệu tiếp sau. Dấu hai chấm là hình thức phát sinh của dấu gạch chéo thông thường / nhưng dấu gạch chéo cũng thường được dùng để bao quanh toàn bộ mẫu và có thể xuất hiện như một ký tự để so khớp - ba nghĩa trong một là quá nhiều.
... \ ...	So khớp ký tự tiếp sau về mặt chữ, nghĩa là thoát khỏi bất kỳ ý nghĩa đặc biệt nào của ký tự đó. Ví dụ, /a\+b/ so khớp "a+b".
... \/ ...	So khớp dấu gạch chéo với tư cách một ký tự. Ví dụ, /a\/bc/ so khớp "a/bc".

1.9.5 Các kiểu dữ liệu chung

Các kiểu dữ liệu chung là các định nghĩa kiểu chưa hoàn thiện. Sự chưa hoàn thiện này được biểu hiện bằng một hay nhiều *tham số* cho định nghĩa kiểu. Thông thường các tham số đại diện cho các kiểu khác. Với việc sử dụng tham số, một kiểu chung có thể khai báo các đặc tính ngữ nghĩa của các kiểu dữ liệu khác chưa được xác định đầy đủ. Ví dụ, kiểu dữ liệu chung Khoảng được khai báo với một tham số *T*, tham số này có thể đại diện cho bất kỳ kiểu dữ liệu Đại lượng (QTY) nào. Các thành phần *low* (thấp) và *high* (cao) được khai báo là thuộc kiểu *T*.

```
template<QTY T>
type Interval<T> alias IVL<T> {
  T  low;
  T  high;
};
```

Thuyết minh một kiểu chung có nghĩa là hoàn thiện định nghĩa của nó. Ví dụ, để thuyết minh cho một Khoảng, phải chỉ ra khoảng đó nên thuộc *kiểu dữ liệu cơ bản* nào. Có thể làm điều này bằng cách *ràng buộc* tham số *T*. Để thuyết minh một Khoảng các Số nguyên, cần ràng buộc tham số *T* với kiểu Số nguyên. Vì vậy, kiểu dữ liệu chưa hoàn thiện Khoảng được hoàn thiện thành kiểu dữ liệu *Khoảng số nguyên*.

Ví dụ, định nghĩa kiểu dưới đây cho MyType khai báo một đặc tính có tên "multiplicity" (số bội) là một khoảng của kiểu dữ liệu số đếm được sử dụng trong các ví dụ trên.

```
type MyType alias MT {
  IVL<CARD>  multiplicity;
};
```

1.9.5.1 Bộ sưu tập chung

Các kiểu dữ liệu chung cho các bộ sưu tập đang được sử dụng trong suốt đặc tả này. Trong số đó quan trọng nhất là:

Set-Tập hợp (SET<T>) Một tập hợp chứa các phần tử không theo một trật tự cụ thể nào và không có các phần tử trùng lặp.

Sequence-Chuỗi (LIST<T>) Một chuỗi là một bộ sưu tập các giá trị có một trật tự tùy ý nhưng cụ thể. Một chuỗi có một đầu và một đuôi, trong đó đầu là một phần tử và đuôi là chuỗi không có đầu.

Interval-Khoảng (IVL<T>) Một khoảng là một tập con liên tục của một kiểu có thứ tự.

Các kiểu chung này và kiểu chung khác được định nghĩa đầy đủ trong Các kiểu dữ liệu chung (§ 1.9.5). Các kiểu dữ liệu chung này và các đặc tính của chúng đã sớm được sử dụng trong đặc tả này. Để có thể hiểu đúng nhất đặc tả này, kiến thức về tập

hợp, chuỗi và khoảng là rất quan trọng và người đọc được khuyến nghị tham khảo phần Các kiểu dữ liệu chung (§ 1.9.5). khi gặp một kiểu chung được dùng để định nghĩa một kiểu khác.

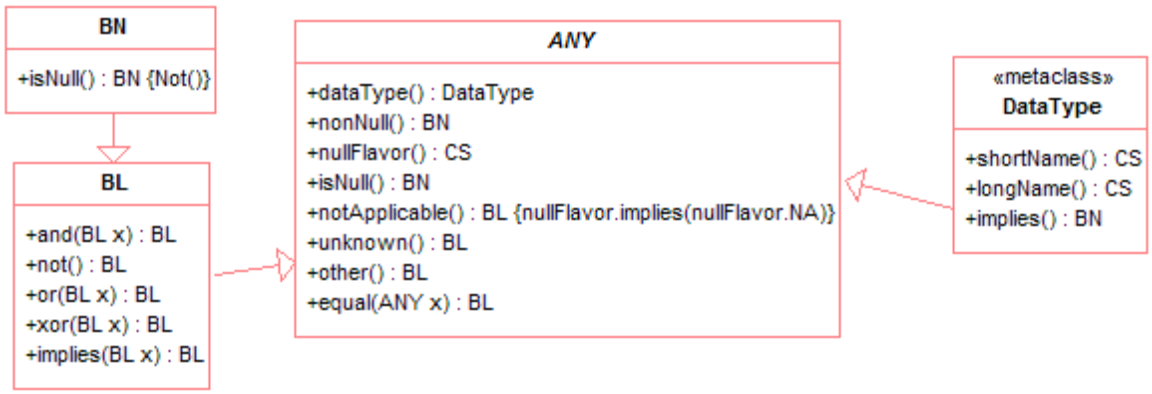
1.9.5.2 Phần mở rộng kiểu dữ liệu chung

Phần mở rộng kiểu dữ liệu chung là các kiểu dữ liệu chung với một kiểu tham số và là chuyên biệt hóa của kiểu dữ liệu chung. Trong ngôn ngữ định nghĩa kiểu dữ liệu chính thức, chuyên biệt hóa kiểu dữ liệu chung tuân theo mẫu sau:

```
template<ANY T> type GenericTypeExtensionName specializes T {
  ...
};
```

Các phần mở rộng kiểu dữ liệu chung thừa kế các đặc tính của kiểu dữ liệu cơ bản của chúng và bổ sung thêm một đặc điểm cụ thể nào đó. Phần mở rộng kiểu dữ liệu chung là một chuyên biệt hóa của kiểu dữ liệu cơ bản, vì vậy có thể sử dụng một giá trị của kiểu dữ liệu mở rộng thay cho kiểu dữ liệu cơ bản của nó.[15]

LƯU Ý: Giá trị của các kiểu dữ liệu được mở rộng có thể thay thế cho kiểu dữ liệu cơ bản của nó. Tuy nhiên, ITS có thể đưa ra một số hạn chế như có thể chứa phần mở rộng nào. Đặc biệt, không cần định nghĩa các phần mở rộng cho những thành phần có mang giá trị của các đặc tính giá trị dữ liệu. Vì vậy, trong khi bất kỳ giá trị dữ liệu nào đều có thể được chú giải bên ngoài đặc tả kiểu dữ liệu, ITS có thể không cung cấp cách thức chú giải giá trị của đặc tính giá trị dữ liệu.



Các kiểu dữ liệu cơ bản

1.10 Hợp chuẩn

Nếu một ứng dụng nhận được hay phân tích một thực thể không hợp lệ đối với đặc tả này, ứng dụng tiếp nhận được phép từ chối thực thể theo bất kỳ cách nào mà nó thấy phù hợp nhưng nó không bắt buộc phải làm như vậy. Lưu ý rằng một vài tiêu chuẩn hay sản phẩm HL7 khác như tuyên bố hợp chuẩn có thể đưa ra thêm một số hạn chế về hành vi trong các trường hợp này.

1.11 DataValue (ANY) – Giá trị dữ liệu

Định nghĩa: Xác định các đặc tính cơ bản của mọi giá trị dữ liệu. Đây là một kiểu trừu tượng, có nghĩa là không có giá trị nào có thể chỉ là giá trị dữ liệu mà không thuộc về bất kỳ kiểu dữ liệu cụ thể nào. Mọi kiểu dữ liệu cụ thể đều là một chuyên biệt hóa của kiểu Giá trị dữ liệu trừu tượng chung này.

```
abstract type DataValue alias ANY {
  TYPE  dataType;
  BN    nonNull;
  CS    nullFlavor;
  BN    isNull;
  BL    notApplicable;
  BL    unknown;
  BL    other;
  BL    equal(ANY x);
};
```

1.11.1 Kiểu dữ liệu: TYPE

Định nghĩa: Biểu diễn thực tế là mỗi giá trị dữ liệu đều ngầm mang thông tin về kiểu dữ liệu của riêng nó. Vì vậy, với một giá trị dữ liệu cho trước có thể hỏi về kiểu dữ liệu của nó.

```
invariant(ANY x) {
  x.dataType.nonNull;
};
```

1.11.2 Giá trị thích hợp: BN

Định nghĩa: Cho biết rằng một giá trị không phải là giá trị ngoại lệ của kiểu dữ liệu.

```
invariant(ANY x) {
  x.isNull.equal(x.nonNull.not);
};
```

Khi một đặc tính, thuộc tính RIM, hay trường bản tin được gọi là *bắt buộc*, điều này có nghĩa là bất kỳ giá trị không-NULL nào của kiểu mà đặc tính thuộc về đều có một giá trị non-NULL (không rỗng) cho đặc tính đó, hay nói cách khác, một trường có thể không NULL (rỗng), miễn là cái chứa nó (đối tượng, đoạn dữ liệu, v.v.) có một giá trị non-NULL.

1.11.3 Giá trị ngoại lệ: BN

Định nghĩa: Chỉ ra rằng một giá trị là giá trị ngoại lệ, hoặc NULL-value (giá trị rỗng). Một giá trị null có nghĩa là thông tin không tồn tại, không có sẵn hoặc không thể xuất hiện được trong tập hợp giá trị thông thường của kiểu dữ liệu.

Mọi phần tử dữ liệu đều hoặc có một giá trị hợp lệ hoặc được coi là NULL. Khi (và chỉ khi) nó là NULL, *isNull (là rỗng)* cung cấp thêm chi tiết về việc theo cách nào hay vì sao không có giá trị hợp lệ nào được cung cấp.

```
invariant(ANY x) {
  x.isNull.equal(x.nullFlavor.implies(NI));
};
```

1.11.4 Chi tiết giá trị ngoại lệ: CS

Định nghĩa: Nếu một giá trị là giá trị ngoại lệ (NULL-value), chi tiết này chỉ ra theo cách nào hay vì sao không có thông tin hợp lệ.

```
invariant(ANY x) {
  x.nonNull.equal(x.nullFlavor.isNull);
};
```

Bảng 3. Miền NullFlavor (trạng thái null)

mã	tên gọi	định nghĩa
NI	NoInformation (Không có thông tin)	Không có thông tin nào có thể được suy ra từ giá trị ngoại lệ này. Đây là giá trị ngoại lệ chung nhất. Nó cũng là giá trị ngoại lệ mặc định.
OTH	other (khác)	Giá trị thực không phải là một phần tử trong miền giá trị của một biến. (ví dụ, hệ thống mã được yêu cầu không cung cấp khái niệm).
NINF	negative infinity (âm vô cùng)	Âm vô cùng của các số.
PINF	positive infinity (dương vô cùng)	Dương vô cùng của các số.
UNK	Unknown	Một giá trị hợp lệ có thể áp dụng được, nhưng không biết.

mã	tên gọi	định nghĩa
	(Không biết)	
ASKU	asked but unknown (được hỏi nhưng không biết)	Thông tin được tìm kiếm nhưng không tìm thấy (ví dụ, bệnh nhân được truy vấn đến nhưng không biết)
NAV	temporarily unavailable (tạm thời không có)	Thông tin không sẵn có vào thời điểm này nhưng được kỳ vọng sau này sẽ có.
NASK	not asked (không được hỏi)	Thông tin này chưa được tìm kiếm (ví dụ, bệnh nhân không được truy vấn đến)
TRC	trace (dấu vết)	Nội dung lớn hơn 0, nhưng quá nhỏ để có thể định lượng.
MSK	Masked (Bị che dấu)	Thông tin về hạng mục này là sẵn có nhưng phía gửi chưa cung cấp vì lí do an ninh, bảo mật hay các lí do khác. Có thể có một cơ chế thay thế để đạt được quyền truy cập vào thông tin này. Lưu ý: việc sử dụng trạng thái null này cung cấp thông tin có thể dẫn tới việc vi phạm về bảo mật, mặc dù không có dữ liệu chi tiết nào được cung cấp. Mục đích chính của nó là để phục vụ những trường hợp cần báo cho phía tiếp nhận rằng thông tin có tồn tại mà không cung cấp bất kỳ chi tiết nào.
NA	not applicable (không áp dụng được)	Không có giá trị hợp lệ nào áp dụng được trong bối cảnh này (ví dụ, kỳ kinh nguyệt cuối cùng đối với nam giới).
NP	not present (không xuất hiện)	Giá trị không xuất hiện trong một bản tin. Loại này chỉ được định nghĩa trong các bản tin, không bao giờ được định nghĩa trong dữ liệu ứng dụng! Tất cả các

mã	tên gọi	định nghĩa
		giá trị không xuất hiện trong bản tin phải được thay thế bởi một mặc định có thể ứng dụng, hoặc không-có-thông-tin (NI) như là mặc định của tất cả các mặc định.

Các trạng thái null là một phần mở rộng miền chung của tất cả các kiểu dữ liệu thông thường. Lưu ý sự khác biệt giữa miền giá trị của bất kỳ kiểu dữ liệu nào với miền từ vựng của các kiểu dữ liệu được mã hóa. Một miền từ vựng là một miền giá trị cho các giá trị được mã hóa, nhưng không phải tất cả miền giá trị đều là miền từ vựng.

Trạng thái null "khác" được sử dụng bất kể khi nào giá trị thực không nằm trong miền giá trị được yêu cầu, ví dụ, có thể là khi giá trị vượt quá một vài ràng buộc được xác định là quá hạn chế (ví dụ, tuổi dưới 100).

LƯU Ý: Các trạng thái NULL ứng dụng được cho bất kỳ đặc tính nào của một giá trị dữ liệu hoặc thuộc tính đối tượng cấp cao hơn. Khi sự khác biệt giữa các trạng thái null không đáng kể, không yêu cầu ITS phải biểu diễn chúng. Nếu không có gì khác được lưu ý trong đặc tả này, ITS không cần biểu diễn các trạng thái NULL chung cho các đặc tính giá trị dữ liệu.

Một vài trong số các trạng thái null này được gắn với các đặc tính được đặt tên mà có thể được dùng như các vị từ đơn giản cho tất cả các giá trị dữ liệu. Điều này là để đơn giản hóa việc xây dựng tính bất biến trong phần còn lại của đặc tả này.

Cần lưu ý rằng sự khác biệt giữa các đặc tính ngữ nghĩa và các "thành phần" biểu diễn của các giá trị dữ liệu. ITS chỉ được biểu diễn các thành phần cần thiết cho việc *suy ra* đặc tính ngữ nghĩa. Các vị từ trạng thái null nonNull (không rỗng), isNull (là rỗng), notApplicable (không áp dụng được), unknown (không biết hoặc không xác định), và other (khác) đều có thể được suy ra từ đặc tính *nullFlavor (trạng thái null)*.

1.11.5 Giá trị thích hợp không được áp dụng: BL

Định nghĩa: Một vị từ chỉ ra rằng giá trị ngoại lệ này là thuộc nullFlavor (trạng thái rỗng) không-áp-dụng-được (NA), nghĩa là một giá trị hợp lệ không có nghĩa trong bối cảnh đã cho.

```
invariant(ANY x) {
  x.notApplicable.equal(x.nullFlavor.implies(NA));
};
```

1.11.6 Không xác định: BL

Định nghĩa: Một vị từ chỉ ra rằng giá trị ngoại lệ này là thuộc nullFlavor (trạng thái null) không-biết (UNK).

```
invariant(ANY x) {
  x.unknown.equal(x.nullFlavor.implies(UNK));
};
```

1.11.7 Loại trừ miền giá trị: BL

Định nghĩa: Một vị từ chỉ ra rằng giá trị ngoại lệ này là thuộc nullFlavor (trạng thái null) khác (OTH), nghĩa là miền giá trị được yêu cầu không chứa giá trị phù hợp.

```
invariant(ANY x) {
  x.other.equal(x.nullFlavor.implies(OTH));
};
```

1.11.8 Bình đẳng: BL

Định nghĩa: Bình đẳng là một quan hệ có tính phản xạ, đối xứng, và bắc cầu giữa hai giá trị dữ liệu bất kỳ. Chỉ các giá trị hợp lệ mới có thể bình đẳng, các giá trị null không bao giờ bình đẳng (ngay cả khi chúng có cùng trạng thái null).

```
invariant(ANY x, y, z)
    where x.nonNull.and(y.nonNull).and(z.nonNull) {
  x.equal(x);         /* reflexivity */
  x.equal(y).equal(y.equal(x));     /* symmetry */
  x.equal(y).and(y.equal(z)).implies(x.equal(z)); /* transitivity */
  x.equal(y).implies(x.dataType.equal(y.dataType);
};
```

Phương thức xác định bình đẳng phải được định nghĩa cho từng kiểu dữ liệu. Nếu không có quy định gì khác được nêu ra, hai giá trị dữ liệu được xem là bình đẳng nếu không thể phân biệt chúng, nghĩa là, nếu chúng không có gì khác biệt trong đặc tính ngữ nghĩa của chúng. Một kiểu dữ liệu có thể "ghi đè hoặc phủ quyết" định nghĩa chung về bình đẳng, bằng cách chỉ ra mối quan hệ bình đẳng của riêng nó. Có thể sử dụng việc phủ quyết mối quan hệ bình đẳng này để loại trừ các đặc tính ngữ nghĩa khỏi thử nghiệm bình đẳng. Nếu một kiểu dữ liệu loại trừ các đặc tính ngữ nghĩa khỏi định nghĩa của nó về bình đẳng, điều này ngụ ý rằng có một vài đặc tính nhất định (hay các khía cạnh của đặc tính), mà không phải một phần của thử nghiệm bình đẳng, không cần thiết đối với ý nghĩa của giá trị.

Ví dụ, đại lượng vật lý có hai đặc tính ngữ nghĩa (1) một số thực và (2) một đơn vị được mã hóa của phép đo. Tuy nhiên, thử nghiệm bình đẳng phải giải thích được thực tế là, ví dụ, 1 mét bằng 100 centimét; bình đẳng độc lập của hai đặc tính ngữ

nghĩa là một tiêu chí quá mạnh cho thử nghiệm bình đẳng. Vì vậy, đại lượng vật lý phải phủ quyết định nghĩa bình đẳng.

1.12 DataType - Kiểu dữ liệu (TYPE) chuyên biệt hóa ANY

Định nghĩa: Một siêu kiểu được khai báo để cho phép định nghĩa chính thức về kiểu dữ liệu của một giá trị. Bất kỳ kiểu dữ liệu nào được định nghĩa trong đặc tả này đều là giá trị của kiểu DataType (Kiểu dữ liệu).

```
private type DataType alias TYPE specializes DataValue {
  CS  shortName;
  CS  longName;
  BN  implies(TYPE that);
};
```

1.12.1 Tên ngắn: CS

Định nghĩa: Một CS chỉ rõ bí danh của kiểu dữ liệu.

```
invariant(DataType x)
    where x.nonNull {
  x.shortName.nonNull;
};
```

1.12.2 Tên dài: CS

Định nghĩa: Một CS chỉ rõ tên đầy đủ của kiểu dữ liệu.

1.12.3 Hàm ý: BN

Định nghĩa: Một kiểu dữ liệu ngụ ý một kiểu dữ liệu khác nếu nó có cùng kiểu hoặc là một chuyên biệt hóa của nó.

2. Các kiểu dữ liệu cơ bản

2.1 Boolean (BL) chuyên biệt hóa ANY

Định nghĩa: *BL* đại diện cho các giá trị logic hai giá trị. Một giá trị *BL* có thể hoặc *true (đúng)* hoặc *false (sai)*, hoặc, giống như bất kỳ giá trị nào khác, có thể là NULL.

```
type Boolean alias BL specializes ANY
  values(true, false) {
        BL  and(BL x);
        BL  not;
  literal  ST;
        BL  or(BL x);
        BL  xor(BL x);
```

```
        BL  implies(BL x);
};
```

Với bất kỳ giá trị dữ liệu nào có tiềm năng là NULL, hai giá trị logic được mở rộng một cách hiệu quả thành ba giá trị logic như được thể hiện trong các bảng chân lý sau đây:

Bảng 4. Bảng chân lý cho logic Boolean với các giá trị NULL

NOT			**AND**	**true**	**false**	**NULL**		**OR**	**true**	**false**	**NULL**
true	false		**true**	true	false	NULL		**true**	true	true	true
false	true		**false**	false	false	false		**false**	true	false	NULL
NULL	NULL		**NULL**	NULL	false	NULL		**NULL**	true	NULL	NULL

Khi một phép toán boolean được thực hiện trên hai kiểu dữ liệu với các trạng thái null khác nhau, trạng thái null của kết quả là tổ tiên chung đầu tiên của hai trạng thái null khác nhau, mặc dù các ứng dụng hợp chuẩn cũng có thể tạo ra một kết quả là tổ tiên chung bất kỳ.

2.1.1 Phủ định: BL

Định nghĩa: Phủ định của một *BL* biến *true (đúng)* thành *false (sai)* và *false (sai)* thành *true (đúng)* và là NULL đối với các giá trị NULL.

```
invariant(BL x) {
  true.not.equal(false);
  false.not.equal(true);
  x.isNull.equal(x.not.isNull);
};
```

2.1.2 Phép Hội: BL

Định nghĩa: Hội (AND) có tính kết hợp và giao hoán, với *true (đúng)* là phần tử trung lập. *False (sai)* AND với bất kỳ giá trị Boolean nào là *false (sai)*. Các quy tắc này có giá trị ngay cả khi một hoặc cả hai toán hạng là NULL. Nếu cả hai toán hạng cho toán tử AND là NULL, kết quả là NULL.

```
invariant(BL x) {
  x.and(true).equal(x);
  x.and(false).equal(false);
  x.isNull.implies(x.and(y).isNull);
};
```

2.1.3 Phép Tuyển: BL

Định nghĩa: Tuyển *x* OR *y* là *false (sai)* khi và chỉ khi *x* là *false (sai)* và *y* là *flase (sai).*

```
invariant(BL x, y) {
  x.or(y).equal(x.not.and(y.not).not);
};
```

2.1.4 Phép tuyển loại trừ: BL

Định nghĩa: OR-loại trừ giới hạn toán tử OR sao cho hai toán hạng không thể cùng là *true (đúng).*

```
invariant(BL x, y) {
  x.xor(y).equal(x.or(y).and(x.and(y).not));
};
```

2.1.5 Phép tất suy: BL

Định nghĩa: Một quy tắc của dạng thức IF *điều kiện* THEN *kết luận*. Về logic, tất suy được định nghĩa là tuyển của điều kiện bị phủ định và kết luận, nghĩa là khi điều kiện là *true (đúng)* kết luận phải là *true (đúng)* để toàn bộ tuyên bố là *true (đúng).* Tất suy logic rất quan trọng để đưa ra các câu lệnh bất biến.

```
invariant(BL condition, conclusion) {
  condition.implies(conclusion).equal(
    condition.not.or(conclusion));
};
```

Tất suy là không thể đảo ngược và không chỉ rõ cái gì là *true (đúng)* khi điều kiện là *false (sai)* (*ex falso quodlibet* tiếng Latin có nghĩa là "cái sai dẫn đến mọi thứ").

2.1.6 Dạng chữ

Dạng chữ của kiểu dữ liệu Boolean được quyết định bởi các giá trị có tên được chỉ ra trong mệnh đề giá trị, nghĩa là *true (đúng)* và *true (sai).*

2.2 BooleanNonNull (BN) chuyên biệt hóa BL

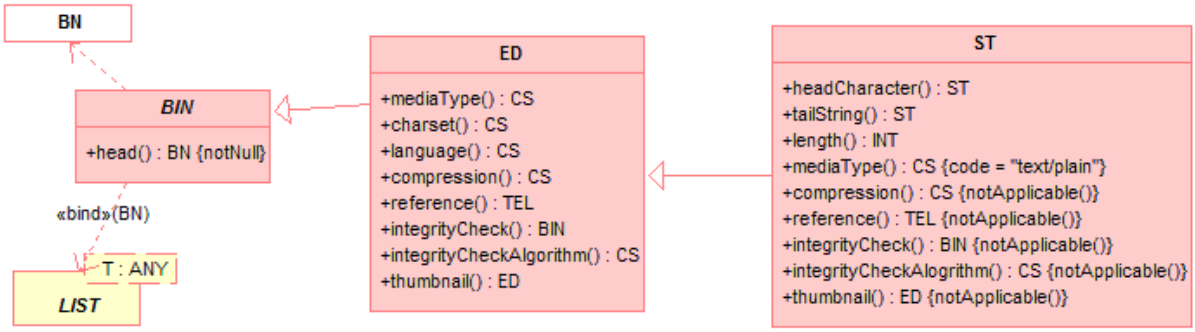
Định nghĩa: *BN* ràng buộc kiểu boolean sao cho giá trị không thể là NULL. Kiểu dữ liệu này được tạo ra để sử dụng trong đặc tả kiểu dữ liệu trong trường hợp việc sử dụng một giá trị null là không phù hợp.

```
private type BooleanNonNull alias BN specializes BL;
};
```

2.2.1 isNull: BN

```
invariant (BN x) {
```

```
    x.isNull.not
};
```



2.3 Dữ liệu nhị phân (BIN) chuyên biệt hóa LIST<BN>

Định nghĩa: *BIN* là một khối bit thô. *BIN* là kiểu dữ liệu được bảo vệ không nên được khai báo bên ngoài đặc tả kiểu dữ liệu.

Về mặt ngữ nghĩa, một bit giống với một giá trị non-null (không rỗng) BL. Vì vậy, tất cả các dữ liệu nhị phân là — về mặt ngữ nghĩa — một chuỗi các giá trị non-null BL

```
protected type BinaryData alias BIN specializes LIST<BN>;
```

LƯU Ý: Việc biểu diễn dữ liệu nhị phân tùy ý là trách nhiệm của ITS. Phương thức ITS làm được việc này phụ thuộc vào Công nghệ Triển khai cơ bản (dựa trên ký tự hay nhị phân) và vào dữ liệu được biểu diễn. Về mặt ngữ nghĩa, dữ liệu ký tự được biểu diễn như dữ liệu nhị phân, tuy nhiên, ITS dựa trên ký tự không nên chuyển đổi dữ liệu ký tự thành dữ liệu nhị phân tùy ý và sau đó biểu diễn dữ liệu nhị phân trong bảng mã ký tự. Cuối cùng, ngay cả công nghệ triển khai dựa trên ký tự cũng sẽ truyền dữ liệu nhị phân.

Một chuỗi rỗng không được xem là dữ liệu nhị phân nhưng được tính là một giá trị NULL. Nói cách khác, dữ liệu nhị phân non-NULL (không rỗng) chứa ít nhất một bit. Không bit nào trong một giá trị dữ liệu nhị phân non-NULL có thể là NULL.

```
invariant(BIN x)
    where x.nonNull {
  x.notEmpty;
  x.length.greaterThan(0);
};
```

2.4 Dữ liệu được đóng gói (ED) chuyên biệt hóa BIN

Định nghĩa: Dữ liệu có mục đích chủ yếu là để cho người đọc hiểu hoặc cho các xử lý máy tính khác bên ngoài phạm vi của HL7. Nó bao gồm ngôn ngữ viết có định dạng hoặc không có định dạng, dữ liệu đa phương tiện, hoặc các thông tin có cấu trúc do một tiêu chuẩn khác xác định (ví dụ, XML – chữ ký). Thay vì chứa bản thân dữ liệu, một *ED* cũng có thể chỉ chứa một đường tham chiếu (xem TEL.) Cần lưu ý

rằng ST là một chuyên biệt hóa của *ED* khi Kiểu phương tiện được cố định là văn bản/đơn thuần.

Bảng 5. Tóm tắt đặc tính của Dữ liệu được đóng gói

Tên	Kiểu	Mô tả
mediaType (Kiểu phương tiện)	CS	Xác định kiểu của dữ liệu được đóng gói và xác định một phương pháp để giải thích hoặc biểu diễn dữ liệu.
charset (tập hợp ký tự)	CS	Đối với các kiểu mã hóa dựa trên ký tự, đặc tính này xác định tập hợp ký tự và mã hóa ký tự được sử dụng. Tập hợp ký tự phải được nhận biết bởi một Đăng ký Tập hợp Ký tự của Tổ chức có thẩm quyền cấp phát số hiệu Internet (IANA) [http://www.iana.org/assignments/character-sets] phù hợp với RFC 2978 [http://www.ietf.org/rfc/rfc2978.txt].
Language (ngôn ngữ)	CS	Đối với thông tin dựa trên ký tự, đặc tính ngôn ngữ quy định cụ thể ngôn ngữ mà con người dùng trong văn bản.
Compression (nén)	CS	Cho biết liệu dữ liệu byte thô có được nén hay không, và thuật toán nén được sử dụng là gì.
Reference (tham chiếu)	TEL	Một địa chỉ viễn thông (TEL), chẳng hạn như địa chỉ URL đối với HTTP hoặc FTP, sẽ cung cấp chính xác cùng một dữ liệu nhị phân mà cũng có thể được cung cấp như dữ liệu nội tuyến.
integrityCheck (Kiểm tra tính toàn vẹn)	BIN	Kiểm tra tính toàn vẹn là một giá trị nhị phân ngắn biểu diễn một tổng kiểm tra đủ mạnh về mặt mã hóa được tính trên các dữ liệu nhị phân. Mục đích của đặc tính này, khi được thông tin cùng với một tham chiếu, là để sau này kiểm tra xem đường tham chiếu có dẫn đến cùng dữ liệu mà đường tham chiếu đã dẫn tới khi giá trị dữ liệu

Tên	Kiểu	Mô tả
		được đóng gói có chứa đường tham chiếu được tạo ra hay không.
integrityCheckAlgorithm (Thuật toán Kiểm tra tính toàn vẹn)	CS	Xác định thuật toán được dùng để tính giá trị integrityCheck (Kiểm tra tính toàn vẹn). Thuật toán của tổng kiểm tra đủ mạnh về mặt mã hóa Thuật toán Băm An toàn-1 (SHA-1) hiện là tiêu chuẩn trong ngành công nghiệp này. Nó mới chỉ thay thế thuật toán MD5 vài năm gần đây, khi người ta phát hiện một số lỗi về bảo mật của MD5. Hiện nay, thuật toán băm SHA-1 là lựa chọn mặc định cho thuật toán kiểm tra tính toàn vẹn. Lưu ý rằng SHA-256 cũng đang bắt đầu được sử dụng rộng rãi.
Thumbnail (biểu tượng thu nhỏ)	ED	Một dạng biểu diễn ngắn gọn của dữ liệu đầy đủ. Biểu tượng thu nhỏ yêu cầu ít tài nguyên hơn rất nhiều so với dữ liệu đầy đủ, trong khi vẫn duy trì được sự tương đồng đặc biệt với dữ liệu đầy đủ. Biểu tượng thu nhỏ thường được dùng với dữ liệu được đóng gói có tham chiếu. Nó cho phép người dùng chọn dữ liệu hiệu quả hơn trước khi thực sự tải xuống qua đường tham chiếu.

```
type EncapsulatedData alias ED specializes BIN {
  CS   mediaType;
  CS   charset;
  CS   language;
  CS   compression;
  TEL  reference;
  BIN  integrityCheck;
  CS   integrityCheckAlgorithm;
  ED   thumbnail;
  BL   equal(ANY x);
};
```

Dữ liệu được đóng gói có thể xuất hiện dưới hai hình thức, nội tuyến và qua tham chiếu. Dữ liệu nội tuyến được truyền đi hoặc di chuyển như một phần của giá trị dữ liệu được đóng gói, trong khi dữ liệu qua tham chiếu có thể nằm ở một vị trí (từ xa) khác. Dữ liệu là như nhau dù nó được đặt nội tuyến hay ở xa.

2.4.1 Kiểu dữ liệu phương tiện (media): CS

Định nghĩa: Xác định kiểu của dữ liệu được đóng gói và xác định một phương pháp để giải thích hoặc trình diễn dữ liệu.

Kiểu phương tiện truyền thông là đặc tính bắt buộc, nghĩa là mọi thực thể non-NULL của kiểu dữ liệu *ED* phải có một đặc tính *Kiểu phương tiện* non-NULL.

```
invariant(ED x)
    where x.nonNull {
  x.mediaType.nonNull;
};
```

Miền kiểu phương tiện do IANA định nghĩa được thiết lập nhờ tiêu chuẩn Internet RFC 2045 [http://www.ietf.org/rfc/rfc2045.txt] và 2046 [http://www.ietf.org/rfc/rfc2046.txt]. RFC 2046 định nghĩa kiểu phương tiện bao gồm hai phần:

1. Kiểu phương tiện cấp cao nhất, và
2. Kiểu phương tiện phụ

Tuy nhiên, đặc tả này coi toàn bộ kiểu phương tiện như một ký hiệu mã nguyên tử dưới hình thức do IANA định nghĩa, tức là kiểu cấp cao nhất rồi đến một dấu gạch chéo "/" rồi đến kiểu phương tiện phụ. Các kiểu phương tiện được định nghĩa gần đây được đăng ký trong cơ sở dữ liệu [http://www.iana.org/assignments/media-types/index.html] do IANA duy trì. Hiện có hơn 160 kiểu phương tiện MIME khác nhau được định nghĩa, và danh sách này đang tăng lên nhanh chóng. Nhìn chung, có thể sử dụng tất cả các kiểu dữ liệu do IANA định nghĩa.

Để gia tăng mức độ liên kết vận hành, đặc tả này ưu tiên một số kiểu phương tiện hơn một số kiểu khác. Điều này là để xác định một mẫu số chung lớn nhất trên đó sự liên kết vận hành không chỉ là có thể, mà còn đủ mạnh để hỗ trợ ngay cả các nhu cầu thông tin đa phương tiện tiên tiến nhất.

Bảng 6 dưới đây gán một trạng thái cho các kiểu phương tiện MIME nhất định, trong đó trạng thái có một trong các nghĩa sau:

- **bắt buộc**: Mọi ứng dụng HL7 phải hỗ trợ ít nhất các kiểu phương tiện bắt buộc nếu nó hỗ trợ một kiểu phương tiện cho trước. Tồn tại một kiểu phương tiện bắt buộc cho mỗi kiểu phương tiện. Một số kiểu phương tiện có thuộc

tính bắt buộc cho một mục đích cụ thể, và sau này sẽ được chỉ rõ là "bắt buộc cho..."

- **khuyến nghị**: Các kiểu phương tiện khác được khuyến nghị cho một mục đích cụ thể. Đối với một mục đích cụ thể bất kỳ, chỉ nên có rất ít các kiểu phương tiện được khuyến nghị bổ sung và phải nêu rất rõ lí do, điều kiện và các giả định của những khuyến nghị này.
- **không ý kiến**: Trạng thái này có nghĩa, HL7 không ngăn cấm cũng không hỗ trợ việc sử dụng kiểu phương tiện này. Tất cả các kiểu phương tiện không được đề cập trong Bảng 6 đều mặc định có trạng thái không ý kiến. Do có một kiểu phương tiện bắt buộc và một vài kiểu được khuyến nghị cho các trường hợp sử dụng phù hợp nhất trên thực tế, nên sử dụng các kiểu phương tiện thuộc trạng thái này hết sức thận trọng.
- **không tán thành**: Không nên sử dụng các kiểu phương tiện không được tán thành, vì các kiểu phương tiện này có lỗi, vì còn có các lựa chọn thay thế khác tốt hơn, hoặc bởi vì có những rủi ro nhất định. Những rủi ro đó có thể là rủi ro về an ninh, ví dụ, rủi ro là kiểu phương tiện đó có thể phát tán virus máy tính. Tuy nhiên, không phải tất cả các kiểu phương tiện có lỗi đều bị gắn mác không tán thành. Một kiểu phương tiện không được đề cập tới trong Bảng 6, và do đó có trạng thái không ý kiến, rất có thể có lỗi.

Bảng 6. Miền MediaType

Mã	Tên	Trạng thái	Định nghĩa
text/plain	Văn bản đơn thuần	bắt buộc	Dành cho văn bản đơn thuần bất kỳ. Đây là kiểu dữ liệu mặc định và tương đương với kiểu dữ liệu chuỗi ký tự (ST).
text/x-hl7-ft	Văn bản định dạng HL7	Khuyến nghị	Để đảm bảo việc tương thích, kiểu dữ liệu này biểu diễn kiểu dữ liệu HL7 v2.x FT. Kiểu dữ liệu này chỉ được khuyến nghị sử dụng để tương thích ngược với các hệ thống HL7 v2.x.
text/html	Văn bản định dạng HTML	Khuyến nghị	Dành cho văn bản được đánh dấu theo Ngôn ngữ Đánh dấu Siêu văn bản. Định dạng HTML đảm bảo việc đánh dấu hầu hết các tài

Mã	Tên	Trạng thái	Định nghĩa
			liệu bằng văn bản. HTML là nền tảng độc lập và được triển khai rộng rãi.
application/pdf	PDF	Khuyến nghị	Định dạng tài liệu di động (Portable Document Format) được khuyến nghị sử dụng cho các văn bản được hiển thị hoàn toàn và chỉ cho phép đọc. Định dạn PDF không phụ thuộc vào nền tảng, được triển khai rộng rãi, mở và có các công cụ tạo và trình diễn miễn phí.
text/xml	Văn bản định dạng XML	không ý kiến	Dùng cho các dữ liệu dựa trên ký tự có cấu trúc. Có nguy cơ là SGML/XML tổng hợp thì quá mạnh để có thể cho phép chia sẻ các tài liệu SGML/XML tổng hợp giữa các ứng dụng khác nhau.
text/rtf	Văn bản định danh RTF	không ý kiến	Định dạng văn bản giàu tính chất (Rich text format) được sử dụng rộng rãi để chia sẻ các tài liệu của bộ xử lý văn bản. Tuy nhiên, định dạng RTF có các vấn đề về tương thích, vì nó phụ thuộc rất nhiều vào bộ xử lý văn bản. Có thể hữu ích nếu chia sẻ các văn bản có thể được biên tập bởi bộ xử lý văn bản.
application/msword	MSWORD	không tán thành	Định dạng này dễ gặp phải các vấn đề về tương thích. Nếu cần chia sẻ các văn bản có thể được biên tập, nên dùng text/plain, text/html hoặc text/rtf thay vì kiểu phương tiện này.

Mã	Tên	Trạng thái	Định nghĩa
audio/basic	Âm thanh cơ bản	bắt buộc	Đây là một định dạng cho âm thanh kênh đơn, được mã hóa nhờ dùng 8 bit ISDN mu-law [PCM] với tốc độ mẫu là 8000 Hz. Định dạng này được tiêu chuẩn hóa bởi CCITT, Fascile III.4 – Khuyến nghị G.711. Điều biên mã xung (PCM) các tần số giọng nói. Geneva, 1972.
audio/mpeg	Lớp âm thanh 3 định dạng MPEG	bắt buộc	MPEG – 1 Lớp âm thanh - 3 là một thuật toán nén âm thanh và định dạng file được định nghĩa trong ISO 11172-3 và ISO 13818-3. MP3 có tần số lấy mẫu điều chỉnh được cho điện thoại nén mạnh sang âm thanh chất lượng CD.
audio/k32adpcm	Âm thanh K32ADPCM	không ý kiến	ADPCM cho phép nén dữ liệu âm thanh. Nó được định nghĩa trong đặc tả internet RFC 2421 [ftp://ftp.isi.edu/in-notes/rfc2421.txt]. Cơ sở triển khai của nó chưa rõ ràng.
image/png	Hình ảnh PNG	bắt buộc	Đồ họa mạng di động (Portable Network Graphics – PNG) [http://www.cdrom.com/pub/png] là một tiêu chuẩn nén hình ảnh không mất dữ liệu được hỗ trợ rộng rãi với mã nguồn mở sẵn có.
image/gif	Hình ảnh GIF	không ý kiến	GIF là một định dạng phổ biến được hỗ trợ rất tốt trên toàn cầu. Tuy nhiên, định dạng GIF có vướng vấn đề tác quyền và do đó nên được sử dụng một cách thận

Mã	Tên	Trạng thái	Định nghĩa
			trọng.
image/jpeg	Hình ảnh JPEG	bắt buộc	Định dạng này là bắt buộc để đạt độ nén cao với các hình ảnh màu nặng. Kiểu dữ liệu này nén "có mất dữ liệu", nhưng mắt thường hầu như không thể nhận ra sự khác biệt với nén không mất mát dữ liệu.
application/dicom	DICOM	Khuyến nghị	Kiểu hình ảnh số và truyền thông trong y tế (DICOM) MIME được định nghĩa trong tiêu chuẩn RFC3240 [http://ietf.org/rfc/rfc3240.txt].
image/g3fax	Hình ảnh G3Fax	Khuyến nghị	Kiểu dữ liệu này chỉ được khuyến nghị sử dụng dành cho các ứng dụng fax.
image/tiff	Hình ảnh TIFF	không ý kiến	Mặc dù định dạng TIFF (Tag Image File Format – Định dạng file hình ảnh nhãn) là một tiêu chuẩn quốc tế, nó có nhiều vấn đề về khả năng liên thông trong thực tế. Có quá nhiều phiên bản khác nhau không được tất cả các phần mềm xử lý giống nhau.
video/mpeg	Video (đoạn phim) MPEG	bắt buộc	MPEG là một tiêu chuẩn quốc tế, được triển khai rộng rãi, rất hiệu quả đối với các video (đoạn phim) màu chất lượng cao; có mã nguồn mở, có khả năng liên thông cao.
video/x-avi	Video (đoạn phim) X-AVI	không tán thành	Định dạng tập tin AVI chỉ là một trình bao bọc cho rất nhiều bộ mã hóa-giải mã (codec) khác nhau;

Mã	Tên	Trạng thái	Định nghĩa
			đây là nguồn gốc của nhiều vấn đề về khả năng liên thông.
model/vrml	Mô hình VRML	khuyến nghị	Đây là một định dạng được tiêu chuẩn hóa mở cho các mô hình 3D, có thể hữu ích cho các ứng dụng thực tế ảo như nghiên cứu giải phẫu hoặc nghiên cứu sinh hóa (trực quan hóa cấu trúc nguyên tử của các đại phân tử)

Tập hợp các kiểu phương tiện bắt buộc là rất nhỏ để cho không có yêu cầu bất hợp lý nào đối với các ứng dụng của HL7, đặc biệt là các hệ thống kế thừa. Nói chung, không có ứng dụng HL7 nào bị buộc phải hỗ trợ bất kỳ kiểu phương tiện nào đã cho ngoài văn bản. Ví dụ, nhiều hệ thống không muốn nhận dữ liệu âm thanh, vì chúng chỉ có thể hiển thị văn bản viết cho người dùng. Khi nói "tôi sẽ không xử lý âm thanh" thì đó chỉ là vấn đề về tính tương hợp với ứng dụng mà thôi. Chỉ khi hệ thống tuyên bố sẽ xử lý phương tiện âm thanh, thì nó mới phải hỗ trợ kiểu phương tiện bắt buộc cho âm thanh

2.4.2 Bộ ký tự: CS

Định nghĩa: Đối với các kiểu mã hóa dựa trên ký tự, đặc tính này xác định tập hợp ký tự và mã hóa ký tự được sử dụng. Tập hợp ký tự phải được nhận biết bởi một Đăng ký Tập hợp Ký tự của Tổ chức cấp phát số hiệu Internet (IANA) [http://www.iana.org/assignments/character-sets] phù hợp với RFC 2978 [http://www.ietf.org/rfc/rfc2978.txt].

Miền tập hợp ký tự do *Tổ chức cấp phát số hiệu Internet* (IANA) [http://www.iana.org/assignments/character-sets] duy trì. Nguồn của IANA xác định tên và nhiều bí danh cho hầu hết các tập hợp ký tự. Đối với các mục đích của HL7, không được phép sử dụng nhiều tên bí danh. Tên tiêu chuẩn cho HL7 là tên được IANA gắn mác là "nên dùng cho MIME." Nếu IANA chưa gắn mác cho một trong số các bí danh là "nên dùng cho MIME", thì tên được sử dụng cho HL7 phải là tên chính.

Bảng 7. liệt kê một số tập hợp ký tự do IANA định nghĩa được các thành viên hiện nay của HL7 quan tâm.

mã	tên	định nghĩa

mã	tên	định nghĩa
EBCDIC	EBCDIC	HL7 không có ý kiến gì về việc sử dụng Tập hợp ký tự này.
ISO-10646-UCS-2	ISO-10646-UCS-2	Không tán thành cho sử dụng HL7.
ISO-10646-UCS-4	ISO-10646-UCS-4	Không tán thành cho sử dụng HL7.
ISO-8859-1	ISO-8859-1	HL7 không có ý kiến gì về việc sử dụng Tập hợp ký tự này.
ISO-8859-2	ISO-8859-2	HL7 không có ý kiến gì về việc sử dụng Tập hợp ký tự này.
ISO-8859-5	ISO-8859-5	HL7 không có ý kiến gì về việc sử dụng Tập hợp ký tự này.
JIS-2022-JP	JIS-2022-JP	HL7 không có ý kiến gì về việc sử dụng Tập hợp ký tự này.
US-ASCII	US-ASCII	Bắt buộc sử dụng cho HL7.
UTF-7	UTF-7	HL7 không có ý kiến gì về việc sử dụng Tập hợp ký tự này.
UTF-8	UTF-8	Bắt buộc để hỗ trợ Unicode.

LƯU Ý: Danh sách trên đây không hoàn chỉnh và càng không phải là duy nhất. Cụ thể, các chi nhánh quốc tế của HL7 có thể đưa ra những đề xuất đặc biệt về các tập hợp ký tự được sử dụng trong khu vực của họ. Các đề xuất này có thể bổ sung các tập hợp ký tự khác và có thể gán lại một trạng thái đề xuất khác cho một tập hợp ký tự đã được liệt kê.

Cần biết đặc tính tập hợp ký tự khi dữ liệu của kiểu dữ liệu *ED* là dữ liệu kiểu ký tự dưới bất kỳ hình thức nào. Nếu dữ liệu được cung cấp nội tuyến, thì phải biết được tập hợp ký tự. Nếu dữ liệu được cung cấp qua tham chiếu, và phương pháp tiếp cận không cung cấp tập hợp ký tự cho dữ liệu, thường dưới dạng một phần đầu MIME, thì tập hợp ký tự phải được chuyển tải như một phần của kiểu dữ liệu *ED*.

Độc giả quan tâm cũng có thể tham khảo "Mô hình ký tự cho World Wide Web" [http://www.w3.org/TR/charmod] để có được thảo luận đầy đủ hơn về tập hợp ký tự và các vấn đề liên quan.

2.4.3 Ngôn ngữ: CS

Định nghĩa: Đối với thông tin dựa trên ký tự, đặc tính ngôn ngữ chỉ ra ngôn ngữ mà con người dùng cho văn bản

Nhu cầu phải có mã ngôn ngữ cho các giá trị dữ liệu văn bản được ghi chép trong RFC 2277, Chính sách IETF về Tập hợp ký tự và Ngôn ngữ [http://www.ietf.org/rfc/rfc2277.txt]. Có thể xem thêm thông tin liên quan trong Sử dụng Ký tự Quốc tế trong Thư tín Internet [http://www.imc.org/mail-i18n.html], một biên bản ghi nhớ của Liên minh Thư tín Internet.

Các nguyên tắc miền mã của thuộc tính này do tiêu chuẩn Internet RFC 3066 [http://www.ietf.org/rfc/rfc3066.txt] quy định. Sơ đồ mã hóa RFC 3066 được xây dựng từ một thành phần nhãn phụ cơ bản được mã hóa bằng các mã ngôn ngữ của ISO 639, cộng với hai mã phần mở rộng cho các ngôn ngữ không được biểu diễn trong ISO 639. Mã bao gồm, ko bắt buộc, một thành phần nhãn phụ thứ hai được mã hóa bằng mã nước gồm hai chữ cái của ISO 3166, hoặc một phần mở rộng mã ngôn ngữ do Tổ chức cấp phát số hiệu Internet đăng ký [http://www.iana.org/assignments/language-tags].[17]

Mặc dù các nhãn ngôn ngữ thường thay đổi ý nghĩa của văn bản, ngôn ngữ không thay đổi ý nghĩa của các ký tự trong văn bản.[18]

LƯU Ý: Việc biểu diễn các nhãn ngôn ngữ sang văn bản phụ thuộc nhiều vào ITS. ITS có thể sử dụng cách gắn nhãn ngôn ngữ riêng mà công nghệ triển khai mục tiêu của nó cung cấp. Một số có thể có thông tin ngôn ngữ trong một thành phần riêng biệt, ví dụ, XML có nhãn xml:lang cho các chuỗi ký tự. Một số khác có thể dựa vào nhãn ngôn ngữ như một phần của biểu diễn chuỗi ký tự nhị phân, ví dụ, ISO 10646 (Unicode) và nhãn ngôn ngữ "plane-14" của nó.

Nhãn ngôn ngữ không nên là bắt buộc nếu nó là không bắt buộc trong công nghệ triển khai. Về mặt ngữ nghĩa, việc gắn nhãn ngôn ngữ của chuỗi tuân theo một logic mặc định. Trong những trường hợp khi một khu vực có thể hỗ trợ nhiều ngôn ngữ, khu vực đó sẽ quyết định các quy tắc xử lý ngôn ngữ khi không có ngôn ngữ nào được xác định. Nếu không có quy tắc nào khác được xác định, ngôn ngữ địa phương của người đọc sẽ được sử dụng. Nếu một ngôn ngữ được thiết lập cho toàn bộ một bản tin hay tài liệu, ngôn ngữ đó là mặc định. Nếu có một giá trị hay phần tử thông tin bất kỳ ở cấp cao hơn trong hệ thống phân cấp cú pháp xác định một ngôn ngữ, ngôn ngữ đó sẽ là mặc định cho tất cả các giá trị văn bản cấp dưới.

Nếu có nhãn ngôn ngữ trong phần đầu của văn bản nhị phân được mã hóa (ví dụ, thông qua các nhãn plane-14 của Unicode) đây là nguồn đặc tính ngôn ngữ của giá trị dữ liệu được đóng gói.

2.4.4 Nén: CS

Định nghĩa: Cho biết liệu dữ liệu byte thô có được nén không, và thuật toán nén được sử dụng là gì.

Bảng 8. Miền Thuật toán nén (CompressionAlgorithm)

mã	tên	tình trạng	định nghĩa
DF	deflate	bắt buộc	Định dạng dữ liệu nén deflate như được quy định trong tiêu chuẩn RFC 1951 [http://www.ietf.org/rfc/rfc1951.txt].
GZ	gzip	không ý kiến	Một định dạng dữ liệu nén tương thích với tiện ích được sử dụng rộng rãi GZIP như được quy định trong tiêu chuẩn RFC 1952 [http://www.ietf.org/rfc/rfc1952.txt] (sử dụng thuật toán nén deflate)
ZL	zlib	không ý kiến	Một định dạng dữ liệu nén cũng sử dụng thuật toán nén deflate. Được quy định trong tiêu chuẩn RFC 1950 [http://www.ietf.org/rfc/rfc1952.txt]
Z	compress	không tán thành	Thuật toán nén UNIX gốc và định dạng file dùng thuật toán LZC (một biến thể của LZW). Có vướng mắc về bản quyền và không hiệu quả bằng thuật toán deflate.

Không bao giờ nén các giá trị của kiểu ST.

2.4.5 Tham chiếu: TEL

Định nghĩa: Một địa chỉ viễn thông (TEL), như địa chỉ URL cho HTTP hoặc FTP, sẽ cung cấp chính xác cùng một dữ liệu nhị phân mà cũng có thể được cung cấp như dữ liệu nội tuyến.

Giá trị ngữ nghĩa của một giá trị dữ liệu được đóng gói là như nhau, bất kể dữ liệu xuất hiện dưới dạng dữ liệu nội tuyến hay qua tham chiếu. Tuy nhiên, một giá trị dữ liệu được đóng gói mà không có dữ liệu nội tuyến sẽ hành xử khác, do bất kỳ nỗ lực nào muốn kiểm tra dữ liệu đều đòi hỏi tải dữ liệu xuống từ đường tham chiếu. Một giá trị dữ liệu được đóng gói có thể có cả dữ liệu nội tuyến và qua tham chiếu.

Tham chiếu phải dẫn tới cùng một dữ liệu như được cung cấp nội tuyến. Sẽ là xảy ra lỗi nếu dữ liệu có được qua tham chiếu không khớp với hoặc kiểm tra tính toàn vẹn, hoặc dữ liệu nội tuyến, hoặc dữ liệu mà trước đó được lấy qua tham chiếu và sau đó được lưu trữ.

Tham chiếu có thể chứa một *Giai đoạn có thể sử dụng* để báo rằng dữ liệu có thể chỉ sẵn có trong một khoảng thời gian hữu hạn. Dù tham chiếu có bị giới hạn bởi Giai đoạn có thể sử dụng hay không, thì nội dung của tham chiếu vẫn luôn là cố định. Bất kỳ ứng dụng nào sử dụng tham chiếu phải luôn nhận được cùng một dữ liệu. Không thể tái sử dụng tham chiếu để gửi một phiên bản khác của cùng một dữ liệu, hay một dữ liệu khác.

Dữ liệu được đóng gói qua tham chiếu có thể không được chấp nhận tùy thuộc vào thuộc tính hay thành phần được khai báo là dữ liệu được đóng gói. Các giá trị của kiểu ST phải luôn là nội tuyến.

2.4.6 Kiểm tra tính toàn vẹn: BIN

Định nghĩa: Kiểm tra tính toàn vẹn là một giá trị nhị phân ngắn biểu diễn một tổng kiểm tra đủ mạnh về mặt mã hóa được tính trên các dữ liệu nhị phân. Mục đích của đặc tính này, khi được thông tin cùng với một tham chiếu, là để sau này kiểm tra xem đường tham chiếu có dẫn đến cùng dữ liệu mà đường tham chiếu đã dẫn tới khi giá trị dữ liệu được đóng gói có chứa đường tham chiếu được tạo ra hay không.

Sẽ là xảy ra lỗi nếu dữ liệu có được thông qua tham chiếu không khớp với kiểm tra tính toàn vẹn.

Kiểm tra tính toàn vẹn được tính toán dựa theo Thuật toán kiểm tra tính toàn vẹn. Theo mặc định, phải sử dụng *Thuật toán Băm An toàn 1* (SHA-1). Kiểm tra tính toàn vẹn được mã hóa nhị phân dựa theo các quy tắc của thuật toán kiểm tra tính toàn vẹn.

Kiểm tra tính toàn vẹn được tính toán trên dữ liệu nhị phân thô, được chứa trong thành phần dữ liệu hoặc có thể tiếp cận thông qua tham chiếu. Không có biến đổi nào được thực hiện trước khi kiểm tra tính toàn vẹn được tính toán. Nếu là dữ liệu nén, Kiểm tra tính toàn vẹn được tính toán trên dữ liệu đã nén.

2.4.7 Thuật toán kiểm tra tính toàn vẹn: CS

Định nghĩa: Xác định thuật toán được dùng để tính giá trị Kiểm tra tính toàn vẹn.[19]

Bảng 9. Miền Thuật toán Kiểm tra Tính toàn vẹn

mã	tên	định nghĩa

mã	tên	định nghĩa
SHA-1	thuật toán băm an toàn - 1	Thuật toán này được định nghĩa trong FIPS PUB 180-1: Tiêu chuẩn băm an toàn. Kể từ ngày 17 tháng tư năm 1995.
SHA-256	thuật toán băm an toàn - 256	Thuật toán này được định nghĩa trong FIPS PUB 180-2: Tiêu chuẩn băm an toàn.

2.4.8 Biểu tượng thu nhỏ: ED

Định nghĩa: Một dạng biểu diễn ngắn gọn của dữ liệu đầy đủ. Biểu tượng thu nhỏ yêu cầu ít tài nguyên hơn rất nhiều so với dữ liệu đầy đủ, trong khi vẫn duy trì được sự tương đồng đặc biệt với dữ liệu đầy đủ. Biểu tượng thu nhỏ thường được dùng với dữ liệu được đóng gói có tham chiếu. Nó cho phép người dùng chọn dữ liệu hiệu quả hơn trước khi thực sự tải xuống qua đường tham chiếu

Ban đầu, thuật ngữ biểu tượng thu nhỏ dùng để chỉ một hình ảnh có độ phân giải thấp hơn (hay kích thước nhỏ hơn) một hình ảnh khác. Tuy nhiên, khái niệm biểu tượng thu nhỏ có thể được sử dụng theo cách ẩn dụ cho các kiểu phương tiện không phải là hình ảnh. Ví dụ, có thể biểu diễn một bộ phim bằng một clip ngắn hơn; biểu diễn một audio clip bằng một audio clip khác ngắn hơn, có tốc độ lấy mẫu thấp hơn, hoặc nén có mất mát dữ liệu.

Biểu tượng thu nhỏ có thể không được chấp nhận tùy thuộc vào thuộc tính hoặc thành phần được khai báo là dữ liệu được đóng gói. Các giá trị của kiểu ST không bao giờ có biểu tượng thu nhỏ, và một biểu tượng thu nhỏ không thể tự nó chứa một biểu tượng thu nhỏ.

```
invariant(ED x)
    where x.thumbnail.nonNull {
  x.thumbnail.thumbnail.isNull;
};
```

LƯU Ý: ITS nên cân nhắc trường hợp khi cả biểu tượng thu nhỏ và bản gốc đều có cùng các đặc tính của kiểu, tập hợp ký tự và nén. Trong trường hợp này, không cần biểu diễn rõ ràng các đặc tính này cho biểu tượng thu nhỏ nhưng chúng có thể được "thừa kế" từ giá trị dữ liệu được đóng gói chính sang cho biểu tượng thu nhỏ của nó.

2.4.9 Tương đương: BL, được thừa kế từ ANY

Hai giá trị của kiểu dữ liệu *ED* là bình đẳng khi và chỉ khi kiểu phương tiện và dữ liệu của chúng là bình đẳng. Với những giá trị *ED* có dữ liệu nén hoặc dữ liệu tham chiếu, chỉ những dữ liệu đã được bỏ tham chiếu và không nén mới được tính cho kiểm

tra tính bình đẳng. Bản thân các đặc tính nén, biểu tượng thu nhỏ và tham chiếu bị loại khỏi kiểm tra tính bình đẳng. Ngoài ra, đặc tính ngôn ngữ cũng bị loại khỏi kiểm tra, do nó có thể mang lại những giá trị của kiểu *ED* trong đó ngôn ngữ không được xác định. Nếu mediaType (Kiểu phương tiện) dựa trên ký tự và đặc tính tập hợp ký tự là không bình đẳng, đặc tính tập hợp ký tự phải được quyết định thông qua ánh xạ dữ liệu giữa các tập hợp ký tự khác nhau.

Thuật toán kiểm tra tính toàn vẹn và kiểm tra tính toàn vẹn bị loại khỏi kiểm tra tính bình đẳng. Tuy nhiên, do tính bình đẳng của giá trị kiểm tra tính toàn vẹn là một chỉ báo mạnh mẽ cho bình đẳng dữ liệu, trên thực tế kiểm tra tính bình đẳng có thể dựa trên kiểm tra tính toàn vẹn, căn cứ vào các đặc tính thuật toán kiểm tra tính toàn vẹn bình đẳng.

2.5 Chuỗi ký tự (ST) chuyên biệt hóa ED

Định nghĩa: Kiểu dữ liệu chuỗi ký tự đại diện cho dữ liệu văn bản, chủ yếu dành cho máy xử lý (ví dụ, sắp xếp, truy vấn, lập chỉ mục, v.v.) Được dùng cho tên, ký hiệu, và các biểu thức chính thống.

ST là một ED bị thu hẹp, với đặc tính ED.mediaType của nó cố định ở *text/plain*, và dữ liệu của nó phải là nội tuyến và không bị nén. Vì vậy, các đặc tính nén, tham chiếu, kiểm tra tính toàn vẹn, thuật toán, và biểu tượng thu nhỏ là không áp dụng được. Kiểu dữ liệu chuỗi ký tự được sử dụng khi hình thức của văn bản không mang nghĩa, điều này đúng đối với văn bản hình thức hóa và tất cả các kiểu tên.

Bảng 10. Tóm tắt đặc tính của Chuỗi ký tự

Tên	Kiểu	Mô tả
mediaType (Kiểu phương tiện)	CS	Xác định kiểu của dữ liệu được đóng gói và xác định một phương pháp để diễn giải hoặc trình diễn dữ liệu.
charset (tập ký tự)	CS	Đối với các kiểu mã hóa dựa trên ký tự, đặc tính này xác định tập ký tự và mã hóa ký tự được sử dụng. Tập ký tự phải được nhận biết bởi một Đăng ký Tập hợp Ký tự của Tổ chức cấp phát số hiệu Internet (IANA) [http://www.iana.org/assignments/character-sets] phù hợp với RFC 2978 [http://www.ietf.org/rfc/rfc2978.txt].
language (ngôn ngữ)	CS	Đối với thông tin dựa trên ký tự, đặc tính ngôn ngữ chỉ ra ngôn ngữ mà con người dùng trong văn bản.

Kiểu dữ liệu *ST* thể hiện dữ liệu được đóng gói như dữ liệu ký tự (trái ngược với bit), tùy thuộc vào đặc tính chuỗi ký tự của kiểu dữ liệu được đóng gói.

```
type CharacterString alias ST specializes ED {
  INT  length;
  ST   headCharacter;
  ST   tailString;
};
```

LƯU Ý: Do nhiều đặc tính của dữ liệu được đóng gói bị ràng buộc với một giá trị mặc định, ITS hoàn toàn không cần phải biểu diễn các đặc tính này. Trong thực tế, nếu mã hóa ký tự cũng bị cố định, ITS chỉ biểu diễn dữ liệu ký tự được mã hóa.

Các đặc tính Ký tự đầu và Ký tự đuôi định nghĩa *ST* là một chuỗi các thực thể với mỗi thực thể xác định duy nhất một ký tự từ tập hợp chung của tất cả các ký tự mà bất kỳ ngôn ngữ nào trên thế giới cũng biết đến.[20]

Đầu của một *ST* là một chuỗi với chỉ một ký tự. Một *ST* phải có ít nhất một ký tự nếu không nó có giá trị NULL. Một *ST* có độ dài bằng 0 là một giá trị ngoại lệ (NULL), không phải một giá trị hợp lệ.

```
invariant(ST x)
    where x.nonNull {
  x.headCharacter.notEmpty;
  x.headCharacter.length.equal(1);
  x.headCharacter.tailString.isEmpty;
  x.tailString.isEmpty.implies(x.length.equal(1));
  x.tailString.notEmpty.implies(x.length.equal(x.tailString.length.successor));
};
```

Độ dài của một *ST* là số ký tự, chứ không phải số byte được mã hóa, trong chuỗi. Mã hóa byte là vấn đề của ITS và không phù hợp ở lớp ứng dụng.

Các quy tắc dưới đây áp dụng cho khoảng trắng được chứa trong các giá trị của kiểu *ST*:

- TAB, dấu cách và cuối dòng đều được coi là các ký tự khoảng trắng.
- Cả khoảng trắng trước và sau đều quan trọng.
- Không thể hoán đổi các ký tự khoảng trắng khác nhau.
- Các cách biểu diễn cuối dòng khác nhau được tiêu chuẩn hóa dựa theo phương pháp được mô tả trong đặc tả XML [Phần 2.11 Xử lý cuối dòng]
- Không thể nén các chuỗi khoảng trắng thành các chuỗi ngắn hơn.

Yêu cầu

ST là một chuyên biệt hóa của ED sao cho bất kỳ thuộc tính RIM nào có kiểu ED đều có thể bị ràng buộc với một *ST*. Trường hợp quan trọng nhất là Act.text,

vốn là một ED để phục vụ cho việc sử dụng tham chiếu và dữ liệu đa phương tiện, nhưng thường bị ràng buộc với văn bản đơn thuần.

2.5.1 Kiểu dữ liệu Media: CS, được thừa kế từ ED

```
invariant(ST x)
    where x.nonNull {
  x.mediaType.equal("text/plain");
};
```

Được cố định là "text/plain".

2.5.2 Bộ ký tự: CS, thừa kế từ ED

```
invariant(ST x)
    where x.nonNull {
  x.charset.nonNull;
};
```

Các giá trị của kiểu *ST* phải có một tập hợp ký tự đã biết.

2.5.3 Ngôn ngữ: CS, thừa kế từ ED

Định nghĩa: Đối với thông tin dựa trên ký tự, đặc tính ngôn ngữ chỉ ra ngôn ngữ mà con người dùng trong văn bản.

Nhu cầu phải có mã ngôn ngữ cho các giá trị dữ liệu văn bản được ghi chép trong RFC 2277, Chính sách IETF về Tập hợp ký tự và Ngôn ngữ [http://www.ietf.org/rfc/rfc2277.txt]. Có thể xem thêm thông tin liên quan trong Sử dụng Ký tự Quốc tế trong Thư tín Internet [http://www.imc.org/mail-i18n.html], một biên bản ghi nhớ của Liên minh Thư tín Internet.

Các nguyên tắc của miền mã của thuộc tính này do tiêu chuẩn Internet RFC 3066 [http://www.ietf.org/rfc/rfc3066.txt] quy định. Sơ đồ mã hóa RFC 3066 được xây dựng từ một thành phần nhãn phụ cơ bản được mã hóa bằng các mã ngôn ngữ của ISO 639, cộng với hai mã phần mở rộng cho các ngôn ngữ không được biểu diễn trong ISO 639. Mã bao gồm, ko bắt buộc, một thành phần nhãn phụ thứ hai được mã hóa bằng mã nước gồm hai chữ cái của ISO 3166, hoặc một phần mở rộng mã ngôn ngữ do Tổ chức cấp phát số hiệu Internet đăng ký [http://www.iana.org/assignments/language-tags].[21]

Mặc dù các nhãn ngôn ngữ thường thay đổi ý nghĩa của văn bản, ngôn ngữ không thay đổi ý nghĩa của các ký tự trong văn bản.[22]

LƯU Ý: Việc biểu diễn các nhãn ngôn ngữ sang văn bản phụ thuộc nhiều vào ITS. ITS có thể sử dụng cách gắn nhãn ngôn ngữ riêng mà công nghệ triển khai mục tiêu của nó cung cấp. Một số có thể có thông tin ngôn ngữ trong một thành phần riêng

biệt, ví dụ, XML có nhãn xml:lang cho các chuỗi. Một số khác có thể dựa vào nhãn ngôn ngữ như một phần của biểu diễn chuỗi ký tự nhị phân, ví dụ, ISO 10646 (Unicode) và nhãn ngôn ngữ "plane-14" của nó.

Nhãn ngôn ngữ không nên là bắt buộc nếu nó là không bắt buộc trong công nghệ triển khai. Về mặt ngữ nghĩa, việc gắn nhãn ngôn ngữ của chuỗi tuân theo một logic mặc định. Trong những trường hợp khi một khu vực có thể hỗ trợ nhiều ngôn ngữ, khu vực đó sẽ quyết định các quy tắc xử lý ngôn ngữ khi không có ngôn ngữ nào được xác định. Nếu không có quy tắc nào khác được xác định, ngôn ngữ địa phương của người đọc sẽ được sử dụng. Nếu một ngôn ngữ được thiết lập cho toàn bộ một bản tin hay tài liệu, ngôn ngữ đó là mặc định. Nếu có một giá trị hay phần tử thông tin bất kỳ ở cấp cao hơn trong hệ thống phân cấp cú pháp xác định một ngôn ngữ, ngôn ngữ đó sẽ là mặc định cho tất cả các giá trị văn bản cấp dưới.

Nếu có nhãn ngôn ngữ trong phần đầu của văn bản nhị phân được mã hóa (ví dụ, thông qua các nhãn plane-14 của Unicode) đây là nguồn đặc tính ngôn ngữ của giá trị dữ liệu được đóng gói.

2.5.4 Nén: CS, (cố định)

```
invariant(ST x)
    where x.nonNull {
  x.compression.notApplicable;
};
```

Không thể nén các giá trị của kiểu *ST*.

2.5.5 Tham chiếu: TEL, (cố định)

```
invariant(ST x)
    where x.nonNull {
  x.reference.notApplicable;
};
```

Các giá trị của kiểu *ST* có thể không tham chiếu nội dung từ một vị trí nào đó khác.

2.5.6 Kiểm tra tính toàn vẹn: BIN, (cố định)

```
invariant(ST x)
    where x.nonNull {
  x.integrityCheck.notApplicable;
};
```

Không sử dụng mã kiểm tra tính toàn vẹn với các giá trị của kiểu *ST*.

2.5.7 Thuật toán kiểm tra tính toàn vẹn: CS, (cố định)

```
invariant(ST x)
    where x.nonNull {
  x.integrityCheckAlgorithm.notApplicable;
};
```

Không sử dụng thuật toán kiểm tra tính toàn vẹn với các giá trị của kiểu *ST*.

2.5.8 Biểu tượng thu nhỏ: ED, (cố định)

```
invariant(ST x)
    where x.nonNull {
  x.thumbnail.notApplicable;
};
```

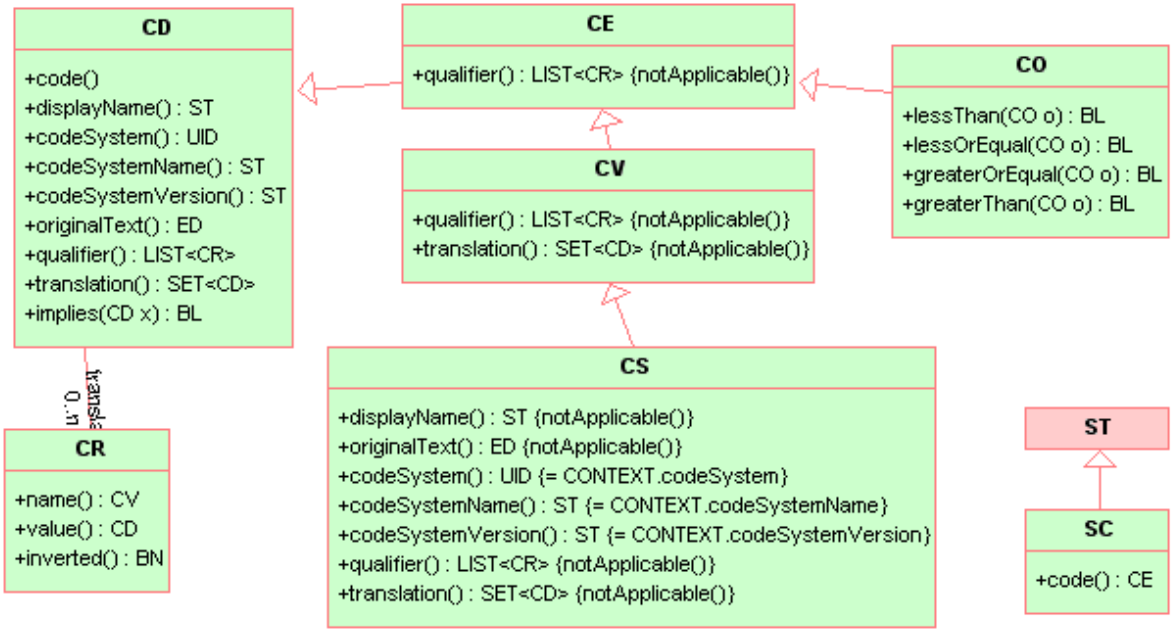
Các giá trị của kiểu *ST* không có biểu tượng thu nhỏ.

2.5.9 Dạng chữ

Hai biến thể của dạng chữ *ST* được định nghĩa, một là dạng thẻ và một là chuỗi trích dẫn.[23] Dạng thẻ chỉ bao gồm các chữ cái Latin viết thường và viết hoa, mười chữ số thập phân và nét gạch dưới. Chuỗi trích dẫn có thể chứa bất kỳ ký tự nào giữa hai dấu ngoặc kép. Dấu ngoặc kép ngăn không cho một chuỗi ký tự bị hiểu thành một dạng chữ nào đó khác. Dạng thẻ cho phép phân tích từ khóa và tên từ ngôn ngữ đặc tả kiểu dữ liệu.

```
ST.literal ST {
  ST : /"[^]+"/        { $.equal($1); }  /* quoted string */
    | /[a-zA-Z0-9_]+/  { $.equal($1); }; /* token form */
};
```

LƯU Ý: Vì dạng chữ *ST* là rất cơ bản đối với công nghệ triển khai, hầu hết các ITS sẽ xác định một dạng chữ chuỗi ký tự biến đổi nào đó. Tuy nhiên, các nhà thiết kế ITS phải nhận thức được sự tương tác giữa dạng chữ *ST* và các dạng chữ được định nghĩa cho các kiểu dữ liệu khác. Điều này đặc biệt quan trọng nếu dạng chữ của các kiểu dữ liệu khác được cấu trúc với các thành phần chính được phân tách bằng các ký tự ngắt (ví dụ, số thực, đại lượng vật lý, tập hợp, và dạng chữ danh sách, v.v.)



Mô hình thông tin Bộ mô tả khái niệm.

2.6 Mô tả khái niệm (CD) chuyên biệt hóa ANY

Định nghĩa: Một *CD* biểu diễn loại khái niệm bất kỳ, thông thường bằng cách gán cho một mã đã được xác định trong một hệ thống mã. Một *CD* có thể chứa văn bản hoặc cụm từ gốc dùng làm cơ sở để mã hóa và một hoặc nhiều dạng chuyển ngữ sang các hệ thống mã hóa khác nhau. Một *CD* cũng có thể chứa các từ hạn định để mô tả, ví dụ khái niệm "bàn chân trái" là một thuật ngữ hậu kết hợp được xây dựng từ mã chính là "BÀN CHÂN" và từ hạn định là "TRÁI". Trong các trường hợp có một giá trị ngoại lệ, kiểu dữ liệu *CD* không cần chứa mã mà chỉ có văn bản gốc mô tả khái niệm.

Bảng 11. Tóm tắt đặc tính của bộ mô tả khái niệm

Tên	Kiểu	Mô tả
code (mã)	ST	Ký hiệu mã đơn giản được hệ thống mã định nghĩa. Ví dụ, "784.0" là ký hiệu mã của mã ICD-9 "784.0" cho bệnh đau đầu.
codeSystem (Hệ thống mã)	UID	Xác định hệ thống mã định nghĩa mã.
codeSystemName (Tên hệ thống mã)	ST	Tên thông thường của hệ thống mã hóa.
codeSystemVersion (Phiên bản Hệ thống mã)	ST	Nếu áp dụng được, thì đây là một bộ mô tả phiên bản được xác định riêng cho hệ thống mã đang được đề cập đến.

Tên	Kiểu	Mô tả
displayName (Tên hiển thị)	ST	Tên hoặc tiêu đề của mã, hệ thống gửi trình bày giá trị mã dưới tên đó cho người sử dụng.
originalText (Văn bản gốc)	ED	Văn bản hoặc cụm từ dùng làm cơ sở để mã hóa.
translation (chuyển ngữ)	SET<CD>	Một tập hợp các bộ mô tả khái niệm khác giúp chuyển ngữ bộ mô tả khái niệm này sang các hệ thống mã khác.
qualifier (từ hạn định)	LIST<CR>	Xác định các mã bổ sung làm tăng tính riêng biệt của mã chính.

```
type ConceptDescriptor alias CD specializes ANY {
        ST      code;
        ST      displayName;
        UID     codeSystem;
        ST      codeSystemName;
        ST      codeSystemVersion;
        ED      originalText;
        LIST<CR>  qualifier;
        SET<CD>   translation;
        BL      equal(ANY x);
        BL      implies(CD x);
 demotion  ED;
};
```

Kiểu dữ liệu *CD* thường được sử dụng là một trong các hình thức thu hẹp hay "sơ lược" của nó, CS, CE, CV. Việc sử dụng toàn bộ kiểu dữ liệu mô tả khái niệm là không phổ biến. Nó đòi hỏi một quyết định tỉnh táo và lí do cho nó phải được ghi lại thành văn bản. Trong tất cả các trường hợp khác, phải sử dụng một trong các dạng thu hẹp của kiểu dữ liệu *CD*.[24]

Tất cả các dạng thu hẹp của kiểu dữ liệu *CD* đều hạn chế các đặc tính nhất định. Các đặc tính có thể bị hạn chế tới mức chỉ có một giá trị được chấp nhận cho đặc tính đó, trong trường hợp này việc đề cập tới đặc tính trở thành thừa thãi. Hạn chế đặc tính ở một giá trị được gọi là ngăn chặn đặc tính đó. Mặc dù về mặt khái niệm một đặc tính bị ngăn chặn vẫn có thể áp dụng được về mặt ngữ nghĩa, sẽ là an toàn nếu một giao diện kết nối HL7 nhận giá trị mặc định ngầm mà không kiểm tra.

LƯU Ý: Nhìn chung, điều này đúng đối với nhiều kiểu dữ liệu trong đặc tả kiểu dữ liệu này, tuy nhiên các dẫn xuất của *CD* là câu hỏi thường được nêu lên.

2.6.1 Mã: ST

Định nghĩa: Ký hiệu mã đơn giản được hệ thống mã định nghĩa. Ví dụ, "784.0" là ký hiệu mã của mã ICD-9 "784.0" cho bệnh đau đầu.

Một giá trị *CD* không ngoại lệ có một đặc tính *mã* non-NULL với giá trị của nó là một chuỗi ký tự mà là một ký hiệu được định nghĩa bởi hệ thống mã hóa do codeSystem (Hệ thống mã) nhận diện. Ngược lại, một giá trị *CD* không có giá trị cho đặc tính mã, hoặc có giá trị không phải từ hệ thống mã hóa được trích dẫn, là một giá trị ngoại lệ (NULL thuộc sắc thái *khác*).

```
invariant(CD x)
    where x.nonNull {
  x.code.nonNull;
};
```

2.6.2 Hệ thống mã: UID

Định nghĩa: Xác định hệ thống mã định nghĩa mã.

Hệ thống mã phải được gọi bằng một UID, để có một tham chiếu rõ ràng tới các mã HL7 tiêu chuẩn, các hệ thống mã tiêu chuẩn khác, cũng như các mã địa phương. HL7 phải gán một UID cho từng bảng mã của nó cũng như cho các hệ thống mã hóa tiêu chuẩn bên ngoài được sử dụng cùng HL7. Các trang địa phương phải sử dụng Định danh đối tượng ISO (OID) của chúng để xây dựng một định danh hệ thống mã hóa địa phương duy nhất toàn cầu.

Dưới nhánh của HL7, 2.16.840.1.113883, các nhánh phụ 5 và 6 lần lượt chứa định danh tiêu chuẩn HL7 và định danh hệ thống mã bên ngoài. Ủy ban Kỹ thuật Từ vựng HL7 duy trì hai nhánh phụ này.

Một giá trị *CD* không ngoại lệ (nghĩa là một giá trị *CD* có đặc tính mã non-null) có một *Hệ thống mã* non-NULL xác định hệ thống khái niệm định nghĩa mã. Nói cách khác, bất cứ khi nào có mã thì cũng có một hệ thống mã.

LƯU Ý: Mặc dù mọi giá trị *CD* non-NULL đều có một hệ thống mã được định nghĩa, trong một vài trường hợp, biểu diễn của ITS cho giá trị *CD* không cần đề cập rõ ràng tới hệ thống mã. Ví dụ, khi bối cảnh quy định bắt buộc có một và chỉ một hệ thống mã được sử dụng, việc chỉ rõ hệ thống mã sẽ trở nên thừa thãi. Tuy nhiên, trong trường hợp đó *Hệ thống mã* chấp nhận giá trị mặc định của bối cảnh cụ thể đó và là không NULL.

```
invariant(CD x)
    where x.code.nonNull {
```

```
    x.codeSystem.nonNull;
};
```

Một *CD* ngoại lệ của trạng thái null *khác* cho biết không thể mã hóa một khái niệm trong hệ thống mã hóa được xác định. Vì vậy, đối với các ngoại lệ về mã hóa này, hệ thống mã không chứa khái niệm thích hợp phải được cung cấp trong *Hệ thống mã*.

Một vài miền mã được chuẩn bị điều kiện để chứa một phần của bất kỳ hệ thống mã hóa địa phương phù hợp nào không chỉ đơn thuần diễn giải hệ thống mã hóa tiêu chuẩn (*được mã hóa với khả năng mở rộng*, CWE.) Nếu một trường dữ liệu đủ điều kiện CWE thực sự chứa một mã địa phương như vậy, hệ thống mã hóa phải xác định hệ thống mã hóa địa phương mà từ đó mã địa phương được lấy. Tuy nhiên, đối với các miền CWE mã địa phương là một thành viên hợp lệ của miền, để các mã địa phương trong miền CWE không tạo ra lỗi cũng không tạo ra một giá trị ngoại lệ (NULL/khác) theo ý nghĩa của đặc tả này.

```
invariant(CD x)
    where x.other {
  x.code.other;
  x.codeSystem.nonNull;
};
```

2.6.3 Tên hệ thống mã: ST

Định nghĩa: Tên thông thường của hệ thống mã hóa.

Tên hệ thống mã không có giá trị tính toán. Mục đích của tên hệ thống mã là để hỗ trợ một người diễn dịch không được hỗ trợ của một giá trị mã trong việc diễn dịch codeSystem (Hệ thống mã). Chúng tôi đề xuất — tuy nhiên không thực sự bắt buộc — rằng ITS hỗ trợ *Tên hệ thống mã* để chú giải UID sao cho con người có thể hiểu được.

Các hệ thống HL7 không được dựa về mặt chức năng vào *Tên hệ thống mã*. *Tên hệ thống mã* không bao giờ có thể thay đổi ý nghĩa của Hệ thống mã và không thể tồn tại mà không có Hệ thống mã.

```
invariant(CD x) {
  x.codeSystemName.nonNull.implies(x.codeSystem.nonNull);
};
```

2.6.4 Phiên bản hệ thống mã: ST

Định nghĩa: Nếu áp dụng được, thì đây là một bộ mô tả phiên bản được xác định riêng cho hệ thống mã đang được đề cập đến.

HL7 phải xác định các chuỗi phiên bản này được hình thành như thế nào cho từng hệ thống mã bên ngoài. Nếu HL7 chưa xác định các chuỗi phiên bản được hình thành như thế nào cho một hệ thống mã hóa cụ thể, thì chỉ định phiên bản không có ý nghĩa được định nghĩa cho hệ thống mã hóa đó.

Các phiên bản khác nhau của một hệ thống mã phải tương thích. Bất cứ khi nào một hệ thống mã thay đổi theo hướng không tương thích, nó sẽ tạo ra một hệ thống mã mới, không đơn giản là một phiên bản khác, bất kể nhà xuất bản từ vựng gọi nó là gì.

Ví dụ, nhà xuất bản của bộ mã ICD-9 và ICD-10 gọi các hệ thống mã này lần lượt là "hiệu chỉnh 9" và "hiệu chỉnh 10". Tuy nhiên, ICD-10 là thiết kế lại hoàn toàn của mã ICD, chứ không phải một phiên bản tương thích ngược. Vì vậy, đối với mục đích của đặc tả kiểu dữ liệu này, ICD-9 và ICD-10 là các hệ thống mã khác biệt, chứ không chỉ là các phiên bản khác biệt. Ngược lại, khi LOINC cập nhật từ hiệu chỉnh "1.0j" lên "1.0k", HL7 sẽ coi đây chỉ là một phiên bản khác của LOINC, do các hiệu chỉnh LOINC có tính tương thích ngược.

```
invariant(CD x) {
  x.codeSystemVersion.nonNull.implies(x.codeSystem.nonNull);
};
```

2.6.5 Tên hiển thị: ST

Định nghĩa: Tên hoặc tiêu đề của mã, hệ thống gửi trình bày giá trị mã dưới tên đó cho người sử dụng.

Tên hiển thị được đưa vào vừa như một hình thức hỗ trợ cho một người diễn dịch không được hỗ trợ của một giá trị mã, tên được sử dụng được định dạng dưới dạng văn bản để hiển thị khái niệm cho người sử dụng. Tên hiển thị không có ý nghĩa chức năng; nó không bao giờ có thể tồn tại mà không có mã đi kèm; và nó không bao giờ có thể thay đổi ý nghĩa của mã.

LƯU Ý: HL7 cung cấp "tên in ấn" trong các miền từ vựng được định nghĩa trước của mình. Các giá trị này phù hợp để sử dụng trong displayName (Tên hiển thị).

LƯU Ý: Tên hiển thị không thể thay đổi ý nghĩa của giá trị mã. Vì vậy, không nên trình bày tên hiển thị với người sử dụng trên một hệ thống ứng dụng nhận mà không biết chắc rằng tên hiển thị đó biểu diễn một cách thỏa đáng khái niệm được giá trị mã nhắc tới. Việc thông tin liên lạc không được chỉ dựa vào tên hiển thị. Mục đích chính của tên hiển thị là để hỗ trợ việc chỉnh lỗi của các đơn vị dữ liệu giao thức HL7 (ví dụ, bản tin).

2.6.6 Văn bản gốc: ED

Định nghĩa: Văn bản hoặc cụm từ dùng làm cơ sở để mã hóa.

Văn bản gốc tồn tại trong một kịch bản khi người khởi tạo thông tin không gán mã, mà sau này mã được gán bởi một người tạo mã (mã hóa sau). Trong quá trình tạo ra một bộ mô tả khái niệm, văn bản gốc do đó có thể tồn tại mà không có mã.

LƯU Ý: Mặc dù mã hóa sau thường được thực hiện từ thông tin văn bản tự do, như tài liệu, hình quét (scan) hay chính tả, dữ liệu đa phương tiện rõ ràng không được chấp nhận là văn bản gốc. Ngoài ra, đặc tính văn bản gốc không phải là một liên kết tới toàn bộ tài liệu nguồn. Liên kết giữa các sản phẩm thông tin y tế khác nhau (ví dụ, tài liệu và kết quả được mã hóa) nằm ngoài phạm vi của đặc tả này và được duy trì ở phần khác trong các đặc tả tiêu chuẩn HL7. Văn bản gốc là một đoạn trích các thông tin liên quan trong các nguồn ban đầu, chứ không phải một con trỏ văn bản hay sản phẩm chính xác. Vì vậy, văn bản gốc được biểu diễn dưới dạng văn bản đơn giản.

Các giá trị của kiểu dữ liệu *CD* có thể có đặc tính văn bản gốc non-NULL dù cho nó có mã là NULL. Bất kỳ giá trị *CD* nào có mã NULL đều biểu thị một ngoại lệ mã hóa. Trong trường hợp này, *Văn bản gốc (originalText)* là tên hay mô tả của khái niệm không được mã hóa. Các giá trị *CD* ngoại lệ như vậy cũng có thể chứa các chuyển ngữ. Các chuyển ngữ này mã hóa trực tiếp khái niệm được mô tả trong *Văn bản gốc*.

Một giá trị *CD* có thể được chuyển đổi giảm thành một giá trị ST chỉ biểu diễn *Văn bản gốc* của giá trị *CD*.

```
invariant(CD x)
    where x.originalText.nonNull {
  ((ST)x).equal(x.originalText);
};
```

2.6.7 Chuyển ngôn ngữ: SET<CD>

Định nghĩa: Một tập hợp các bộ mô tả khái niệm khác giúp chuyển ngữ bộ mô tả khái niệm này sang các hệ thống mã khác.

Chuyển ngữ là một tập hợp các giá trị *CD* khác với mỗi *CD* trong số đó chuyển ngữ *CD* đầu tiên sang các hệ thống mã khác. Mỗi phần tử của tập hợp chuyển ngữ được chuyển ngữ từ giá trị *CD* đầu tiên. Tuy nhiên, mỗi chuyển ngữ cũng có thể chứa nhiều chuyển ngữ. Vì vậy, khi một mã được chuyển ngữ nhiều lần, thông tin về việc mã nào được dùng làm đầu vào cho chuyển ngữ nào sẽ được bảo tồn và duy trì.

LƯU Ý: Các chuyển ngữ gần như là các đồng nghĩa của một khái niệm có thực. Mọi chuyển ngữ trong tập hợp được cho là thể hiện cùng một ý nghĩa "bằng các ngôn từ khác nhau." Tuy nhiên, khó có thể tồn tại một đồng nghĩa tuyệt đối giữa hai hệ thống mã hóa khác nhau về cấu trúc. Vì vậy, không phải tất cả các chuyển ngữ đều có độ chính xác như nhau.

2.6.8 Từ hạn định: LIST<CR>

Định nghĩa: Xác định các mã bổ sung làm tăng tính riêng biệt của mã chính.

Mã chính và tất cả các từ hạn định cùng với nhau tạo nên một khái niệm. Một giá trị *CD* với các từ hạn định cũng được gọi là một *cụm từ mã (code phrase)* hoặc *biểu thức hậu kết hợp (postcoordinated)*.

Các từ hạn định hạn chế ý nghĩa của mã chính, nhưng không thể phủ định nó hay thay đổi nghĩa của nó thành nghĩa của một giá trị khác trong hệ thống mã hóa chính.

Chỉ có thể sử dụng các từ hạn định dựa theo các quy tắc đã được định nghĩa rõ ràng về hậu kết hợp. Một giá trị của kiểu *CD* chỉ có thể có các từ hạn định nếu hệ thống mã của nó định nghĩa việc sử dụng các từ hạn định đó hoặc nếu có một hệ thống mã thứ ba xác định cách thức có thể kết hợp các hệ thống mã khác.

Ví dụ, SNOMED CT cho phép xây dựng các khái niệm là sự kết hợp của nhiều mã. SNOMED CT định nghĩa một khái niệm "viêm mô tế bào (rối loạn)" (128045006) một thuộc tính "vị trí tìm kiếm" (363698007) và một khái niệm khác "cấu trúc chân (cấu trúc cơ thể)" (56459004). SNOMED CT cho phép kết hợp các mã này trong một cụm từ mã:

Ví dụ 1:

```
<observation>
  ...
  <value code="128045006" codeSystem="&amp;amp;SNOMED-CT;" displayName="cellulitis (disorder)">
    <qualifier code="56459004" displayName="foot structure">
      <name code="363698007" displayName="finding site"/>
    </qualifier>
  </value>
  ...
</observation>
```

Trong ví dụ này, có một hệ thống mã, SNOMED-CT định nghĩa mã chính và tất cả các từ hạn định cũng như chúng được sử dụng như thế nào, đó là lí do vì sao trong ví dụ của chúng ta việc biểu diễn Hệ thống mã không cần phải được đề cập tới cho tên từ hạn định và giá trị (Hệ thống mã được thừa kế từ mã chính).

Điều quan trọng cần lưu ý là hệ thống mã xác định các từ hạn định có thể được chấp nhận. Ví dụ, trong SNOMED CT, có một tập hợp được định nghĩa của các thuộc tính đủ điều kiện, và chỉ Kết quả và Rối loạn là có thể đủ điều kiện cho thuộc tính "vị trí tìm kiếm". Việc sử dụng các từ hạn định bên ngoài phạm vi mà hệ thống mã đã chỉ

ra là cách sử dụng không phù hợp kiểu dữ liệu *CD*. Tuân thủ các quy tắc mà hệ thống mã đã chỉ ra cho phép so sánh các biểu thức hậu kết hợp với các khái niệm tiền kết hợp (ví dụ như khi có thể so sánh cụm từ mã trên đây với khái niệm tiền kết hợp "viêm mô tế bào ở chân (rối loạn)" (128276007) - khái niệm được xác định trong bộ mã SNOMED CT là có một vị trí tìm kiếm là cấu trúc chân). Kiểu dữ liệu *CD* không hỗ trợ việc tiêu chuẩn hóa các biểu thức thành phần, vì vậy có thể tạo ra các biểu thức không rõ ràng. Người sử dụng nên hiểu rằng họ phải cung cấp các ràng buộc bổ sung cần thiết để đảm bảo việc biểu diễn dữ liệu một cách rõ ràng, nếu họ định tạo ra các biểu thức thành phần sử dụng kiểu dữ liệu *CD*. Nếu không, họ phải đối mặt với nguy cơ là không thể tìm được một tập hợp đầy đủ tất cả các hồ sơ tương ứng với bất kỳ truy vấn cụ thể nào.

Một ví dụ phổ biến khác là các mã thủ tục của các Trung tâm dịch vụ bảo hiểm y tế cho người già và người nghèo của Hoa Kỳ (CMS) (trước đây được biết đến dưới tên gọi Cơ quan quản trị tài chính y tế, HCFA). Các mã thủ tục CMS (HCPCS) được dựa trên bộ mã CPT-4 và bổ sung thêm các từ hạn định cho nó. Ví dụ, bệnh nhân với phát hiện như trên (cộng với bệnh động mạch ngoại biên, tiểu đường, và tổn thương da mãn tính ở ngón chân cái bên trái) có thể phải cắt ngón chân đó. Khái niệm CPT-4 là "Cắt, khớp đốt bàn chân-ngón chân" (28820) và cần bổ sung một từ định lượng HCPCS để chỉ rõ "bàn chân trái, ngón cái" (TA). Vì vậy chúng ta mã hóa như sau:

Ví dụ 2:

```
<procedure>
  ...
  <cd code="28820" codeSystem="&amp;amp;CP4;" displayName="Amputation, toe metatarsophalangeal joint">
    <qualifier code="TA" codeSystem="&amp;amp;HCP;" displayName="left foot, great toe"/>
  </cd>
  ...
</procedure>
```

Trong ví dụ này, hệ thống mã của từ hạn định (HCPCS) khác với hệ thống mã của mã chính (CPT-4). Chỉ vì có các quy tắc được định nghĩa rõ ràng định nghĩa cách thức liên kết các mã này mà các từ hạn định có thể được sử dụng. Cũng cần lưu ý rằng tên vai trò là tùy chọn, và đối với các mã HCPCS không có các tên vai trò riêng biệt.

Trật tự của các từ hạn định được bảo tồn, đặc biệt đối với trường hợp hệ thống mã hóa cho phép hậu kết hợp nhưng không định nghĩa tên vai trò (ví dụ, một số mã ICD-9-CM, hoặc mã "đa trục" SNOMED cũ).

2.6.9 Tương đương: BL, thừa kế từ ANY

Các bộ mô tả khái niệm chủ yếu được sử dụng để phục vụ việc lập chỉ mục, truy vấn và đưa ra quyết định dựa trên một giá trị được mã hóa. Vì vậy, một đặc tả rõ ràng về ngữ nghĩa các giá trị được mã hóa đòi hỏi một định nghĩa rõ ràng về việc tính bình đẳng của các giá trị bộ mô tả khái niệm có ý nghĩa gì và các giá trị *CD* nên được so sánh thế nào. (Để biết thêm chi tiết về so sánh các biểu thức tiền và hậu kết hợp, xem Dolin RH, Spackman KA, Markwell D. Truy tìm có chọn lọc các khái niệm SNOMED tiền và hậu kết hợp. Mùa thu AMIA 2002; 210-14, hoặc Hướng dẫn triển khai SNOMED CT tháng bảy 2003).

Tính bình đẳng giữa hai giá trị *CD* được quyết định chỉ dựa trên mã (code) và Hệ thống mã (codeSystem). Phiên bản Hệ thống mã (codeSystemVersion) bị loại khỏi kiểm tra tính bình đẳng.[25] Nếu có các từ hạn định, chúng được đưa vào kiểm tra tính bình đẳng. Kiểm tra tính bình đẳng không bao gồm các chuyển ngữ.[26] Các giá trị *CD* ngoại lệ là không bình đẳng ngay cả khi chúng có cùng NULL-flavor (trạng thái rỗng) hoặc có cùng văn bản gốc.[27]

```
invariant(CD x, y)
    where x.nonNull.and(y.nonNull) {
  x.equal(y).equal(x.code.equal(y.code)
        .and(x.codeSystem.equal(y.codingSystem))
        .and(x.qualifier.equal(y.qualifier)));
};
```

Một vài hệ thống mã xác định các tùy chọn có kiểu nhất định cho các giá trị mã của chúng. Ví dụ, Mã thuốc quốc gia của Hoa Kỳ (NDC) có dạng có gạch ngang và không gạch ngang. Một ví dụ của dạng có gạch ngang là có thể là 1234-5678-90 trong khi dạng không gạch ngang là 01234567890. Một ví dụ khác cho vấn đề này là khi các bảng mã ISO hoặc ANSI nhất định định nghĩa các dạng chữ-số và số tùy chọn với độ dài hai hay ba ký tự trong một tiêu chuẩn.

Trong trường hợp các hệ thống mã hỗ trợ nhiều phương thức biểu diễn, HL7 sẽ ra quyết định về dạng được ưu tiên. HL7 phải có văn bản cho quyết định đó khi hệ thống mã hóa bên ngoài tương ứng đó được công nhận. HL7 phải lựa chọn dạng được ưu tiêu dựa trên các tiêu chí thực tiễn và sử dụng phổ biến. Khi không có các tiêu chí rõ ràng về tính thực tiễn và sử dụng phổ biến, thì hình thức an toàn nhất, dễ mở rộng nhất, và ít cách điệu hóa nhất (ít được trang trí nhất) phải được ưu tiên.[28]

2.6.10 Hàm ý: BL

Định nghĩa: Xác định liệu kiểu dữ liệu *CD* này có là một chuyên biệt hóa của toán hạng *CD*.

Một cách tự nhiên, có thể thu hẹp hoặc mở rộng các khái niệm để bao gồm hoặc loại trừ các khái niệm khác. Nhiều hệ thống mã hóa có một ý niệm rõ ràng về chuyên biệt hóa và khái quát hóa khái niệm. Các nguyên tắc từ vựng của HL7 cũng hỗ trợ chuyên biệt hóa khái niệm cho các tập hợp giá trị đã được định nghĩa của HL7. N*gụ ý (Implies)* là một vị từ so sánh xem liệu một khái niệm có phải là một chuyên biệt hóa của một khái niệm khác, và từ đó ngụ ý khái niệm khác đó.

Khi viết vị từ (ví dụ, các câu điều kiện) so sánh hai mã, nên đôi lúc kiểm tra tất suy chứ không phải tính bình đẳng của mã.

Ví dụ, trong Bảng 20 các khái niệm "sử dụng viễn thông – telecommunication use": cơ quan (W), nhà (H), nhà chính (HP), và nhà nghỉ trong dịp lễ (HV) được định nghĩa, trong đó cả HP và HV ngụ ý H. Khi lựa chọn bất kỳ số điện thoại nhà nào, nên kiểm tra liệu mã sử dụng đã cho *c* có *ngụ ý* H. Kiểm tra *c* có *bình đẳng* với H sẽ chỉ tìm ra số điện thoại nhà không xác định, chứ không tìm ra số điện thoại nhà chính.

Về thao tác, có thể đánh giá tất suy bằng một trong hai cách. Dạng chữ hệ thống mã có thể được thiết kế sao cho một hệ thống phân cấp duy nhất được phản ánh trong bản thân dạng chữ mã (ví dụ, ICD-9). Tuy nhiên, ngoại trừ các trường hợp đặc biệt như vậy, sẽ cần một nền tảng kiến thức về thuật ngữ và một thuật toán gộp thích hợp để đánh giá các câu tất suy. Đối với các hệ thống mã hóa sau kết hợp, thiết kế một thuật toán gộp như vậy là một nhiệm vụ không hề đơn giản.[29]

2.7 Vai trò khái niệm (CR) chuyên biệt hóa ANY

Định nghĩa: Một mã từ hạn định khái niệm với vai trò được đặt tên một cách tùy chọn. Cả vai trò từ hạn định và mã giá trị phải được định nghĩa bởi hệ thống mã hóa của CD có chứa từ hạn định khái niệm. Ví dụ, nếu SNOMED RT định nghĩa một khái niệm "chân", một quan hệ vai trò "có-thuận bên", và một khái niệm khác "trái", quan hệ vai trò khái niệm cho phép bổ sung từ hạn định "có-thuận bên: trái" vào mã chính "chân" để tạo ý nghĩa "chân trái".

Bảng 12. Tóm tắt đặc tính Vai trò khái niệm

Tên	Kiểu	Mô tả
name (tên)	CV	Xác định thể thức theo đó giá trị vai trò khái niệm đóng góp vào nghĩa của cụm từ mã. Ví dụ, nếu bộ mã SNOMED RT định nghĩa một khái niệm “chân”, một quan hệ vai trò “có-thuận bên”, và một khái niệm khác “trái”, quan hệ vai trò khái niệm cho phép bổ sung từ hạn định “có-thuận bên: trái” vào mã chính “chân” để tạo nghĩa “chân trái”. Trong ví dụ này, *tên* là “có-thuận bên”.
value (giá trị)	CD	Khái niệm điều chỉnh mã chính của cụm từ mã qua quan hệ vai trò. Ví dụ, nếu bộ mã SNOMED RT định nghĩa một khái niệm

Tên	Kiểu	Mô tả
		"chân", một quan hệ vai trò "có-thuận bên", và một khái niệm khác "trái", quan hệ vai trò khái niệm cho phép bổ sung từ hạn định "có-thuận bên: trái" vào mã chính "chân" để tạo nghĩa "chân trái". Trong ví dụ này, *giá trị* là "trái".
inverted (đảo ngược)	BN	Chỉ báo nếu nghĩa của tên bị đảo ngược. Nó có thể được sử dụng dùng trong các trường hợp khi hệ thống mã hóa cơ bản định nghĩa sự đảo ngược nhưng không cung cấp cặp tên vai trò thuận-nghịch. Theo mặc định, chỉ báo đảo ngược là *sai (false).*

Việc sử dụng các từ hạn định được quản lý chặt chẽ bởi hệ thống mã được sử dụng. Kiểu dữ liệu CD không cho phép dùng các từ hạn định mã với những hệ thống mã không hỗ trợ từ hạn định (ví dụ, các hệ thống tiền kết hợp, như LOINC, ICD-10 PCS).

```
protected type ConceptRole alias CR specializes ANY {
  CV  name;
  BN  inverted;
  CD  value;
};
```

2.7.1 Tên: CV

Định nghĩa: Xác định thể thức theo đó giá trị vai trò khái niệm đóng góp vào nghĩa của cụm từ mã. Ví dụ, nếu bộ mã SNOMED RT định nghĩa một khái niệm "chân", một quan hệ vai trò "có-thuận bên", và một khái niệm khác "trái", quan hệ vai trò khái niệm cho phép bổ sung từ hạn định "có-thuận bên: trái" vào mã chính "chân" để tạo nghĩa "chân trái". Trong ví dụ này, "có-thuận bên" là *tên*

2.7.2 Giá trị: CD

Định nghĩa: Khái niệm điều chỉnh mã chính của cụm từ mã qua quan hệ vai trò. Ví dụ, nếu bộ mã SNOMED RT định nghĩa một khái niệm "chân", một quan hệ vai trò "có-thuận bên", và một khái niệm khác "trái", quan hệ vai trò khái niệm cho phép bổ sung từ hạn định "có-thuận bên: trái" vào mã chính "chân" để tạo nghĩa "chân trái". Trong ví dụ này, "trái" là giá trị.

giá trị thuộc kiểu CD và do đó có thể lần lượt có các từ hạn định. Điều này cho phép các từ hạn định được lồng vào nhau. Các từ hạn định chỉ có thể được sử dụng khi nào hệ thống mã cơ sở định nghĩa chúng. Không được phép sử dụng bất kỳ kiểu từ hạn định nào cho các hệ thống mã không chấp nhận và điều chỉnh một cách rõ ràng việc sử dụng các từ hạn định này.

```
invariant(CR x)
```

```
    where x.nonNull {
  x.value.nonNull;
};
```

2.7.3 Chỉ báo đảo ngược: BN

Định nghĩa: Chỉ báo nếu nghĩa của tên bị đảo ngược. Nó có thể được sử dụng trong các trường hợp khi hệ thống mã hóa cơ bản định nghĩa sự đảo ngược nhưng không cung cấp cặp tên vai trò thuận-nghịch. Theo mặc định, chỉ báo đảo ngược là *sai (false).*

Ví dụ, một hệ thống mã có thể định nghĩa quan hệ vai trò "nguyên nhân" và các khái niệm "Phế cầu khuẩn - Streptococcus pneumoniae" và "Viêm phổi". Nếu hệ thống mã đó cho phép các vai trò của nó bị đảo ngược, có thể tạo khái niệm hậu kết hợp "Viêm phổi do phế cầu" thông qua "Viêm phổi - nguyên nhân, đảo ngược - Phế cầu khuẩn".

Chỉ có thể đảo ngược các vai trò nếu hệ thống mã hóa cơ sở cho phép đảo ngược này. Đáng chú ý là, nếu một hệ thống mã hóa định nghĩa các vai trò trong các cặp đảo ngược hoặc cố ý không định nghĩa một số đảo ngược nhất định, thì mã vai trò phù hợp (ví dụ, "gây ra bởi") phải được sử dụng thay vì đảo ngược. Phải biết được liệu đặc tính bị đảo ngược là *đúng (true)* hay *sai (false)*, do nếu nó có giá trị là NULL, thì không thể giải thích vai trò.

LƯU Ý: nên chuyển tải *đảo ngược* trong một thuộc tính chỉ báo, với giá trị mặc định là *sai (false)*. Bằng cách đó, không phải gửi chỉ báo đảo ngược khi vai trò không bị đảo ngược

2.8 Giá trị đơn giản được mã hóa (CS) chuyên biệt hóa CV

Định nghĩa: Dữ liệu được mã hóa trong hình thức đơn giản nhất, trong đó chỉ có mã không được xác định trước. Hệ thống mã và phiên bản hệ thống mã được cố định bởi ngữ cảnh mà giá trị *CS* xuất hiện. Kiểu dữ liệu *CS* được dùng cho các thuộc tính được mã hóa mà chỉ có một tập hợp giá trị do HL7 định nghĩa

Bảng 13. Tóm tắt đặc tính của Giá trị đơn giản được mã hóa

Tên	Kiểu	Mô tả
code (mã)	ST	Ký hiệu mã đơn giản được hệ thống mã định nghĩa. Ví dụ, “784.0” là ký hiệu mã của mã ICD-9 “784.0” cho bệnh đau đầu.

```
type CodedSimpleValue alias CS specializes CV {
        ST  code;
  literal  ST;
};
```

Chỉ có thể sử dụng kiểu dữ liệu *CS* cho một trong các trường hợp sau:

1. Cho một thuộc tính được mã hóa mà có chỉ một hệ thống mã được HL7 định nghĩa, và khi việc bổ sung mã vào tập hợp giá trị đó yêu cầu hành động HL7 chính thức (như hòa hợp.) Các thuộc tính được mã hóa này phải có kiểu *CS*.
2. Cho một thuộc tính trong đặc tả này mà được gán cho một hệ thống mã duy nhất được định nghĩa hoặc trong đặc tả này hoặc bên ngoài HL7 bởi một đơn vị có thẩm quyền đối với khái niệm và việc duy trì hệ thống mã đó.

Ví dụ, do ED chấp thuận thiết kế MIME, nó tin tưởng IETF trong việc quản lý kiểu phương tiện. Điều này bao gồm việc đặc tả chấp thuận cơ chế mở rộng được tạo ra trong mã kiểu phương tiện MIME (ví dụ, "application/x-myapp").

Đối với các giá trị *CS*, chỉ định từ hạn định miền sẽ luôn là giá trị CNE (*được mã hóa, không thể mở rộng*) và ngữ cảnh sẽ quyết định sử dụng giá trị HL7 nào.[30]

2.8.1 Mã: ST, thừa kế từ CD

```
invariant(CS x)
    where x.nonNull {
  x.code.nonNull;
};
```

2.8.2 Hệ thống mã: UID, (cố định)

Mọi giá trị *CS* non-NULL đều có một hệ thống mã được định nghĩa. Biểu diễn của ITS cho kiểu dữ liệu *CS* không cần đề cập rõ ràng tới hệ thống mã, bởi vì bối cảnh quy định bắt buộc có một và chỉ một hệ thống mã được sử dụng. Việc chỉ rõ hệ thống mã sẽ trở nên thừa thãi. Tuy nhiên, hệ thống mã chấp nhận giá trị mặc định của bối cảnh cụ thể đó và giá trị này là không NULL.

```
invariant(CS x)
    where x.code.nonNull {
  x.codeSystem.nonNull;
  x.codeSystem.equal(CONTEXT.codeSystem);
};
```

Một giá trị *CS* ngoại lệ thuộc NULL-flavor (trạng thái null) *khác* chỉ ra rằng không thể mã hóa một khái niệm trong hệ thống mã hóa được xác định. Trong các trường hợp này, mã phải có giá trị là NULL.

```
invariant(CS x)
    where x.other {
  x.code.isNull;
  x.codeSystem.nonNull;
};
```

2.8.3 Tên hệ thống mã: ST, (cố định)

```
invariant(CS x) {
  x.codeSystemName.equal(CONTEXT.codeSystemName);
};
```

2.8.4 Phiên bản hệ thống mã: ST, (cố định)

```
invariant(CS x) {
  x.codeSystemVersion.equal(CONTEXT.codeSystemVersion);
};
```

2.8.5 Tên hiển thị: ST, (cố định)

```
invariant(CS x) {
  x.displayName.notApplicable;
};
```

2.8.6 Văn bản gốc: ED, (cố định)

```
invariant(CS x) {
  x.originalText.notApplicable;
};
```

2.8.7 Chuyển ngôn ngữ: SET<CD>, (cố định)

```
invariant(CS x) {
  x.translation.notApplicable;
};
```

2.8.8 Từ hạn định: LIST<CR>, (cố định)

```
invariant(CS x) {
  x.qualifier.notApplicable;
};
```

2.8.9 Dạng chữ

```
CS.literal ST {
  ST : /[a-zA-Z0-9_]+/  { $.equal($1); };
};
```

Dạng chữ chuỗi ký tự của kiểu dữ liệu *CS* chủ yếu được định nghĩa để phục vụ mục đích của đặc tả này. Dạng chữ là một biểu diễn chuỗi ký tự của mã dành cho Hệ thống mã trong ngữ cảnh *CS*. Không thể định rõ Hệ thống mã hay Phiên bản Hệ thống mã từ bản thân dạng chữ, vì vậy dạng chữ chỉ sử dụng khi đã biết rõ ngữ cảnh.

2.9 Giá trị được mã hóa (CV) chuyên biệt hóa CE

Định nghĩa: Dữ liệu được mã hóa, chỉ xác định một mã, hệ thống mã và, không bắt buộc, tên hiển thị và văn bản gốc. Chỉ được dùng như kiểu các đặc tính của các kiểu dữ liệu khác.

Bảng 14. Tóm tắt đặc tính của giá trị được mã hóa

Tên	Kiểu	Mô tả
code (mã)	ST	Ký hiệu mã đơn giản được hệ thống mã định nghĩa. Ví dụ, "784.0" là ký hiệu mã của mã ICD-9 "784.0" cho bệnh đau đầu.
codeSystem (Hệ thống mã)	UID	Xác định hệ thống mã định nghĩa mã.
codeSystemName (Tên Hệ thống mã)	ST	Tên thông thường của hệ thống mã hóa.
codeSystemVersion (Phiên bản Hệ thống mã)	ST	Nếu áp dụng được, thì đây là một bộ mô tả phiên bản được xác định riêng cho hệ thống mã đang được đề cập đến.
displayName (Tên hiển thị)	ST	Tên hoặc tiêu đề của mã, hệ thống gửi trình bày giá trị mã dưới tên đó cho người sử dụng.
originalText (Văn bản gốc)	ED	Văn bản hoặc cụm từ dùng làm cơ sở để mã hóa.

```
type CodedValue alias CV specializes CE {
  ST   code;
  UID  codeSystem;
  ST   codeSystemName;
  ST   codeSystemVersion;
  ST   displayName;
  ED   originalText;
};
```

Kiểu dữ liệu *CV* được sử dụng khi có bất kỳ trường hợp sử dụng hợp lý nào yêu cầu chỉ gửi duy nhất một giá trị mã. Vì vậy, không nên sử dụng nó trong các trường hợp khi nhiều mã thay thế cho một giá trị nhất định được mong muốn. Kiểu này có thể dùng được với cả từ hạn định miền CNE (*được mã hóa, không thể mở rộng*) và CWE (*được mã hóa, có khả năng mở rộng*).

2.9.1 Mã: ST, thừa kế từ CD

Định nghĩa: Ký hiệu mã đơn giản được hệ thống mã định nghĩa. Ví dụ, "784.0" là ký hiệu mã của mã ICD-9 "784.0" cho bệnh đau đầu.

Một giá trị không ngoại lệ có một đặc tính *mã* non-NULL với giá trị của nó là một chuỗi ký tự mà là một ký hiệu được định nghĩa bởi hệ thống mã hóa do Hệ thống mã nhận diện. Ngược lại, một giá trị không có giá trị cho đặc tính mã, hoặc có giá trị không phải từ hệ thống mã hóa được trích dẫn, là một giá trị ngoại lệ (NULL thuộc sắc thái *khác*).

```
invariant(CD x)
    where x.nonNull {
  x.code.nonNull;
};
```

2.9.2 Hệ thống mã: UID, thừa kế từ CD

Định nghĩa: Xác định hệ thống mã định nghĩa mã.

Hệ thống mã phải được gọi bằng một UID, để có một tham chiếu rõ ràng tới các mã HL7 tiêu chuẩn, các hệ thống mã tiêu chuẩn khác, cũng như các mã địa phương. HL7 phải gán một UID cho từng bảng mã của nó cũng như cho các hệ thống mã hóa tiêu chuẩn bên ngoài được sử dụng cùng tiêu chuẩn HL7. Các địa phương triển khai phải sử dụng Định danh đối tượng ISO (OID) của chúng để xây dựng một định danh hệ thống mã hóa địa phương duy nhất toàn cầu

Dưới nhánh của HL7, 2.16.840.1.113883, các nhánh phụ 5 và 6 lần lượt chứa định danh tiêu chuẩn HL7 và định danh hệ thống mã bên ngoài. Ủy ban Kỹ thuật Từ vựng HL7 duy trì hai nhánh phụ này.

Một giá trị không ngoại lệ (nghĩa là một giá trị có đặc tính mã non-null) có một *Hệ thống mã* non-NULL xác định hệ thống khái niệm định nghĩa mã. Nói cách khác, bất cứ khi nào có mã thì cũng có một hệ thống mã.

LƯU Ý: Mặc dù mọi giá trị non-NULL đều có một hệ thống mã được định nghĩa, trong một vài trường hợp, biểu diễn của ITS cho giá trị không cần đề cập rõ ràng tới hệ thống mã. Ví dụ, khi bối cảnh quy định bắt buộc có một và chỉ một hệ thống mã được sử dụng, việc chỉ rõ hệ thống mã sẽ trở nên thừa thãi. Tuy nhiên, trong trường hợp đó *Hệ thống mã* chấp nhận giá trị mặc định của bối cảnh cụ thể đó và là không NULL.

```
invariant(CD x)
    where x.code.nonNull {
  x.codeSystem.nonNull;
};
```

Một ngoại lệ của NULL-flavor (trạng thái null) *khác* cho biết không thể mã hóa một khái niệm trong hệ thống mã hóa được xác định. Vì vậy, đối với các ngoại lệ về mã hóa này, hệ thống mã không chứa khái niệm thích hợp phải được cung cấp trong *Hệ thống mã.*

Một vài miền mã được chuẩn bị điều kiện để chứa một phần của bất kỳ hệ thống mã hóa địa phương phù hợp nào không chỉ đơn thuần diễn giải hệ thống mã hóa tiêu chuẩn (*được mã hóa với khả năng mở rộng*, kiểu dữ liệu CWE.) Nếu một trường dữ liệu CWE đủ điều kiện thực sự chứa một mã địa phương như vậy, hệ thống mã hóa phải xác định hệ thống mã hóa địa phương mà từ đó mã địa phương được truy xuất. Tuy nhiên, đối với các miền CWE mã địa phương là một thành viên hợp lệ của miền, để các mã địa phương trong miền CWE không tạo ra lỗi cũng không tạo ra một giá trị ngoại lệ (NULL/khác) theo ý nghĩa của đặc tả này.

```
invariant(CD x)
    where x.other {
  x.code.other;
  x.codeSystem.nonNull;
};
```

2.9.3 Tên hệ thống mã: ST, thừa kế từ CD

Định nghĩa: Tên thông thường của hệ thống mã hóa.

Tên hệ thống mã không có giá trị tính toán. Mục đích của tên hệ thống mã là để hỗ trợ một người diễn dịch không được hỗ trợ của một giá trị mã trong việc diễn dịch Hệ thống mã. Chúng tôi đề xuất — tuy nhiên không thực sự bắt buộc — rằng ITS hỗ trợ *Tên hệ thống mã* để chú giải UID sao cho con người có thể hiểu được.

Các hệ thống HL7 không được dựa về mặt chức năng vào *Tên hệ thống mã*. *Tên hệ thống mã* không bao giờ có thể thay đổi ý nghĩa của Hệ thống mã và không thể tồn tại mà không có Hệ thống mã

```
invariant(CD x) {
  x.codeSystemName.nonNull.implies(x.codeSystem.nonNull);
};
```

2.9.4 Phiên bản hệ thống mã: ST, thừa kế từ CD

Định nghĩa: Nếu áp dụng được, thì đây là một bộ mô tả phiên bản được xác định riêng cho hệ thống mã đang xét.

HL7 phải xác định các chuỗi phiên bản này được hình thành như thế nào cho từng hệ thống mã bên ngoài. Nếu HL7 chưa xác định các chuỗi phiên bản được hình thành như thế nào cho một hệ thống mã hóa cụ thể, thì chỉ định phiên bản không có ý nghĩa được định nghĩa cho hệ thống mã hóa đó.

Các phiên bản khác nhau của một hệ thống mã phải tương thích. Bất cứ khi nào một hệ thống mã thay đổi theo hướng không tương thích, nó sẽ tạo ra một hệ thống mã mới, không chỉ đơn giản là một phiên bản khác, bất kể nhà xuất bản từ vựng gọi nó là gì.

Ví dụ, nhà xuất bản của bộ mã ICD-9 và ICD-10 gọi các hệ thống mã này lần lượt là "hiệu chỉnh 9" và "hiệu chỉnh 10". Tuy nhiên, ICD-10 là thiết kế lại hoàn toàn của mã ICD, chứ không phải một phiên bản tương thích ngược. Vì vậy, đối với mục đích của đặc tả kiểu dữ liệu này, ICD-9 và ICD-10 là các hệ thống mã khác biệt, chứ không chỉ là các phiên bản khác biệt. Ngược lại, khi bộ mã LOINC cập nhật từ hiệu chỉnh "1.0j" lên "1.0k", HL7 sẽ coi đây chỉ là một phiên bản khác của LOINC, do các hiệu chỉnh LOINC có tính tương thích ngược.

```
invariant(CD x) {
  x.codeSystemVersion.nonNull.implies(x.codeSystem.nonNull);
};
```

2.9.5 Tên hiển thị: ST, kế thừa từ CD

Định nghĩa: Tên hoặc tiêu đề của mã, hệ thống gửi trình bày giá trị mã dưới tên đó cho người sử dụng.

Tên hiển thị được đưa vào vừa như một hình thức hỗ trợ cho một người diễn dịch không được hỗ trợ của một giá trị mã, vừa như một dạng văn bản cho tên được sử dụng để hiển thị khái niệm với người sử dụng. Tên hiển thị không có ý nghĩa chức năng; nó không bao giờ có thể tồn tại mà không có mã; và nó không bao giờ có thể thay đổi ý nghĩa của mã.

LƯU Ý: HL7 cung cấp "tên in ấn" trong các miền từ vựng được định nghĩa trước của mình. Các giá trị này phù hợp để sử dụng trong Tên hiển thị.

LƯU Ý: Tên hiển thị không thể thay đổi ý nghĩa của giá trị mã. Vì vậy, không nên trình bày tên hiển thị với người sử dụng trên một hệ thống ứng dụng nhận mà không biết chắc rằng tên hiển thị đó biểu diễn một cách thỏa đáng khái niệm được giá trị mã nhắc tới. Việc thông tin liên lạc không được chỉ dựa vào tên hiển thị. Mục đích chính của tên hiển thị là để hỗ trợ việc chỉnh lỗi của các đơn vị dữ liệu giao thức HL7 (ví dụ, bản tin).

```
invariant(CD x) {
  x.displayName.nonNull.implies(x.code.nonNull);
};
```

2.9.6 Văn bản gốc: ED, thừa kế từ CD

Định nghĩa: Văn bản hoặc cụm từ dùng làm cơ sở để mã hóa

Văn bản gốc tồn tại trong một kịch bản khi người khởi tạo thông tin không gán mã, mà sau này mã được gán bởi một người tạo mã (mã hóa sau). Trong quá trình tạo ra một bộ mô tả khái niệm, văn bản gốc do đó có thể tồn tại mà không có mã.

LƯU Ý: Mặc dù mã hóa sau thường được thực hiện từ thông tin văn bản tự do, như tài liệu, hình quét (scan) hay chính tả, dữ liệu đa phương tiện rõ ràng không được chấp nhận là văn bản gốc. Ngoài ra, đặc tính văn bản gốc không phải là một liên kết tới toàn bộ tài liệu nguồn. Liên kết giữa các tạo tác thông tin y tế khác nhau (ví dụ, tài liệu và kết quả được mã hóa) nằm ngoài phạm vi của đặc tả này và được duy trì ở phần khác trong các tiêu chuẩn HL7. Văn bản gốc là một đoạn trích các thông tin liên quan trong các nguồn ban đầu, chứ không phải một con trỏ văn bản hay tái tạo chính xác. Vì vậy, văn bản gốc được biểu diễn dưới dạng văn bản đơn giản.

Các giá trị của kiểu dữ liệu có thể có đặc tính văn bản gốc non-NULL dù cho nó có mã NULL. Bất kỳ giá trị nào có mã NULL đều biểu thị một ngoại lệ mã hóa. Trong trường hợp này, *Văn bản gốc* là tên hay mô tả của khái niệm không được mã hóa. Các giá trị ngoại lệ như vậy cũng có thể chứa các chuyển ngữ. Các chuyển ngữ này mã hóa trực tiếp khái niệm được mô tả trong *Văn bản gốc*.

Một giá trị *CD* có thể được chuyển đổi giảm thành một giá trị ST chỉ biểu diễn *Văn bản gốc* của giá trị.

```
invariant(CD x)
    where x.originalText.nonNull {
  ((ST)x).equal(x.originalText);
};
```

2.9.7 Chuyển ngôn ngữ: SET<CD>, (cố định)

```
invariant(CV x) {
  x.translation.notApplicable;
};
```

2.9.8 Từ hạn định: LIST<CR>, (cố định)

```
invariant(CV x) {
  x.qualifier.notApplicable;
};
```

2.10 Số thứ tự được mã hóa (CO) chuyên biệt hóa CV

Định nghĩa: Dữ liệu được mã hóa, mà trong hệ thống mã hóa mã này được xếp thứ tự. *CO* bổ sung ngữ nghĩa liên quan đến việc xếp thứ tự sao cho các mô hình sử dụng các miền từ vựng đó có thể đưa vào sử dụng các phần tử mô hình có bao hàm các câu lệnh về thứ tự của các thuật ngữ trong một miền.

```
type CodedOrdinal alias CO specializes CV {
```

```
  BL  lessOrEqual(CO o);
  BL  lessThan(CO o);
  BL  greaterThan(CO o);
  BL  greaterOrEqual(CO o);
};
```

Thứ tự tương đối của các giá trị *CO* không cần phải rõ ràng một cách độc lập trong biểu diễn dạng chữ của chúng. Một ứng dụng được kỳ vọng sẽ tra cứu thứ tự của các giá trị này từ một bảng nào đó.

2.10.1 Nhỏ hơn hoặc bằng: BL

Định nghĩa: Quan hệ thứ tự dựa trên việc so sánh nhỏ-hơn-hay-bằng, vốn được coi là quan hệ nguyên thủy trong đặc tả này.

Tất cả các quan hệ thứ tự khác có thể bắt nguồn từ quan hệ này. Do *nhỏ hơn hay bằng* là nguyên thủy, nó cũng chứa sắp xếp thứ tự từng phần.

Các quan hệ thứ tự thường chỉ có giá trị trong chỉ một hệ thống mã hóa.

2.10.2 Nhỏ hơn: BL

```
invariant(CO x, y)
    where x.nonNull.and(y.nonNull) {
  x.lessThan(y).equal(y.lessOrEqual(x).and(x.equal(y).not));
};
```

2.10.3 Lớn hơn: BL

```
invariant(CO x, y)
    where x.nonNull.and(y.nonNull) {
  x.greaterThan(y).equal(y.lessThan(x));
};
```

2.10.4 Lớn hơn hoặc bằng: BL

```
invariant(CO x, y)
    where x.nonNull.and(y.nonNull) {
  x.greaterOrEqual(y).equal(y.lessOrEqual(x));
};
```

2.11 Được mã hóa có tương đương (CE) chuyên biệt hóa CD

Định nghĩa: Dữ liệu được mã hóa bao gồm một giá trị được mã hóa và, không bắt buộc, (các) giá trị được mã hóa từ các hệ thống mã hóa khác mà cùng định danh khái niệm đó. Được sử dụng khi có thể có các mã thay thế cùng tồn tại.

Bảng 15. Tóm tắt đặc tính của Được mã hóa có tương đương

Tên	Kiểu	Mô tả
code (mã)	ST	Ký hiệu mã đơn giản được hệ thống mã định nghĩa. Ví dụ, “784.0” là ký hiệu mã của mã ICD-9 “784.0” cho bệnh đau đầu
codeSystem (Hệ thống mã)	UID	Xác định hệ thống mã định nghĩa mã.
codeSystemName (Tên Hệ thống mã)	ST	Tên thông thường của hệ thống mã hóa.
codeSystemVersion (Phiên bản Hệ thống mã)	ST	Nếu áp dụng được, thì đây là một bộ mô tả phiên bản được xác định riêng cho hệ thống mã đang được đề cập đến.
displayName (Tên hiển thị)	ST	Tên hoặc tiêu đề của mã, hệ thống gửi trình bày giá trị mã dưới tên đó cho người sử dụng.
originalText (Văn bản gốc)	ED	Văn bản hoặc cụm từ dùng làm cơ sở để mã hóa
translation (chuyển ngữ)	SET<CD>	Một tập hợp các bộ mô tả khái niệm khác giúp chuyển ngữ bộ mô tả khái niệm này sang các hệ thống mã khác

```
type CodedWithEquivalents alias CE specializes CD {
  ST      code;
  UID     codeSystem;
  ST      codeSystemName;
  ST      codeSystemVersion;
  ST      displayName;
  ED      originalText;
  SET<CV>  translation;
};
```

Kiểu dữ liệu *CE* được sử dụng khi trường hợp sử dụng cho thấy có thể có các mã thay thế tồn tại và khi việc thông tin về chúng là hữu ích. *CE* hỗ trợ một giá trị mã chính, cộng với một tập hợp các biểu diễn thay thế hay tương đương.

2.11.1 Mã: ST, thừa kế từ CD

Định nghĩa: Ký hiệu mã đơn giản được hệ thống mã định nghĩa. Ví dụ, “784.0” là ký hiệu mã của mã ICD-9 “784.0” cho bệnh đau đầu.

Một giá trị không ngoại lệ có một đặc tính *mã* non-NULL với giá trị của nó là một chuỗi ký tự mà là một ký hiệu được định nghĩa bởi hệ thống mã hóa do Hệ thống mã nhận diện. Ngược lại, một giá trị không có giá trị cho đặc tính mã, hoặc có giá trị không phải từ hệ thống mã hóa được trích dẫn, là một giá trị ngoại lệ (NULL thuộc sắc thái *khác*)

```
invariant(CD x)
    where x.nonNull {
  x.code.nonNull;
};
```

2.11.2 Hệ thống mã: UID, thừa kế từ CD

Định nghĩa: Xác định hệ thống mã định nghĩa mã.

Hệ thống mã phải được gọi bằng một UID, để có một tham chiếu rõ ràng tới các mã HL7 tiêu chuẩn, các hệ thống mã tiêu chuẩn khác, cũng như các mã địa phương. HL7 phải gán một UID cho từng bảng mã của nó cũng như cho các hệ thống mã hóa tiêu chuẩn bên ngoài được sử dụng cùng HL7. Các địa phương triển khai phải sử dụng Định danh đối tượng ISO (OID) của chúng để xây dựng một định danh hệ thống mã hóa địa phương duy nhất toàn cầu.

Dưới nhánh của HL7, 2.16.840.1.113883, các nhánh phụ 5 và 6 lần lượt chứa định danh tiêu chuẩn HL7 và định danh hệ thống mã bên ngoài. Ủy ban Kỹ thuật Từ vựng HL7 duy trì hai nhánh phụ này.

Một giá trị không ngoại lệ (nghĩa là một giá trị có đặc tính mã non-null) có một *Hệ thống mã* non-NULL xác định hệ thống khái niệm định nghĩa mã. Nói cách khác, bất cứ khi nào có mã thì cũng có một hệ thống mã.

LƯU Ý: Mặc dù mọi giá trị non-NULL đều có một hệ thống mã được định nghĩa, trong một vài trường hợp, biểu diễn của ITS cho giá trị không cần đề cập rõ ràng tới hệ thống mã. Ví dụ, khi bối cảnh quy định bắt buộc có một và chỉ một hệ thống mã được sử dụng, việc chỉ rõ hệ thống mã sẽ trở nên thừa thãi. Tuy nhiên, trong trường hợp đó *Hệ thống mã* chấp nhận giá trị mặc định của bối cảnh cụ thể đó và có giá trị không NULL.

```
invariant(CD x)
    where x.code.nonNull {
  x.codeSystem.nonNull;
};
```

Một ngoại lệ của NULL-flavor (trạng thái null) *khác* cho biết không thể mã hóa một khái niệm trong hệ thống mã hóa được xác định. Vì vậy, đối với các ngoại lệ về mã hóa này, hệ thống mã không chứa khái niệm thích hợp phải được cung cấp trong *Hệ thống mã.*

Một vài miền mã được chuẩn bị điều kiện để chứa một phần của bất kỳ hệ thống mã hóa địa phương phù hợp nào không chỉ đơn thuần diễn giải hệ thống mã hóa tiêu chuẩn (*được mã hóa với khả năng mở rộng*, kiểu dữ liệu CWE.) Nếu một trường đủ điều kiện CWE thực sự chứa một mã địa phương như vậy, hệ thống mã hóa phải xác định hệ thống mã hóa địa phương mà từ đó mã địa phương được lấy. Tuy nhiên, đối với các miền CWE mã địa phương là một thành viên hợp lệ của miền, để các mã địa phương trong miền CWE không tạo ra lỗi cũng không tạo ra một giá trị ngoại lệ (NULL/khác) theo ý nghĩa của đặc tả này.

```
invariant(CD x)
    where x.other {
  x.code.other;
  x.codeSystem.nonNull;
};
```

2.11.3 Tên hệ thống mã: ST, thừa kế từ CD

Định nghĩa: Tên thông thường của hệ thống mã hóa.

Tên hệ thống mã không có giá trị tính toán. Mục đích của tên hệ thống mã là để hỗ trợ một người diễn dịch không được hỗ trợ của một giá trị mã trong việc diễn dịch Hệ thống mã. Chúng tôi đề xuất — tuy nhiên không thực sự bắt buộc — rằng ITS hỗ trợ *Tên hệ thống mã* để chú giải UID sao cho con người có thể hiểu được.

Các hệ thống HL7 không được dựa về mặt chức năng vào *Tên hệ thống mã*. *Tên hệ thống mã* không bao giờ có thể thay đổi ý nghĩa của Hệ thống mã và không thể tồn tại mà không có Hệ thống mã

```
invariant(CD x) {
  x.codeSystemName.nonNull.implies(x.codeSystem.nonNull);
};
```

2.11.4 Phiên bản hệ thống mã: ST, thừa kế từ CD

Định nghĩa: Nếu áp dụng được, thì đây là một bộ mô tả phiên bản được xác định riêng cho hệ thống mã đang xét.

HL7 phải xác định các chuỗi phiên bản này được hình thành như thế nào cho từng hệ thống mã bên ngoài. Nếu HL7 chưa xác định các chuỗi phiên bản được hình thành như thế nào cho một hệ thống mã hóa cụ thể, thì chỉ định phiên bản không có ý nghĩa được định nghĩa cho hệ thống mã hóa đó.

Các phiên bản khác nhau của một hệ thống mã phải tương thích. Bất cứ khi nào một hệ thống mã thay đổi theo hướng không tương thích, nó sẽ tạo ra một hệ thống mã mới, không đơn giản là một phiên bản khác, bất kể nhà xuất bản từ vựng gọi nó là gì.

Ví dụ, nhà xuất bản của ICD-9 và ICD-10 gọi các hệ thống mã này lần lượt là "hiệu chỉnh 9" và "hiệu chỉnh 10". Tuy nhiên, ICD-10 là thiết kế lại hoàn toàn của mã ICD, chứ không phải một phiên bản tương thích ngược. Vì vậy, đối với mục đích của đặc tả kiểu dữ liệu này, ICD-9 và ICD-10 là các hệ thống mã khác biệt, chứ không chỉ là các phiên bản khác biệt. Ngược lại, khi LOINC cập nhật từ hiệu chỉnh "1.0j" lên "1.0k", HL7 sẽ coi đây chỉ là một phiên bản khác của LOINC, do các hiệu chỉnh LOINC có tính tương thích ngược.

```
invariant(CD x) {
  x.codeSystemVersion.nonNull.implies(x.codeSystem.nonNull);
};
```

2.11.5 Tên hiển thị: ST, thừa kế từ CD

Định nghĩa: Tên hoặc tiêu đề của mã, hệ thống gửi trình bày giá trị mã dưới tên đó cho người sử dụng.

Tên hiển thị được đưa vào vừa như một hình thức hỗ trợ cho một người diễn dịch không được hỗ trợ của một giá trị mã, vừa như một dạng văn bản cho tên được sử dụng để hiển thị khái niệm với người sử dụng. Tên hiển thị không có ý nghĩa chức năng; nó không bao giờ có thể tồn tại mà không có mã; và nó không bao giờ có thể thay đổi ý nghĩa của mã.

LƯU Ý: HL7 cung cấp "tên in ấn" trong các miền từ vựng được định nghĩa trước của mình. Các giá trị này phù hợp để sử dụng trong Tên hiển thị.

LƯU Ý: Tên hiển thị không thể thay đổi ý nghĩa của giá trị mã. Vì vậy, không nên trình bày tên hiển thị với người sử dụng trên một hệ thống ứng dụng nhận mà không biết chắc rằng tên hiển thị đó biểu diễn một cách thỏa đáng khái niệm được giá trị mã nhắc tới. Việc thông tin liên lạc không được chỉ dựa vào tên hiển thị. Mục đích chính của tên hiển thị là để hỗ trợ việc chỉnh lỗi của các đơn vị dữ liệu giao thức HL7 (ví dụ, bản tin).

```
invariant(CD x) {
  x.displayName.nonNull.implies(x.code.nonNull);
};
```

2.11.6 Văn bản gốc: ED, thừa kế từ CD

Định nghĩa: Văn bản hoặc cụm từ được dùng làm cơ sở để mã hóa

Văn bản gốc tồn tại trong một kịch bản khi người khởi tạo thông tin không gán mã, nhưng sau này mã được gán bởi một người tạo mã (mã hóa sau). Trong quá trình tạo ra một bộ mô tả khái niệm, văn bản gốc do đó có thể tồn tại mà không có mã.

LƯU Ý: Mặc dù mã hóa sau thường được thực hiện từ thông tin văn bản tự do, như tài liệu, hình quét (scan) hay chính tả, dữ liệu đa phương tiện rõ ràng không được

chấp nhận là văn bản gốc. Ngoài ra, đặc tính văn bản gốc không phải là một liên kết tới toàn bộ tài liệu nguồn. Liên kết giữa các sàn phẩm thông tin y tế khác nhau (ví dụ, tài liệu và kết quả được mã hóa) nằm ngoài phạm vi của đặc tả này và được duy trì ở phần khác trong các tiêu chuẩn HL7. Văn bản gốc là một đoạn trích các thông tin liên quan trong các nguồn ban đầu, chứ không phải một con trỏ văn bản hay tái tạo chính xác. Vì vậy, văn bản gốc được biểu diễn dưới dạng văn bản đơn thuần.

Các giá trị của kiểu có thể có đặc tính văn bản gốc non-NULL dù cho nó có mã NULL. Bất kỳ giá trị nào có mã NULL đều biểu thị một ngoại lệ mã hóa. Trong trường hợp này, *Văn bản gốc* là tên hay mô tả của khái niệm không được mã hóa. Các giá trị ngoại lệ như vậy cũng có thể chứa các chuyển ngữ. Các chuyển ngữ này mã hóa trực tiếp khái niệm được mô tả trong *Văn bản gốc*.

Một giá trị *CD* có thể được chuyển đổi giảm thành một giá trị ST chỉ biểu diễn *Văn bản gốc* của giá trị.

```
invariant(CD x)
    where x.originalText.nonNull {
  ((ST)x).equal(x.originalText);
};
```

2.11.7 Chuyển ngôn ngữ: SET<CD>, thừa kế từ CD

Định nghĩa: Một tập hợp các bộ mô tả khái niệm khác giúp chuyển ngữ bộ mô tả khái niệm này sang các hệ thống mã khác

chuyển ngữ (translation) là một tập hợp các giá trị *CD* khác với mỗi giá trị *CD* trong số đó chuyển ngữ giá trị *CD* đầu tiên sang các hệ thống mã khác. Mỗi phần tử của tập hợp chuyển ngữ được chuyển ngữ từ giá trị *CD* đầu tiên. Tuy nhiên, mỗi chuyển ngữ cũng có thể chứa nhiều chuyển ngữ. Vì vậy, khi một mã được chuyển ngữ nhiều lần, thông tin về việc mã nào được dùng làm đầu vào cho chuyển ngữ nào sẽ được bảo tồn và duy trì.

LƯU Ý: Các chuyển ngữ gần như là các đồng nghĩa của một khái niệm có thực. Mọi chuyển ngữ trong tập hợp được cho là thể hiện cùng một ý nghĩa "bằng các ngôn từ khác nhau." Tuy nhiên, khó có thể tồn tại một đồng nghĩa tuyệt đối giữa hai hệ thống mã hóa khác nhau về cấu trúc. Vì vậy, không phải tất cả các chuyển ngữ đều có độ chính xác như nhau

2.11.8 Từ hạn định: LIST<CR>, (cố định)

```
invariant(CE x) {
  x.qualifier.notApplicable;
};
```

2.12 Chuỗi ký tự có mã (SC) chuyên biệt hóa ST

Định nghĩa: Một chuỗi ký tự có thể có một mã đi kèm. Phải luôn có văn bản nếu có mã. Mã thường là một mã địa phương.

Bảng 16. Tóm tắt đặc tính của Chuỗi ký tự có mã

Tên	Kiểu	Mô tả
code (mã)	CE	Một mã biểu diễn dữ liệu chuỗi ký tự. Ví dụ, dữ liệu chuỗi ký tự có thể là bản tin-người sử dụng lấy từ một danh mục bản tin trong đó mã biểu diễn định danh của bản tin trong danh mục bản tin.

```
type CharacterStringWithCode alias SC specializes ST {
  CE code;
};
```

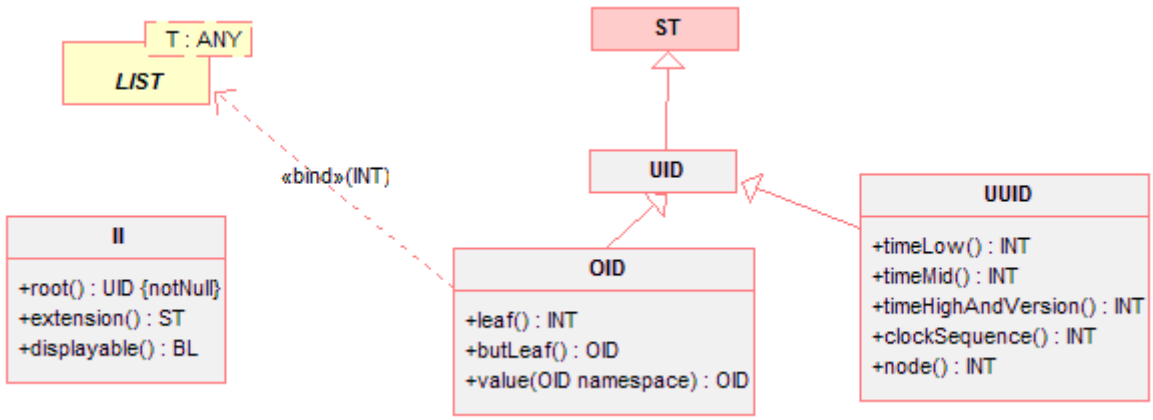
Kiểu dữ liệu *SC* được sử dụng trong các trường hợp khi mã hóa là ngoại lệ (ví dụ, các bản tin văn bản người sử dụng về bản chất là các bản tin văn bản, và một bản tin có thể in được có nội dung quan trọng. Tuy nhiên, đôi khi các bản tin đến từ một danh mục các bản tin có sẵn mẫu mà kiểu dữ liệu *SC* cho phép tham chiếu.

Bất kỳ giá trị *SC* non- NULL nào đều CÓ THỂ có một mã, tuy nhiên, KHÔNG ĐƯỢC cho mã mà không có văn bản.

```
invariant(SC x)
    where x.nonNull {
  x.code.nonNull.implies(x.notEmpty);
};
```

2.12.1 Mã: CE

Định nghĩa: Một mã biểu diễn dữ liệu chuỗi ký tự. Ví dụ, dữ liệu chuỗi ký tự có thể là bản tin-người sử dụng lấy từ một danh mục bản tin trong đó mã biểu diễn định danh của bản tin trong danh mục bản tin.



Các kiểu dữ liệu Định danh thực thể.

2.13 Chuỗi định danh duy nhất (UID) chuyên biệt hóa ST

Định nghĩa: Chuỗi định danh duy nhất là một chuỗi ký tự định danh một đối tượng một cách duy nhất trên toàn cầu và không kể thời gian. Các định dạng, giá trị và thủ tục được cho phép của kiểu dữ liệu này được HL7 kiểm soát chặt chẽ. Tại thời điểm hiện tại, các định danh do người dùng gán có thể là dạng biểu diễn ký tự nhất định của các Định danh đối tượng theo ISO (OID) và Định danh duy nhất toàn cầu DCE (UUID). HL7 cũng giữ quyền gán các dạng UID khác (RUID), như các định danh dễ nhớ khác cho các hệ thống mã hóa.

Mục đích duy nhất của *UID* là trở thành một định danh duy nhất trên toàn cầu và bất kể thời gian. Dạng thức của *UID*, cho dù là một OID, UUID hay RUID đều không ảnh hưởng. Trong phạm vi quan tâm của HL7, việc duy nhất có thể làm với *UID* là để chỉ đến đối tượng mà nó đại diện cho. Việc so sánh các *UID* là theo dạng chữ, tức là, nếu hai UID có cùng một chuỗi ký tự, chúng được coi là chỉ đến cùng một đối tượng. Nếu hai *UID* không giống nhau về dạng chữ, chúng có thể không chỉ cùng đối tượng.

```
type UniqueIdentifierString alias UID specializes ST { };
```

Giữa hai dạng thức được cho phép khác nhau của UID không có khác biệt gì về ngữ nghĩa. Các dạng thức khác nhau không được phân biệt bởi một thành phần nào nằm trong hoặc nằm ngoài bản thân chuỗi định danh.

Mặc dù đặc tả này không nhận thấy sự khác biệt ngữ nghĩa nào giữa các dạng thức khác nhau của các dạng thức định danh duy nhất, có sự khác biệt trong cách thức xây dựng và quản lý những định danh này, đó là lí do duy nhất giải thích việc định nghĩa kiểu phụ *UID* cho từng biến thể.

2.14 Định danh đối tượng ISO (OID) chuyên biệt hóa UID

Định nghĩa: Một chuỗi ký tự duy nhất trên toàn cầu biểu diễn một Định danh đối tượng ISO (OID) theo dạng chỉ gồm các số và các dấu chấm (ví dụ, "2.16.840.1.113883.3.1"). Theo ISO, OID là đường dẫn trong một cấu trúc cây, trong đó số ở ngoài cùng bên trái biểu diễn phần gốc và số ngoài cùng bên phải biểu diễn phần lá.

Mỗi nhánh sau gốc tương ứng với một hệ thống cấp định danh. Mỗi hệ thống cấp định danh này lại có thể chỉ định một tập hợp các hệ thống cấp định danh của riêng nó làm việc dưới sự bảo trợ của nó, và cứ tiếp tục như vậy. Cuối dùng, một trong các hệ thống cấp định danh này gán một số duy nhất (như một số của hệ thống cấp định danh) tương ứng với nút lá trên cây. Nút lá có thể biểu diễn một hệ thống cấp định danh (trong trường hợp đó phần gốc *OID* định danh hệ thống), hoặc một thực thể của một đối tượng. Một hệ thống cấp định danh sở hữu một không gian tên, bao gồm cây con của nó.

OID là sơ đồ định danh được ưa chuộng hơn cho các định danh duy nhất. Nên thường xuyên sử dụng *OID* trừ khi có một trong các tiêu chí sử dụng cho các sơ đồ định danh khác áp dụng cho trường hợp đó.

ISO/IEC 8824:1990(E) khoản 28 định nghĩa định danh đối tượng là

28.9 Ngữ nghĩa của một giá trị định danh đối tượng được xác định nhờ tham chiếu tới một **cây định danh đối tượng**. Cây định danh đối tượng là một cây trong đó phần gốc tương ứng với [tiêu chuẩn ISO/IEC 8824] và các đỉnh [nghĩa là các nút] tương ứng với các hệ thống hành chính có trách nhiệm phân bổ các cung [nghĩa là các cành] từ nút đó. Mỗi cung từ cây được dán nhãn bởi một thành phần định danh đối tượng, vốn là [một số nguyên]. Mỗi đối tượng thông tin cần được xác định được phân bổ chính xác một đỉnh (thường là một lá) và sẽ không có đối tượng thông tin nào khác (thuộc cùng một kiểu hoặc khác kiểu) được phân bổ cho cùng một đỉnh đó. Vì vậy, một đối tượng thông tin được xác định duy nhất và rõ ràng bởi một chuỗi các [số nguyên] (các thành phần định danh đối tượng) mà dán nhãn cho các cung theo một đường dẫn từ gốc tới đỉnh được phân bổ cho đối tượng thông tin.

28.10 Một giá trị định danh đối tượng về mặt ngữ nghĩa là một danh sách có thứ tự các giá trị thành phần định danh đối tượng. Bắt đầu từ phần gốc của cây định danh đối tượng, mỗi giá trị thành phần định danh đối tượng xác định một cung trong cây định danh đối tượng. Giá trị thành phần định danh đối tượng cuối cùng xác định một cung dẫn tới một đỉnh mà một đối tượng thông tin đã được gán cho. Chính đối tượng thông tin này được xác định bởi giá trị định danh đối tượng. [...]

```
type ObjectIdentifier alias OID specializes UID, LIST<INT> {
        INT  leaf;
        OID  butLeaf;
        OID  value(namespace OID);
  literal ST;
};
```

Theo ISO/IEC 8824 một định danh đối tượng là một chuỗi các giá trị thành phần định danh đối tượng, vốn là các số nguyên. Các giá trị thành phần này được sắp xếp thứ tự sao cho phần gốc của cây định danh đối tượng là phần đầu của danh sách, được tiếp nối bởi tất cả các cung xuống tới lá đại diện cho đối tượng thông tin được định danh bởi OID. Thực tế rằng *OID* chuyên biệt hóa LIST<INT> biểu diễn đường dẫn các giá trị thành phần định danh đối tượng này từ gốc tới lá.

Đặc tính **lá** và "**ngoại trừ Lá**" có quan điểm trái ngược. Lá là giá trị thành phần định danh đối tượng cuối cùng trong danh sách, còn đặc tính "ngoại trừ Lá" là tất cả của OID *ngoại trừ* phần *lá*. Hiểu theo một cách nhất định, lá là giá trị định danh và tất cả của OID ngoại trừ lá chỉ không gian tên trong đó lá là duy nhất và có nghĩa.

Tuy nhiên, phần nào của OID được coi là **giá trị** và phần nào là **không gian tên** có thể được nhìn nhận khác nhau. Nhìn chung, bất kỳ chuỗi thành phần OID nào ở phía bên trái đều có thể được coi là không gian tên trong đó phần còn lại của chuỗi nằm về phía bên phải được định nghĩa là một giá trị định danh có nghĩa và duy nhất. Đặc tính giá trị với một không gian tên OID là đối số của nó biểu diễn quan điểm này.[31]

```
invariant(OID x)
    where x.nonNull {
  x.notEmpty;
  x.tail.isEmpty.implies(x.leaf.equal(x.tail));
  x.tail.notEmpty.implies(x.leaf.equal(x.tail.leaf);
  x.tail.isEmpty.implies(x.butLeaf.isNull);
  x.tail.notEmpty.implies(x.butLeaf.head.equal(x.head)
      .and(x.butLeaf.tail.equal(x.butLeaf(x.tail))));
  forall(OID v; OID n) where v.equal(x.value(n)) {
    n.isEmpty.implies(v.equal(x));
    n.notEmpty.implies(v.equal(x.value(n.tail)));
    };
};
```

2.14.1 OID do HL7 cấp phát

HL7 có thể thiết lập hệ thống đăng ký OID và gán OID trong các nhánh của nó cho người sử dụng HL7 và đại lý cung cấp theo yêu cầu của họ. HL7 cũng có thể gán OID cho các hệ thống cấp định danh công cộng trong nước Mỹ (ví dụ, cơ quan cấp bằng lái Bang của Mỹ, cơ quan An ninh xã hội Mỹ, hệ thống đăng ký ID HIPAA, v.v.) và ngoài nước Mỹ (ví dụ, các cơ quan an ninh xã hội của các nước khác, cơ quan đăng ký căn cước công dân, v.v.). Các OID mà HL7 đăng ký phải được sử dụng cho các tổ chức này, cho dù các tổ chức này có các OID khác được gán từ các nguồn khác hay không.

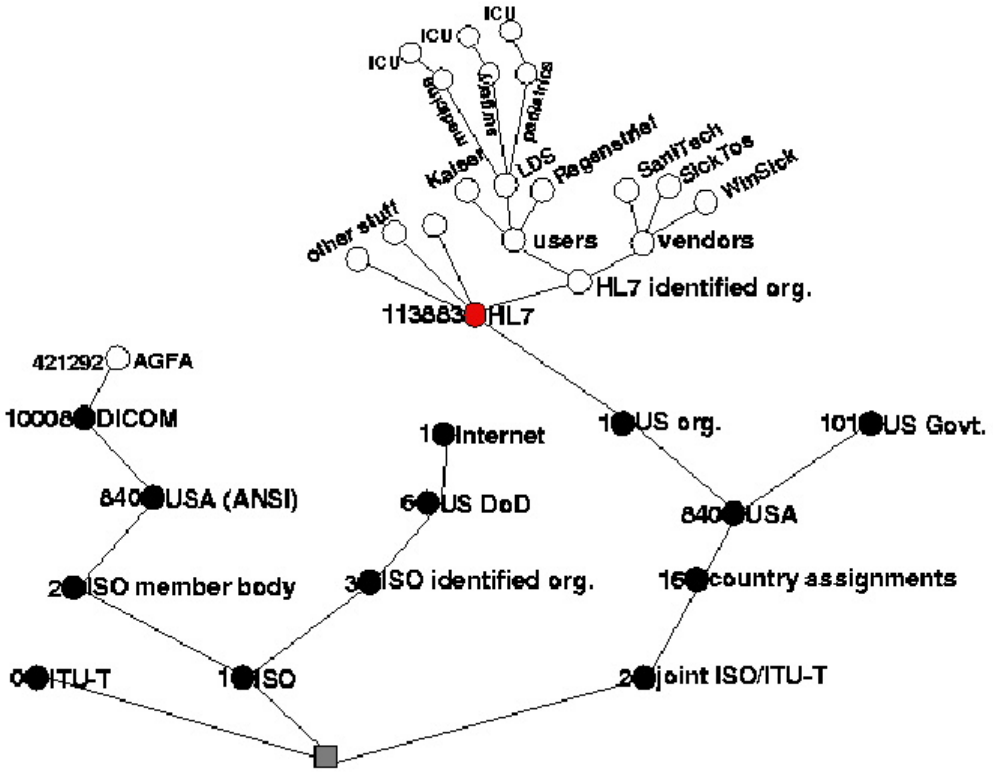
Khi gán OID cho các bên hay thực thể thứ ba, HL7 phải tìm hiểu liệu có một OID nào đã được gán cho các bản thế đó thông qua các nguồn khác hay chưa. Nếu đúng là như vậy, HL7 phải ghi OID đó vào một danh mục, nhưng HL7 không được gán một OID trùng lặp trong nhánh HL7. Nếu có thể, HL7 phải thông báo cho một bên thứ ba khi có một OID sắp được gán cho bên thứ ba đó trong nhánh HL7.

Mặc dù HL7 phải tiến hành khảo sát trước khi gán một OID trong nhánh HL7 cho bên thứ ba, với việc chưa có một cơ chế đăng ký OID toàn cầu, không thể hoàn toàn chắc chắn rằng không có một OID nào đó đã được gán từ trước cho bên thứ ba đó. Ngoài ra, việc gán trùng lặp có thể xảy ra trong tương lai thông qua một nguồn

khác. Nếu HL7 biết được những trường hợp gán trùng lặp đó, HL7 phải nỗ lực để giải quyết tình trạng này. Để đảm bảo mức độ liên kết vận hành liên tục trong khi chờ đợi, OID mà HL7 gán sẽ là OID được ưu tiên sử dụng.

Mặc dù hầu hết chủ sở hữu một OID sẽ "thiết kế" cây con không gian tên của họ theo một cách nào đó có nghĩa, không có cách nào để suy luận chung chung bất kỳ ý nghĩa nào trên các phần của một OID. HL7 không chuẩn hóa hay yêu cầu bất kỳ cấu trúc con không gian tên nào. Một chủ sở hữu OID, hay bất kỳ ai có kiến thức về cấu trúc logic của các phần của một OID, vẫn có thể sử dụng kiến thức đó để suy luận thông tin về đối tượng liên quan; tuy nhiên, không thể khái quát hóa các kỹ thuật này.

Ví dụ cho một cây định danh đối tượng ISO. OID của HL7 là 2.16.840.1.113883.



Một giao diện HL7 không được dựa vào bất kỳ kiến thức nào về cấu trúc con của một OID mà nó không thể kiểm soát các chính sách gán.

2.14.2 Dạng chữ

Định nghĩa có cấu trúc của OID được cung cấp chính xác với đặc tả OID. Trong phạm vi HL7, OID được sử dụng chỉ như các chuỗi ký tự UID, nghĩa là giá trị chuỗi dạng chữ là thứ duy nhất được thông tin và là thứ duy nhất mà một người nhận nên phải cân nhắc khi làm việc với UID trong phạm vi đặc tả HL7.

```
OID.literal ST {
  OID : INT "." OID { $.head.equal($1);
                      $.tail.equal($3); }
```

```
    | INT       { $.head.equal($1);
                $.tail.isEmpty; }
};
```

Để tương thích với tiêu chuẩn DICOM, dạng chữ của OID không nên vượt quá 64 ký tự. (xem DICOM phần 5, mục 9).

2.15 Định danh duy nhất toàn cầu DCE (UUID) chuyên biệt hóa UID

Định nghĩa: Một chuỗi ký tự duy nhất toàn cầu biểu diễn một Định danh duy nhất toàn cầu DCE (UUID) trong định dạng UUID phổ biến bao gồm 5 nhóm số thập lục phân được phân cách bởi các dấu gạch nối lần lượt có 8, 4, 4, 4 và 12 vị trí.

Cả UUID và biểu diễn chuỗi ký tự của nó đều được định nghĩa bởi Nhóm Mở, đặc tả Gọi thủ tục từ xa CDE 1.1, Phụ lục A.

UUID được gán dựa trên các địa chỉ Ethernet MAC (địa chỉ phần cứng của card mạng), thời điểm tạo ra và một số thành phần ngẫu nhiên. Sự kết hợp này được tin rằng sẽ tạo ra các định danh đủ duy nhất mà không cần bất kỳ chính sách tổ chức nào trong việc gán định danh (trên thực tế nó dựa trên việc tổ chức gán địa chỉ MAC).

UUID **không** phải là sơ đồ định danh được ưa chuộng để sử dụng như UID HL7. UUID có thể được sử dụng khi định danh được cấp cho các đối tượng biểu diễn đơn lẻ (ví dụ, định danh thực thể thực thể, định danh sự kiện hành động, v.v.). Đối với các đối tượng mô tả lớp sự vật hoặc sự kiện (ví dụ, các hạng mục của danh mục), thì OID là sơ đồ định danh được ưa chuộng hơn

```
type UniversalUniqueIdentifier alias UUID specializes UID {
  INT  timeLow;
  INT  timeMid;
  INT  timeHighAndVersion;
  INT  clockSequence;
  INT  node;
};
```

2.15.1 Dạng chữ

Định nghĩa có cấu trúc của UUID được cung cấp chính xác với đặc tả UUID. Trong phạm vi HL7, UUID được sử dụng chỉ như các chuỗi ký tự UID, nghĩa là giá trị chuỗi dạng chữ là thứ duy nhất được thông tin và là thứ duy nhất mà một người nhận nên phải cân nhắc khi làm việc với UID trong phạm vi đặc tả HL7.

Dạng chữ cho UUID được định nghĩa dựa theo đặc tả gốc của UUID. Tuy nhiên, do UID HL7 phân biệt loại chữ hoa và chữ thường, để sử dụng với HL7, các số thập lục phân A-F trong UUID phải được chuyển đổi thành chữ hoa.

```
UUID.literal ST {
```

```
UUID       : hex8 "-" hex4 "-" hex4 "-" hex4 "-" hex12 {
             $.timeLow.equal($1);
             $.timeMid.equal($3);
             $.timeHighAndVersion.equal($5);
             $.clockSequence.equal($7);
             $.node.equal($9);
             }

INT hex4    : hexDigit hexDigit hexDigit hexDigit {
             $.equal($1.times(16).plus($2)
              .times(16).plus($3)
              .times(16).plus($4);
             }

INT hex8    : hexDigit hexDigit hexDigit hexDigit
               hexDigit hexDigit hexDigit hexDigit {
             $.equal($1.times(16).plus($2)
              .times(16).plus($3)
              .times(16).plus($4)
              .times(16).plus($5)
              .times(16).plus($6)
              .times(16).plus($7)
              .times(16).plus($8);
              }

INT hex12    : hexDigit hexDigit hexDigit hexDigit
                hexDigit hexDigit hexDigit hexDigit
                hexDigit hexDigit hexDigit hexDigit {
              $.equal($1.times(16).plus($2)
               .times(16).plus($3)
               .times(16).plus($4)
               .times(16).plus($5)
               .times(16).plus($6)
               .times(16).plus($7)
               .times(16).plus($8)
               .times(16).plus($9)
               .times(16).plus($10)
```

```
                    .times(16).plus($11)
                    .times(16).plus($12);
                   }


  INT hexDigit : "0"    { $.equal(0); }
               | "1"    { $.equal(1); }
               | "2"    { $.equal(2); }
               | "3"    { $.equal(3); }
               | "4"    { $.equal(4); }
               | "5"    { $.equal(5); }
               | "6"    { $.equal(6); }
               | "7"    { $.equal(7); }
               | "8"    { $.equal(8); }
               | "9"    { $.equal(9); }
               | "A"    { $.equal(10); }
               | "B"    { $.equal(11); }
               | "C"    { $.equal(12); }
               | "D"    { $.equal(13); }
               | "E"    { $.equal(14); }
               | "F"    { $.equal(15); }
};
```

LƯU Ý: Kết quả đầu ra của các chương trình và chức năng liên quan tới UUID có thể sử dụng tất cả các loại dạng thức, chữ hoa, chữ thường, và có hay không có gạch nối để nhóm các số. Kết quả đầu ra này phải được xử lý sau để phù hợp với đặc tả HL7, nghĩa là, phải thêm dấu gạch nối để nhóm 8-4-4-4-12 và tất cả các số thập lục phân phải được chuyển đổi thành viết hoa.

2.16 Lược đồ định danh của riêng HL7 (RUID) chuyên biệt hóa UID

Định nghĩa: Một chuỗi ký tự duy nhất toàn cầu chỉ do HL7 định nghĩa. Các định danh trong sơ đồ này chỉ được định nghĩa bởi các đặc tả HL7 đã được bỏ phiếu. Các cộng đồng hay hệ thống địa phương không bao giờ được sử dụng các định danh dành riêng này dựa trên đàm phán song phương.

Các định danh của riêng HL7 là các chuỗi ký tự chỉ bao gồm các chữ cái (US-ASCII), chữ số và dấu gạch nối, trong đó ký tự đầu tiên phải là một chữ cái. HL7 có thể gán các định danh riêng này như các định danh dễ nhớ cho các khái niệm chính được quan tâm trong HL7.

2.17 Định danh thực thể (II) chuyên biệt hóa ANY

Định nghĩa: Một định danh xác định duy nhất một vật thể hoặc đối tượng. Ví dụ như định danh đối tượng cho các đối tượng của HL7 RIM, số bệnh án, ID chỉ định/yêu cầu, ID hạng mục trong danh mục dịch vụ, Số định danh phương tiện giao thông (VIN), v.v. Định danh thực thể được xác định dựa trên các định danh đối tượng của ISO.

Bảng 17. Tóm tắt đặc tính Định danh đối tượng

Tên	Kiểu	Description Mô tả
root (gốc)	UID	Một định danh duy nhất đảm bảo tính duy nhất trên toàn cầu của định danh thực thể. Riêng gốc cũng có thể là định danh thực thể đầy đủ.
extension (phần mở rộng)	ST	Một chuỗi ký tự là định danh duy nhất trong phạm vi của gốc định danh.
assigningAuthorityName (Tên Hệ thống cấp định danh)	ST	Một tên mà con người có thể đọc được hoặc tên dạng dễ nhớ của hệ thống cấp định danh. Tên Hệ thống Cấp Định danh không có giá trị tính toán. Mục đích của Tên Hệ thống Cấp Định danh là để hỗ trợ một người diễn dịch một giá trị kiểu dữ liệu II mà không được trợ giúp trong việc diễn dịch hệ thống. Lưu ý: mọi quá trình xử lý tự động hóa đều không phải phụ thuộc vào sự có mặt của tên hệ thống định danh dưới bất kỳ dạng nào.
displayable (hiển thị được)	BL	Chỉ báo liệu định danh có được dự định hiển thị cho người xem và để nhập dữ liệu hay không (hiển thị được = đúng), trái với trường hợp chỉ để liên kết hoạt động của máy (hiển thị được = sai).

```
type InstanceIdentifier alias II specializes ANY {
  ST   extension;
  UID  root;
  ST   assigningAuthorityName;
  BL   equal(ANY x);
};
```

2.17.1 Gốc: UID

Định nghĩa: Một định danh duy nhất đảm bảo tính duy nhất trên toàn cầu của định danh thực thể. Riêng gốc cũng có thể là định danh thực thể đầy đủ.

Khi có một phần mở rộng non-null, gốc thường được diễn dịch thành "hệ thống cấp định danh", nghĩa là, cho rằng gốc bằng cách nào đó liên quan tới một tổ chức cấp các định danh được gửi đi trong phần mở rộng. Tuy nhiên, gốc không cần phải là một UID tổ chức, nó cũng có thể là một UID được đăng ký riêng cho một sơ đồ định danh.[32]

```
invariant(II x)
    where x.nonNull {
  root.nonNull;
};
```

2.17.2 Phần mở rộng: ST

Định nghĩa: Một chuỗi ký tự là định danh duy nhất trong phạm vi của gốc định danh.

Phần mở rộng là một chuỗi ký tự duy nhất trong không gian tên do gốc chỉ định. Nếu có một phần mở rộng non-NULL tồn tại, gốc sẽ chỉ ra một không gian tên (đôi khi được gọi là "hệ thống cấp định danh" hoặc "kiểu định danh".) Đặc tính phần mở rộng có thể là NULL, trong trường hợp đó gốc OID là định danh duy nhất hoàn chỉnh.

Sơ đồ phần mở rộng và gốc có ý nghĩa là sự ghép nối giữa gốc và phần mở rộng phải là một định danh duy nhất toàn cầu cho hạng mục mà giá trị kiểu dữ liệu II này định danh.

Chúng tôi đề xuất các hệ thống sử dụng sơ đồ OID cho các định danh bên ngoài của các đối tượng được truyền thông. Đặc tính phần mở rộng được cung cấp chủ yếu để cho phù hợp với các sơ đồ định danh chữ-số kế thừa.

Một vài sơ đồ định danh định nghĩa các lựa chọn kiểu nhất định cho các giá trị mã của chúng. Ví dụ, Số An sinh Xã hội của Mỹ (SSN) thường được viết với các dấu gạch ngang để nhóm các chữ số thành một kiểu "123-12-1234". Tuy nhiên, các dấu gạch ngang không có nghĩa và một SSN cũng có thể được biểu diễn thành "123121234" mà không có dấu gạch ngang.

Trong trường hợp các sơ đồ định danh hỗ trợ nhiều biểu diễn, HL7 sẽ ra quyết định về dạng được ưu tiên. HL7 phải có văn bản cho quyết định đó khi sơ đồ định danh bên ngoài tương ứng đó được công nhận. HL7 phải lựa chọn dạng được ưu tiêu dựa trên các tiêu chí thực tiễn và sử dụng phổ biến. Khi không có các tiêu chí rõ ràng về tính thực tiễn và việc sử dụng phổ biến, thì hình thức an toàn nhất, dễ mở rộng nhất, và ít cách điệu hóa nhất (ít được trang trí nhất) phải được ưu tiên.[33]

HL7 cũng có thể quyết định kết hợp các định danh bên ngoài phổ biến với phần giá trị của gốc II OID. Ví dụ, số SSN của Mỹ có thể được biểu diễn thành

2.16.840.1.113883.4.1.123121234. Các tiêu chí tính thực tiễn và sử dụng phổ biến sẽ định hướng cho quyết định của HL7 đối với từng trường hợp riêng lẻ.

2.17.3 Tên hệ thống cấp định danh: ST

Định nghĩa: Một tên mà con người có thể đọc được hoặc tên dạng dễ nhớ của hệ thống cấp định danh. Tên Hệ thống Cấp Định danh không có giá trị tính toán. Mục đích của Tên Hệ thống Cấp Định danh là để hỗ trợ một người diễn dịch một giá trị kiểu dữ liệu II mà không được trợ giúp trong việc diễn dịch hệ thống. Lưu ý: mọi quá trình xử lý tự động hóa đều không phải phụ thuộc vào sự có mặt của tên hệ thống định danh dưới bất kỳ dạng nào

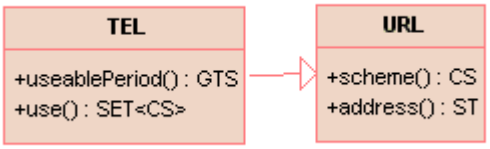
2.17.4 Có thể được hiển thị: BL

Định nghĩa: Chỉ báo liệu định danh có được dự định hiển thị cho người xem và để nhập dữ liệu hay không (hiển thị được = đúng), trái với trường hợp chỉ để liên kết hoạt động của máy (hiển thị được = sai).

2.17.5 Tương đương BL, thừa kế từ ANY

Hai định danh thực thể là bình đẳng (tương đương) khi và chỉ khi các đặc tính gốc và phần mở rộng của chúng là bình đẳng.

```
invariant(II x, y)
    where x.nonNull.and(y.nonNull) {
  x.equal(y).equal(x.root.equal(y.root)
        .and(x.extension.equal(y.extension)));
};
```



Các kiểu dữ liệu URL và TEL

2.18 Định vị tài nguyên toàn cầu (URL) chuyên biệt hóa ANY

Định nghĩa: Một địa chỉ viễn thông được xác định dựa theo tiêu chuẩn Internet RFC 2396 [http://www.ietf.org/rfc/rfc2396.txt]. URI xác định giao thức và điểm liên lạc được định nghĩa bởi giao thức đó cho tài nguyên. Các cách sử dụng đáng lưu ý của kiểu dữ liệu địa chỉ viễn thông là cho số điện thoại, số fax, địa chỉ email, tham chiếu siêu văn bản, tham chiếu FTP, v.v.

Tiêu chuẩn Internet RFC 2396 [http://www.ietf.org/rfc/rfc2396.txt] định nghĩa một URI như sau:

Cũng như việc có nhiều phương pháp khác nhau để truy cập tài nguyên, có một vài sơ đồ để mô tả vị trí của các tài nguyên đó. Cú pháp chung cho URL cung cấp một khuôn khổ để thiết lập các sơ đồ mới sử dụng những giao thức khác với những giao thức được định nghĩa trong tài liệu này.

URL được dùng để "định vị" tài nguyên, bằng cách cung cấp một định danh trừu tượng của vị trí tài nguyên. Sau khi đã định vị được tài nguyên, một hệ thống có thể thực hiện một loạt thao tác trên tài nguyên đó, như một loạt thao tác đặc trưng được thể hiện bởi các từ như "truy cập", "cập nhật", "thay thế", "tìm kiếm thuộc tính". Nhìn chung, chỉ có phương pháp "truy cập" cần được xác định cho sơ đồ URL bất kỳ.

Theo thỏa thuận, có thể cho phép sử dụng một URI thay cho một URL. Trong các trường hợp này, người ta vẫn kỳ vọng rằng có thể truy cập các tài nguyên được định danh bởi một phương pháp nào đó đã được nhất trí. Một cách sử dụng phổ biến của URI là tham khảo các đính kèm SOAP.

```
protected type UniversalResourceLocator alias URL specializes ANY {
        CS  scheme;
        ST  address;
  literal ST;
};
```

2.18.1 Lược đồ: CS

Định nghĩa: Xác định giao thức được sử dụng để diễn dịch chuỗi ký tự địa chỉ và để truy cập tài nguyên có địa chỉ đó.

Một vài sơ đồ URL được đăng ký bởi *Tổ chức cấp phát số hiệu Internet* (IANA) [http://www.iana.org], tuy nhiên IANA chỉ đăng ký các sơ đồ URL được định nghĩa trong các tài liệu Internet RFC. Trên thực tế, có một số sơ đồ URL được định nghĩa bên ngoài các tài liệu RFC, một phần trong số đó được đăng ký với tổ chức World Wide Web Consortium (W3C).[34]

Tương tự như đối với ED.mediaType, HL7 cũng đưa ra các gợi ý về giá trị *sơ đồ* để phân loại chúng thành *bắt buộc, khuyến nghị, khác*, và *không tán thành*. Mọi sơ đồ không được đề cập ở đây đều có trạng thái là *khác*.

Bảng 18. : Miền Lược đồ URL

mã	tên	định nghĩa
fax	Fax	Một số điện thoại dùng thiết bị fax [http://www.ietf.org/rfc/rfc3966.txt và http://www.ietf.org/rfc/rfc2806.txt].

mã	tên	định nghĩa
file	File (Tập tin)	Tên tệp địa phương của riêng máy chủ [RCF 1738]. Cần lưu ý rằng sơ đồ tệp chỉ có tác dụng cho các tệp địa phương. Ít có trường hợp sử dụng để trao đổi các tên tệp địa phương giữa các hệ thống, vì hệ thống nhận có khả năng không thể truy cập tệp.
ftp	FTP	Giao thức truyền tải tập tin (FTP) [http://www.ietf.org/rfc/rfc1738.txt].
http	HTTP	Giao thức truyền tải siêu văn bản [http://www.ietf.org/rfc/rfc2368.txt].
mailto	Mailto (Gửi đến)	Địa chỉ thư điện tử [http://www.ietf.org/rfc/rfc2368.txt].
mllp	MLLP	Giao thức tầng thấp tối thiểu truyền thống của HL7. URL có dạng một IP URL phổ biến, ví dụ mllp://<host>:<port>/ trong đó <host> là địa chỉ IP hoặc tên máy chủ DNS và <port> là số hiệu cổng trên đó giao thức MLLP được phục vụ.
modem	Modem	Một số điện thoại dùng thiết bị modem [http://www.ietf.org/rfc/rfc3966.txt and http://www.ietf.org/rfc/rfc2806.txt].
nfs	NFS	Giao thức hệ thống tập tin mạng [http://www.ietf.org/rfc/rfc2224.txt]. Một số địa điểm sử dụng máy chủ NFS để chia sẻ các tập tin dữ liệu.
tel	Telephone (Điện thoại)	Số điện thoại dùng lời nói [http://www.ietf.org/rfc/rfc3966.txt và http://www.ietf.org/rfc/rfc2806.txt].
telnet	Telnet	Tham chiếu đến các phiên tương tác [http://www.ietf.org/rfc/rfc1738.txt]. Một số địa điểm triển khai (ví dụ, các phòng thí nghiệm) có phiên truy vấn từ xa dựa trên TTY có thể truy cập thông qua telnet.

Lưu ý rằng đặc tả này rõ ràng chỉ giới hạn ở các URL. Tên tài nguyên toàn cầu (URN) không được đặc tả này đề cập đến. URN là một kiểu sơ đồ định danh cho các nguồn tài nguyên không truy cập được. Tuy nhiên, đặc tả này chỉ quan tâm tới các tài nguyên truy cập được, là những loại thuộc nhóm URL.

2.18.2 Địa chỉ: ST

Định nghĩa: Địa chỉ là một chuỗi ký tự với định dạng hoàn toàn do sơ đồ định nghĩa.

2.18.3 Dạng chữ

Mặc dù về mặt khái niệm URL có các đặc tính sơ đồ và địa chỉ, nhưng hình thức phổ biến của một URL là một chuỗi ký tự dạng chữ được hình thành dựa theo tiêu chuẩn Internet. Cú pháp chung của dạng chữ URL là:

```
URL.literal ST {
  URL : /[a-z0-9+.-]+/ ":" ST   { $.scheme.equal($1);
                                  $.address.equal($3); }
};
```

Số điện thoại và số FAX

Lưu ý rằng không có kiểu dữ liệu đặc biệt cho số điện thoại, số điện thoại là TEL và được xác định như các *URL*.

Số điện thoại *URL* được định nghĩa trong Internet RFC 2806 [http://www.ietf.org/rfc/rfc2806.txt]. Định nghĩa của nó được tóm tắt trong mục con này. Phần tóm tắt này không ghi đè hay thay đổi bất kỳ quy định nào của đặc tả Internet.

Các *URL* cho điện thoại dùng lời nói bắt đầu với "tel:" và URL cho fax bắt đầu với "fax:"

địa chỉ là số điện thoại tuân thủ quy định ITU-T E.123 *Mạng điện thoại và hoạt động ISDN, đánh số, định tuyến và dịch vụ di động: Ký hiệu cho số điện thoại quốc gia và quốc tế (1993)*. Mặc dù HL7 không bổ sung hay rút bớt gì từ đặc tả URL, tập hợp con cú pháp địa chỉ ưa chuộng được cung cấp dưới đây:

```
proctected type TelephoneURL specializes URL {
  literal ST {
    URL              : /(tel)|(fax)/ ":" address   { $.scheme.equal($1);
                                                     $.address.equal($3); };
    ST address       : "+" phoneDigits
    ST phoneDigits   : digitOrSeparator phoneDigits
                     | digitOrSeparator
```

```
        ST digitOrSeparator : digit
                            | separator;
        ST digit            : /[0..9]/;
        ST separator        : /[().-]/;
        };
    };
```

Số điện thoại tuyệt đối toàn cầu bắt đầu bằng dấu "+" và mã nước được ưa chuộng. Các ký tự ngăn cách có ý nghĩa trang trí nhưng không có giá trị đối với ý nghĩa của số điện thoại. Ví dụ: "tel:+13176307960" và "tel:+1(317)630-7960" là cùng một số điện thoại; "fax:+49308101724" và "fax:+49(30)8101-724" là cùng một số fax.

2.19 Địa chỉ viễn thông (TEL) chuyên biệt hóa URL

Định nghĩa: Một số điện thoại (giọng nói hoặc fax), địa chỉ email, hoặc một địa chỉ định vị khác cho một tài nguyên có thể liên lạc bởi một thiết bị viễn thông. Địa chỉ được xác định như một Định vị tài nguyên toàn cầu (URL) được hạn định bởi đặc tả thời gian và các mã sử dụng để giúp quyết định sử dụng địa chỉ nào tại một thời điểm và cho một mục đích cho trước

Bảng 19. Tóm tắt đặc tính Địa chỉ viễn thông

Tên	Kiểu	Mô tả
useablePeriod (Giai đoạn có thể sử dụng)	GTS	Xác định các khoảng thời gian có thể sử dụng địa chỉ viễn thông. Đối với một số điện thoại, nó có thể chỉ ra thời gian trong ngày mà có thể liên lạc với đối tượng qua điện thoại. Đối với một địa chỉ web, nó có thể chỉ ra khoảng thời gian mà nội dung web được cam kết sẽ hiển thị theo địa chỉ đã cho.
use (sử dụng)	SET<CS>	Một hoặc nhiều mã thông báo cho một hệ thống hoặc người sử dụng nên lựa chọn địa chỉ viễn thông nào trong một tập hợp các địa chỉ tương tự nhau để đáp ứng một nhu cầu viễn thông cụ thể.

Ngữ nghĩa của một địa chỉ viễn thông là một thực thể giao tiếp (bộ trả lời) lắng nghe và hồi đáp với địa chỉ đó, và do đó có thể được liên hệ bởi một thực thể giao tiếp khác (bộ khởi tạo).

Bộ trả lời của một địa chỉ viễn thông có thể là một dịch vụ tự động có khả năng hồi đáp với thông tin (ví dụ, các dịch vụ FTP hay HTTP). Trong trường hợp này, một địa chỉ viễn thông là một tham chiếu tới thông tin có thể truy cập được qua địa chỉ đó. Một giá trị địa chỉ viễn thông vì vậy có thể được chuyển tới một thông tin nào đó (dưới hình thức dữ liệu được đóng gói, kiểu dữ liệu ED.)

```
type TelecommunicationAddress alias TEL specializes URL {
  GTS     useablePeriod;
  SET<CS>  use;
  BL      equal(ANY x);
};
```

Địa chỉ viễn thông là một phần mở rộng của Định vị tài nguyên toàn cầu (URL) được xác định dựa theo tiêu chuẩn Internet RFC 2396 [http://www.ietf.org/rfc/rfc2396.txt]. URL xác định giao thức và điểm liên lạc được định nghĩa bởi giao thức đó cho tài nguyên. Các trường hợp sử dụng đáng lưu ý của kiểu dữ liệu địa chỉ viễn thông là cho số điện thoại, số fax, địa chỉ e-mail, tham chiếu siêu văn bản, tham chiếu FTP, v.v.

2.19.1 Giai đoạn có thể sử dụng: GTS

Định nghĩa: Xác định các khoảng thời gian có thể sử dụng địa chỉ viễn thông. Đối với một số điện thoại, nó có thể chỉ ra thời gian trong ngày mà có thể liên lạc với đối tượng qua điện thoại. Đối với một địa chỉ web, nó có thể chỉ ra khoảng thời gian mà nội dung web được cam kết sẽ hiển thị theo địa chỉ đã cho.

2.19.2 Mã sử dụng: SET<CS>

Định nghĩa: Một hoặc nhiều mã thông báo cho một hệ thống hoặc người sử dụng nên lựa chọn địa chỉ viễn thông nào trong một tập hợp các địa chỉ tương tự nhau để đáp ứng một nhu cầu viễn thông cụ thể.

Bảng 20. Miền Sử dụng địa chỉ viễn thông

mã	tên	định nghĩa
H	home address (địa chỉ nhà)	Một địa chỉ liên lạc tại nhà, nỗ lực liên lạc vì các mục đích công việc có thể xâm phạm tính riêng tư và có nguy cơ gặp gia đình hoặc các thành viên khác trong nhà thay vì gặp người muốn gọi. Thường dùng trong các trường hợp khẩn cấp, hoặc khi không có địa chỉ liên lạc khác.
HP	primary home (nhà chính)	Địa chỉ nhà chính, để liên hệ với một người sau giờ làm việc
HV	vacation home (nhà cho dịp nghỉ lễ)	Một địa chỉ nhà trong dịp nghỉ lễ, để liên hệ với một người đang đi nghỉ phép hoặc nghỉ lễ.
WP	work place	Địa chỉ văn phòng. Lựa chọn đầu tiên để liên lạc vì mục đích công việc trong giờ làm việc

mã	tên	định nghĩa
	(nơi làm việc)	
DIR	Direct (Trực tiếp)	Cho biết địa chỉ nơi làm việc hoặc địa chỉ viễn thông có thể giúp tiếp cận được cá nhân hoặc tổ chức một cách trực tiếp không qua trung gian. Đối với số điện thoại, thường được gọi là "đường dây riêng".
PUB	Public (Công cộng)	Cho biết địa chỉ nơi làm việc hoặc địa chỉ viễn thông mà là địa chỉ "tiêu chuẩn", giúp tiếp cận một dịch vụ lễ tân, phòng văn thư, hoặc một trung gian khác trước khi gặp được đối tượng đích.
BAD	bad address (địa chỉ sai)	Cờ hiệu báo rằng địa chỉ sai, và thực tế không dùng được.
TMP	temporary address (địa chỉ tạm thời)	Địa chỉ tạm thời, có thể dùng để đến gặp hoặc gửi thư. Lưu ý rằng lịch sử địa chỉ có thể cung cấp nhiều thông tin chi tiết hơn.
AS	answering service (dịch vụ trả lời)	Máy trả lời tự động dùng cho những trường hợp ít khẩn cấp hơn và nếu mục đích liên lạc chính là để lại tin nhắn hoặc truy cập một thông báo tự động.
EC	emergency contact (địa chỉ liên lạc khẩn cấp)	Địa chỉ liên lạc dành riêng để dùng cho các trường hợp khẩn cấp. Đây là lựa chọn đầu tiên trong các trường hợp khẩn cấp, độc lập với các mã sử dụng khác.
MC	mobile contact (thiết bị liên lạc di động)	Thiết bị viễn thông di chuyển cùng và luôn ở bên cạnh chủ sở hữu. Có thể có các đặc điểm của toàn bộ các mã sử dụng khác, phù hợp cho các vấn đề khẩn cấp, không phải là lựa chọn đầu tiên cho công việc thường ngày
PG	Pager (Máy nhắn tin)	Thiết bị nhắn tin phù hợp để đề nghị gọi lại hoặc để lại một tin nhắn rất ngắn

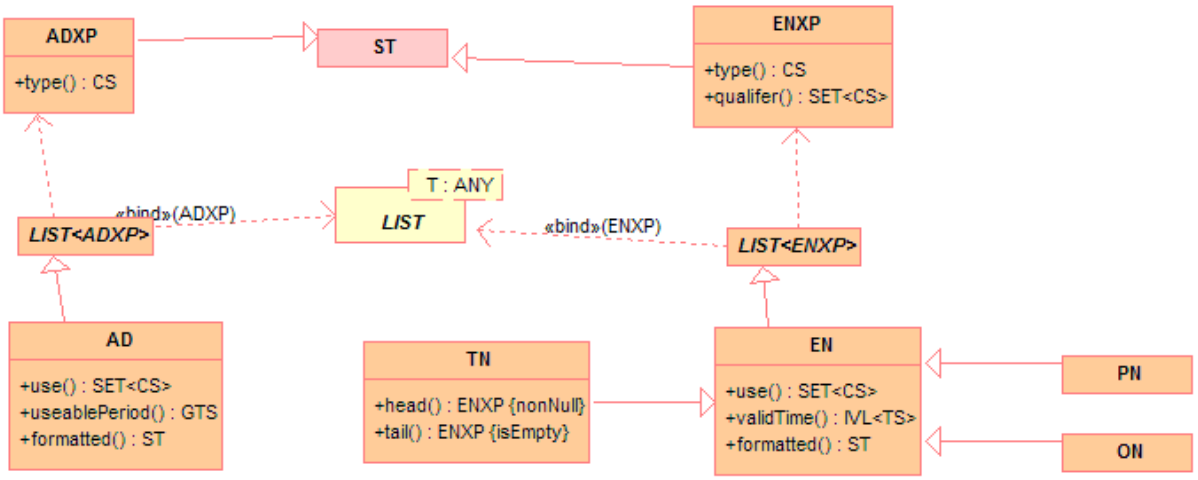
Mã sử dụng viễn thông không phải là cách phân loại hoàn chỉnh cho các loại thiết bị hoặc địa điểm liên hệ. Mục đích chính của nó là để gợi ý hoặc ngăn cản việc sử dụng một địa chỉ viễn thông cụ thể. Khó có thể định nghĩa các quy tắc cho việc lựa chọn địa chỉ viễn thông.

2.19.3 Tương đương: BL, kế thừa từ ANY

Hai giá trị địa chỉ viễn thông được coi là tương dương (hoặc bình đẳng) nếu cả hai URL của chúng là bình đẳng. Mã sử dụng và thời gian có hiệu lực bị loại trừ khỏi kiểm tra tương đương.

```
invariant(TEL x, y)
    where x.nonNull.and(y.nonNull) {
  x.equal(y).equal(((URL)x).equal((URL)y));
};
```

Các kiểu dữ liệu cho Địa chỉ bưu điện và Tên thực thể (Người, Tổ chức, và Tên thông thường) đều dựa trên phần mở rộng của một chuỗi ký tự.



2.20 Thành phần của địa chỉ (ADXP) chuyên biệt hóa ST

Định nghĩa: Một chuỗi ký tự có thể có nhãn kiểu cho biết vai trò của nó trong địa chỉ. Các thành phần tiêu biểu tồn tại trong hầu hết mọi địa chỉ là tên phố, số nhà, hoặc hộp thư, mã bưu điện, thành phố, đất nước nhưng các vai trò khác có thể được xác định theo từng vùng, từng quốc gia, hoặc ở mức độ doanh nghiệp (ví dụ trong địa chỉ quân đội). Địa chỉ thường được chia thành các dòng, được chỉ báo bởi các phần tử phân cách chia dòng đặc biệt (ví dụ, DEL).

Bảng 21. Tóm tắt đặc tính của Thành phần địa chỉ

Tên	Kiểu	Mô tả
partType (Kiểu thành phần)	CS	Xác định liệu một thành phần của địa chỉ cho biết tên phố, thành phố, đất nước, mã bưu điện, hay hòm thư, v.v. Nếu kiểu có giá trị là NULL, thì thành phần của địa chỉ không được phân loại và sẽ chỉ đơn giản giữ nguyên hình thức khi xuất hiện trên nhãn địa chỉ.

```
protected type AddressPart alias ADXP specializes ST {
  CS  type;
```

```
};
```

2.20.1 Loại thành phần của địa chỉ: CS

Định nghĩa: Xác định liệu một thành phần của địa chỉ cho biết tên phố, thành phố, đất nước, mã bưu điện, hay hòm thư, v.v. Nếu kiểu có giá trị là NULL, thì thành phần của địa chỉ không được phân loại và sẽ chỉ đơn giản giữ nguyên hình thức khi xuất hiện trên nhãn địa chỉ.

Bảng 22. Miền Kiểu thành phần của địa chỉ

mã	tên	định nghĩa
ADL	additional locator (định vị bổ sung)	Đây có thể là thành phần mô tả đơn vị, ví dụ số căn hộ, số dãy phòng, hoặc tầng. Có thể có nhiều thành phần mô tả đơn vị trong một địa chỉ (ví dụ "tầng 3, Căn hộ 342"). Đây cũng có thể là thành phần chỉ ra khỏi một địa điểm, thay vì chỉ một địa điểm nhỏ hơn trong một địa điểm lớn (ví dụ, tiếng Hà Lan "t.o" có nghĩa là 'đối diện với" để chỉ các nhà thuyền phía bên kia đường đối diện các ngôi nhà.
UNID	unit identifier (định danh đơn vị)	Số hiệu hoặc tên một đơn vị cụ thể được chứa trong một tòa nhà hoặc một khu phức hợp, được gán bởi tòa nhà hoặc khu phức hợp đó.
UNIT	unit designator (thành phần mô tả đơn vị)	Cho biết kiểu của một đơn vị cụ thể chứa trong một tòa nhà hoặc một khu phức hợp. Ví dụ: Căn hộ, Tầng
DAL	delivery address line (dòng địa chỉ giao nhận)	Dòng địa chỉ giao nhận thường được dùng thay vì phân tách phương thức giao nhận, cơ sở giao nhận, v.v. Địa chỉ thường chỉ có một dòng địa chỉ giao nhận hoặc một dòng địa chỉ đường phố, nhưng không bao gồm cả hai.
DINST	delivery installation type (kiểu cơ sở giao nhận)	Chỉ ra kiểu cơ sở giao nhận (cơ sở mà thư sẽ được giao đến trước khi vận chuyển sau cùng bằng phương thức giao nhận). Ví dụ: bưu điện, kho hãng chuyển phát thư, trung tâm bưu phẩm công cộng, trạm thu nhận, v.v.
DINSTA	delivery installation area (khu vực cơ sở	Địa điểm của cơ sở giao nhận, thường là một thị trấn hay thành phố, và chỉ cần cung cấp nếu khu vực này khác với đô thị. Là khu vực mà dịch vụ chuyển phát thư được cung cấp bởi bất kỳ

mã	tên	định nghĩa
	giao nhận)	cơ sở bưu điện hoặc dịch vụ nào như hãng chuyển phát cá nhân, tuyến nông thôn, hay tuyến bưu điện.
DINSTQ	delivery installation qualifier (từ hạn định cơ sở giao nhận)	Là số hiệu, chữ cái hay tên định danh một cơ sở giao nhận. Ví dụ, đối với Trạm A, thì từ hạn định cơ sở giao nhận sẽ là "A".
DMOD	delivery mode (phương thức giao nhận)	Cho biết kiểu dịch vụ được cung cấp, phương thức giao nhận. Ví dụ: hộp thư bưu điện, tuyến nông thôn, giao nhận thông thường, v.v.
DMODID	delivery mode identifier (định danh phương thức giao nhận)	Biểu diễn thông tin định tuyến như một số hiệu tuyến chuyển phát thư. Đây là số hiệu định danh của thành phần mô tả (số hiệu hộp thư hay số hiệu tuyến nông thôn).
SAL	street address line (dòng địa chỉ đường phố)	
BNR	building number (số hiệu tòa nhà)	Số hiệu của tòa nhà, căn nhà hoặc khu đất dọc theo phố. Cũng được gọi là "số hiệu phố chính". Nó không cho biết số hiệu của phố mà là số hiệu của tòa nhà
BNN	building number numeric (phần chữ số trong số hiệu tòa nhà)	Là phần chữ số trong số hiệu tòa nhà.
BNS	building number suffix	Bất kỳ ký tự chữ cái, phân số hoặc các văn bản khác nào có thể xuất hiện sau phần chữ số trong số hiệu tòa nhà.

mã	tên	định nghĩa
	(hậu tố của số hiệu tòa nhà)	
STR	street name (tên phố)	
STB	street name base (phần cơ sở của tên phố)	Tên cơ sở của một đường phố hoặc đường chính do chính quyền đô thị ghi nhận (không bao gồm kiểu phố và hướng)
STTYP	street type (kiểu phố)	Mô tả cho phố (ví dụ: phố, đại lộ, đường cung, v.v.)
DIR	Direction (Hướng)	Hướng (ví dụ, Bắc Nam, Tây, Đông)
CAR	care of (người chịu trách nhiệm)	Tên của người sẽ nhận tại địa điểm được xác định, và chịu trách nhiệm đảm bảo chuyển phát đến người nhận đích
CEN	census tract (dãy điều tra nhân khẩu)	Một đơn vị con về địa lý được quy định cho các mục đích nhân khẩu học.
CNT	Country (Quốc gia)	Quốc gia
CPA	county or parish (hạt hoặc giáo xứ)	Một đơn vị con của bang hay tỉnh. (49 bang của Hoa Kỳ dùng thuật ngữ “hạt”, Louisiana dùng thuật ngữ “giáo xứ.
CTY	Municipality (Đô thị)	Tên của thành phố, thị trấn, làng hoặc cộng đồng hay trung tâm giao nhận khác.

mã	tên	định nghĩa
DEL	Delimiter (Thành phần phân tách)	Các thành phần phân tách được in ra mà không tạo khoảng trắng. Nếu không có thành phần giá trị nào được cung cấp, thì thành phần phân tách sẽ xuất hiện dưới dạng dấu ngắt dòng.
POB	post box (hộp thư)	Là hộp thư có số hiệu đặt tại một trạm bưu điện.
PRE	Precinct (Phân khu)	Một khu vực con của đô thị
STA	state or province (bang hoặc tỉnh)	Là đơn vị con của một đất nước với chủ quyền hạn chế trong một nước được tổ chức kiểu liên bang.
ZIP	postal code (mã bưu điện)	Là mã bưu điện chỉ định một vùng do dịch vụ bưu điện định nghĩa.

2.21 Địa chỉ bưu điện (AD) chuyên biệt hóa LIST<ADXP>

Định nghĩa: Địa chỉ gửi thư và địa chỉ nhà hay địa chỉ văn phòng. Là một chuỗi ký tự các phần của địa chỉ, như tên phố hoặc hòm thư bưu điện, thành phố, mã bưu điện, nước, v.v.

Kiểu dữ liệu *AD* chủ yếu được dùng cho việc truyền thông dữ liệu cần có để in nhãn thư, là thông tin giúp một người trực tiếp tìm đến địa chỉ đó. Kiểu dữ liệu địa chỉ bưu điện không chứa thông tin bổ sung có thể hữu ích trong việc tìm kiếm các vị trí địa lý (ví dụ, tọa độ GPS) hoặc thực hiện các nghiên cứu dịch tễ học. Các thông tin bổ sung này được lưu trữ bởi các phần tử HL7 khác phù hợp hơn.

Bảng 23. Tóm tắt đặc tính Địa chỉ bưu điện

Tên	Kiểu	Mô tả
use (sử dụng)	SET<CS>	Một tập hợp mã thông báo cho một hệ thống hoặc người sử dụng nên lựa chọn địa chỉ nào trong một tập hợp các địa chỉ tương tự nhau để đáp ứng một nhu cầu cụ thể.

Tên	Kiểu	Mô tả
useablePeriod (Giai đoạn có thể sử dụng)	GTS	Đặc tả định thời gian tổng quát (GTS) quy định các khoảng thời gian có thể sử dụng địa chỉ. Nó được sử dụng để chỉ ra các địa chỉ khác nhau cho các khoảng thời gian khác nhau trong tuần hoặc trong năm.
isNotOrdered (Không được Sắp Xếp)	BL	Một giá trị Boolean cho biết liệu thứ tự của các thành phần của địa chỉ có được biết hay không. Mặc dù các thành phần của địa chỉ luôn là một Chuỗi thứ tự, thứ tự trình bày của chúng lại có thể được biết hoặc không. Khi vấn đề này là quan trọng, đặc tính Không được Sắp Xếp có thể được dùng để truyền tải thông tin này.
formatted (được định dạng)	ST	Một giá trị chuỗi ký tự với địa chỉ được định dạng theo dòng và có dấu cách hợp lý. Đây chỉ là một đặc tính ngữ nghĩa để định nghĩa chức năng của một số kiểu thành phần của địa chỉ. Cần nhớ rằng các đặc tính ngữ nghĩa không có tất cả các ngữ nghĩa luồng điều khiển. Phần được định dạng có thể được triển khai như một "thủ tục" sẽ "trả về" địa chỉ được định dạng, nhưng nó thường không phải là một biến để có thể gán cho nó một địa chỉ được định dạng. Tuy nhiên, HL7 không định nghĩa các ứng dụng mà chỉ định nghĩa ngữ nghĩa của các giá trị dữ liệu được trao đổi. Vì vậy, mô hình ngữ nghĩa rút ra từ các khái niệm như "thủ tục", "trả", và "gán" nhưng chỉ đề cập đến đặc tính và giá trị.

Địa chỉ được định nghĩa là văn bản với các đánh dấu logic bổ sung. Dấu có thể chia địa chỉ thành các dòng và có thể mô tả chi tiết vai trò của từng thành phần của địa chỉ nếu biết. Các thành phần của địa chỉ xuất hiện trong địa chỉ theo trật tự mà chúng sẽ được in trên nhãn bưu điện. Cách thức này giống với kiểu đánh dấu văn bản trong HTML hay XML (nhưng không bị giới hạn về mặt kỹ thuật ở các biểu diễn dạng XML).

Địa chỉ về bản chất là các chuỗi sắp xếp thứ tự các thành phần của địa chỉ, nhưng bổ sung một mã "sử dụng" và một khoảng thời gian có hiệu lực cho thông tin về việc khi nào có thể sử dụng địa chỉ cho một mục đích nhất định

```
type PostalAddress alias AD specializes LIST<ADXP> {
  SET<CS>  use;
  GTS      useablePeriod;
  BL       isNotOrdered;
```

```
        BL      equal(ANY x);
        ST      formatted;
    };
```

2.21.1 Mã sử dụng: SET<CS>

Định nghĩa: Một tập hợp mã thông báo cho một hệ thống hoặc người sử dụng nên lựa chọn địa chỉ nào trong một tập hợp các địa chỉ tương tự nhau để đáp ứng một nhu cầu cụ thể.

Bảng 24. Miền Sử dụng địa chỉ bưu điện (PostalAddressUse)

mã	tên	định nghĩa
H	home address (địa chỉ nhà)	Một địa chỉ liên lạc tại nhà, nỗ lực liên lạc vì các mục đích công việc có thể xâm phạm tính riêng tư và có nguy cơ gặp gia đình hoặc các thành viên khác trong nhà thay vì gặp người muốn gọi. Thường dùng trong các trường hợp khẩn cấp, hoặc khi không có địa chỉ liên lạc khác.
HP	primary home (nhà chính)	Địa chỉ nhà chính, để liên hệ với một người sau giờ làm việc
HV	vacation home (nhà trong dịp nghỉ lễ)	Một địa chỉ nhà trong dịp nghỉ lễ, để liên hệ với một người đang đi nghỉ lễ.
WP	work place (nơi làm việc)	Địa chỉ văn phòng. Lựa chọn đầu tiên để liên lạc vì mục đích công việc trong giờ làm việc
DIR	Direct (Trực tiếp)	Cho biết địa chỉ nơi làm việc hoặc địa chỉ viễn thông có thể giúp tiếp cận được cá nhân hoặc tổ chức một cách trực tiếp không qua trung gian. Đối với số điện thoại, thường được gọi là "đường dây riêng".
PUB	Public (Công cộng)	Cho biết địa chỉ nơi làm việc hoặc địa chỉ viễn thông mà là địa chỉ "tiêu chuẩn", giúp tiếp cận một dịch vụ lễ tân, phòng văn thư, hoặc một trung gian khác trước khi gặp được đối tượng đích.

mã	tên	định nghĩa
BAD	bad address (địa chỉ sai)	Cờ hiệu báo rằng địa chỉ sai, và thực tế không dùng được.
TMP	temporary address (địa chỉ tạm thời)	Địa chỉ tạm thời, có thể dùng để đến gặp hoặc gửi thư. Lưu ý rằng lịch sử địa chỉ có thể cung cấp nhiều thông tin chi tiết hơn.
Nhận biết cách biểu diễn tên khác nhau. Cách biểu diễn có thể ảnh hưởng đến cách sử dụng tên. (ví dụ, dùng chữ tượng hình cho liên lạc chính thức)		
ABC	Alphabetic (Chữ cái)	Phiên âm tên theo bảng chữ cái (tiếng Nhật: romaji)
IDE	Ideographic (Tượng hình)	Biểu diễn tên theo chữ tượng hình (ví dụ, chữ kanji của Nhật, ký tự tiếng Trung)
SYL	Syllabic (Âm tiết)	Phiên âm tên theo âm tiết (ví dụ, chữ kana của Nhật, chữ hangul của Hàn Quốc)
PHYS	physical visit address (địa chỉ thăm viếng trực tiếp)	Chủ yếu dùng để thăm viếng trực tiếp đến một địa chỉ.
PST	postal address (địa chỉ bưu điện)	Dùng để gửi thư.

Một địa chỉ không có mã sử dụng cụ thể có thể là địa chỉ mặc định hữu ích cho bất kỳ mục đích nào, nhưng một địa chỉ có mã sử dụng cụ thể sẽ được ưa chuộng hơn cho một mục đích tương ứng.

2.21.2 Giai đoạn có thể sử dụng được: GTS

Định nghĩa: Đặc tả quy định thời gian tổng quát (GTS) chỉ ra các khoảng thời gian có thể sử dụng địa chỉ. Nó được sử dụng để chỉ ra các địa chỉ khác nhau cho các khoảng thời gian khác nhau trong tuần hoặc trong năm

2.21.3 Không được sắp xếp thứ tự: BL

Định nghĩa: Một giá trị Boolean cho biết liệu thứ tự của các thành phần của địa chỉ có được biết hay không. Mặc dù các thành phần của địa chỉ luôn là một Chuỗi có thứ tự, thứ tự trình bày của chúng lại có thể được biết hoặc không. Khi vấn đề này là quan trọng, đặc tính Không được Sắp Xếp có thể được dùng để truyền tải thông tin này

2.21.4 Tương đương: BL

Hai giá trị địa chỉ được coi là tương đương (hoặc) bình đẳng nếu cả hai chứa các thành phần giống nhau của địa chỉ, không phụ thuộc vào thứ tự. Mã sử dụng và thời gian có hiệu lực bị loại khỏi việc kiểm tra bình đẳng.

```
invariant(AD x, y)
    where x.nonNull.and(y.nonNull) {
  x.equal(y).equal((
    forall(ADXP p) where x.contains(p) {
      y.contains(p);
      }).and.(
        forall(ADXP p) where x.contains(p) {
          y.contains(p);
          }));
};
```

2.21.5 Địa chỉ được định dạng: ST

Định nghĩa: Một giá trị chuỗi ký tự với địa chỉ được định dạng theo dòng và có dấu cách khoảng trắng hợp lý. Đây chỉ là một đặc tính ngữ nghĩa để định nghĩa chức năng của một số kiểu thành phần của địa chỉ.[36]

Mục đích chính của kiểu dữ liệu *AD* là để nắm bắt các địa chỉ bưu điện, để một người có thể viếng thăm địa chỉ đó hoặc gửi thư tới đó. Con người sẽ nhìn vào địa chỉ ở dạng đã được in ra, ví dụ như trên một nhãn bưu điện. Kiểu dữ liệu *AD* định nghĩa các quy tắc chính xác về cách thức định dạng dữ liệu của nó.[37]

Địa chỉ là danh sách có thứ tự các thành phần của địa chỉ. Mỗi thành phần của địa chỉ được in theo thứ tự của danh sách từ trái sang phải và từ trên xuống dưới (hoặc theo chiều đọc của bất kỳ ngôn ngữ cụ thể nào khác, điều này nằm ngoài phạm vi của đặc tả này). Mọi giá trị thành phần của địa chỉ đều được in. Hầu hết các thành phần của địa chỉ đều được đóng khung bởi các khoảng trắng. Sáu quy tắc dưới đây điều chỉnh việc thiết lập các khoảng trắng.

1. Khoảng trắng không bao giờ tích lũy, nghĩa là hai khoảng trắng liên tiếp tương đương một khoảng trắng. Nhiều dấu ngắt dòng liên tiếp có thể được

rút về thành một. Khoảng trắng xung quanh một dấu ngắt dòng thì không có ý nghĩa.

2. Dạng chữ có thể chứa các khoảng trắng rõ ràng, và cũng áp dụng các quy tắc rút gọn khoảng trắng tương tự. Không có khái niệm về một ngắt dòng dạng chữ bên trong văn bản của chỉ một thành phần của địa chỉ.
3. Các khoảng trắng hiện phía trước và sau là không có ý nghĩa trong tất cả các thành phần của địa chỉ, ngoại trừ đối với các thành phần phân tách (DEL) của địa chỉ.
4. Theo mặc định, một thành phần của địa chỉ được bao quanh bởi khoảng trắng ẩn.
5. Các thành phần phân tách (DEL) của địa chỉ không được bao quanh bởi bất kỳ khoảng trắng ẩn nào.
6. Các khoảng trắng hiện phía trước và sau là quan trọng trong các thành phần của địa chỉ thành phần phân tách (DEL).

Điều này có nghĩa là tất cả các thành phần của địa chỉ thường được bao quanh bởi các khoảng trắng, nhưng khoảng trắng không bao giờ tích lũy. Thành phần phân tách không bao giờ được bao quanh bởi các khoảng trắng ẩn và mọi khoảng trắng của các thành phần địa chỉ trước hoặc sau đó đều bị loại bỏ, dù nó là ẩn hay hiện.

Dưới đây là các ví dụ về địa chỉ dưới dạng ITS XML.

1050 Tây đại lộ Wishard,

RG tầng 5,

Indianapolis, bang Indiana 46240.

Có thể được mã hóa bằng bất kỳ dạng nào dưới đây:[38]

Dạng đầu tiên có được từ một hệ thống chỉ lưu các địa chỉ là văn bản tự do hoặc trong một danh sách các trường dữ liệu dòng1, dòng2, v.v.:

Ví dụ 3:

```
<addr use="WP">
  1050 W Wishard Blvd,
  RG 5th floor,
  Indianapolis, IN 46240
</addr>
```

Dạng thứ hai cụ thể hơn dạng một về vai trò của các thành phần của địa chỉ:

Ví dụ 4:

```
<addr use="WP">
  <streetAddressLine>1050 W Wishard Blvd</streetAddressLine>,
  <streetAddressLine>RG 5th floor</streetAddressLine>,
  <city>Indianapolis</city>,
  <state>IN</state>
  <postalCode>46240</postalCode>
</addr>
```

Đây là dạng điển hình thường thấy ở Mỹ, trong đó địa chỉ phố đôi khi bị phân tách, còn thành phố, bang và mã ZIP luôn luôn bị phân tách.

Dạng thứ ba thậm chí còn cụ thể hơn:

Ví dụ 5:

```
<addr use="WP">
  <houseNumber>1050</houseNumber>
  <direction>W</direction>
  <streetName>Wishard Blvd</streetName>,
  <additionalLocator>RG 5th floor</additionalLocator>,
  <city>Indianapolis</city>,
  <state>IN</state>
  <postalCode>46240</postalCode>
</addr>
```

Dạng sau cùng trên đây không được sử dụng ở Mỹ. Tuy nhiên, nó rất hữu dụng tại Đức, nơi nhiều hệ thống coi số nhà là một trường riêng biệt. Ví dụ, địa chỉ ở Đức:

Windsteiner Weg 54a,

D-14165 Berlin

nhiều khả năng sẽ được mã hóa như sau[39]

Ví dụ 6:

```
<addr use="HP">
  <streetName>Windsteiner Weg</streetName>
  <houseNumber>54a</houseNumber>,
  <country>D</country>-
  <postalCode>14165</postalCode>
  <city>Berlin</city>
</addr>
```

2.22 Thành phần của tên Thực thể (ENXP) chuyên biệt hóa ST

Định nghĩa: Một thẻ chuỗi ký tự biểu diễn một phần của tên. Có thể có một mã kiểu biểu thị vai trò của thành phần trong toàn bộ tên thực thể, và một mã từ hạn định để cung cấp thêm chi tiết về kiểu thành phần của tên. Các thành phần tiêu biểu của tên cho tên người là tên, họ, danh xưng, v.v.

Bảng 25. Tóm tắt đặc tính của phần của tên thực thể

Tên	Kiểu	Mô tả
partType (Kiểu thành phần)	CS	Cho biết liệu thành phần của tên là tên, họ, tiền tố, hay hậu tố, v.v.
qualifier (từ hạn định)	SET<CS>	Từ hạn định là một tập hợp các mã với mỗi mã xác định một loại con nhất định thành phần của tên bên cạnh kiểu thành phần của tên chính. Ví dụ, tên có thể được đánh dấu là tên thân mật, họ có thể là biệt hiệu hoặc tên trong hồ sơ quốc gia.

```
protected type EntityNamePart alias ENXP specializes ST {
  CS      type;
  SET<CS>  qualifier;
};
```

2.22.1 Loại thành phần của Tên: CS

Định nghĩa: Cho biết liệu thành phần của tên là tên, họ, tiền tố, hay hậu tố, v.v.

Bảng 26. Miền Kiểu thành phần của tên thực thể (EntityNamePartType)

mã	tên	định nghĩa
FAM	family (họ)	Họ, đây là tên liên kết với gia phả. Trong một số nền văn hóa (ví dụ Eritrea), họ của con trai là tên gọi của cha.
GIV	given (tên)	Tên (không gọi là "tên đầu tiên" vì tên này không phải luôn được đặt ở đầu).
PFX	prefix (tiền tố)	Tiền tố có liên hệ mật thiết với thành phần của tên ngay sau nó. Tiền tố không có khoảng trắng ẩn theo sau (mặc dù có khoảng trắng ẩn ngay trước). Cần lưu ý rằng có thể đảo ngược các tiền tố.
SFX	suffix (hậu tố)	Hậu tố có liên hệ mật thiết thành với phần của tên ngay trước nó. Hậu tố không có khoảng trắng ẩn ngay trước (mặc dù có

mã	tên	định nghĩa
		khoảng trắng ẩn ngay sau). Không thể đảo ngược các hậu tố.
DEL	delimiter (thành phần phân tách)	Thành phần phân tách không có ý nghĩa gì khác ngoài việc được in ra ở dạng chữ trong biểu diễn tên. Thành phần phân tách không có khoảng trắng ẩn ngay trước và ngay sau.

Không phải mọi thành phần của tên đều phải có một mã kiểu, nếu mã kiểu chưa được biết, hoặc không áp dụng được, hoặc đơn giản là chưa được định nghĩa, thì mã kiểu được biểu diễn bằng giá trị NULL (type.isNull). Ví dụ, với tên "Rogan Sulma" có thể chưa rõ đâu là tên, đâu là họ, hoặc Rogan có thể chỉ là danh xưng.

Tên thực thể được khái niệm hóa là văn bản có bổ sung các đánh dấu (mark-up). Phần đánh dấu có thể mô tả chi tiết vai trò của từng thành phần của tên nếu biết. Các thành phần của tên được sắp xếp theo trật tự được in trên nhãn bưu điện. Mô hình này giống với kiểu đánh dấu văn bản của HTML hoặc XML.

2.22.2 Từ hạn định: SET<CS>

Định nghĩa: Từ hạn định là một tập hợp các mã với mỗi mã xác định một loại con nhất định thành phần của tên bên cạnh kiểu thành phần của tên chính. Ví dụ, tên có thể được đánh dấu là tên thân mật, họ có thể là biệt hiệu hoặc tên trong hồ sơ quốc gia.

Bảng 27. Miền Từ hạn định thành phần của tên thực thể

mã	tên	định nghĩa
LS	Legal status (Tình trạng pháp lý)	Dùng cho các tổ chức, là một hậu tố cho biết trạng thái pháp lý, ví dụ ""Inc.", "Co.", "AG", "GmbH", "B.V." "S.A.", "Ltd.", v.v.
AC	academic (học thuật)	Cho biết rằng một tiền tố như "Dr." (Bác sĩ), hoặc một hậu tố như "M.D" (Bác sĩ y khoa) hoặc "Ph.D" (Tiến sĩ) là một danh xưng trong lĩnh vực học thuật.
NB	nobility (quý tộc)	Ở châu Âu và châu Á, vẫn có những người có danh xưng quý tộc (tầng lớp quý tộc). Từ "von" trong tiếng Đức thường là một danh xưng quý tộc, chứ không chỉ là một tiếp đầu ngữ (voorvoegsel) đơn giản. Các danh xưng khác là "Bá tước vùng…", hay "Đức vua..", v.v. Ngày nay rất hiếm khi được dùng, nhưng một số hệ thống vẫn ghi lại danh

mã	tên	định nghĩa
		xưng này.
PR	professional (nghề nghiệp)	Chủ yếu trong văn hóa Vương quốc Anh, mọi người vẫn có xu hướng giữ tên gọi tắt của tổ chức nghề nghiệp của họ như một phần của hậu tố của tên.
VV	voorvoegsel	Một "voorvoegsel" trong tiếng Hà Lan là một yếu tố kiểu "van" hay "de" có thể chỉ danh xưng quý tộc trong quá khứ nhưng giờ không còn nữa. Các tiền tố tương tự vẫn tồn tại trong các ngôn ngữ khác như tiếng Tây Ban Nha, Pháp hoặc Bồ Đào Nha.
AD	adopted (tên nhận nuôi)	Tên một người được đặt khi được nhận nuôi.
BR	birth (tên lúc sinh)	Tên một người có ngay sau khi được sinh ra. Thường dùng cho họ nhưng có thể được dùng để làm tên gọi lúc mới sinh và sau này có thể thay đổi.
SP	Spouse (tên vợ/chồng)	Tên lấy từ đối tác trong quan hệ hôn nhân (do đó có chữ "M"). Thường là họ của vợ/chồng. Cần lưu ý rằng không thể suy luận về giới tính dựa trên tên vợ/chồng.
CL	callme (tên thường gọi)	Tên thường gọi (thường là tên), là tên được đề nghị dùng khi xưng hô trực tiếp với một người.
IN	initial (tên viết tắt)	Chỉ ra rằng một thành phần của tên chỉ là chữ cái đầu của tên. Tên viết tắt không hàm ẩn một khoảng trống ngay sau vì sẽ không phù hợp với chữ không thuộc hệ Latinh. Tên viết tắt có thể gồm nhiều hơn một chữ cái, ví dụ, "Ph." có thể là viết tắt của "Philippe" hay "Th." là viết tắt của "Thomas".
TITLE	title (danh xưng)	Chỉ ra rằng tiền tố hoặc hậu tố là danh xưng áp dụng cho toàn bộ tên, chứ không chỉ thành phần của tên ngay cạnh.

2.23 Tên thực thể (EN) chuyên biệt hóa LIST<ENXP>

Định nghĩa: Tên cho một người, tổ chức, địa điểm hay vật thể. Là một chuỗi các thành phần của tên, như tên hay họ, tiền tố, hậu tố, v.v. Ví dụ cho các giá trị tên thực thể là "Jim Bob Walton, con", "Health Level Seven, Inc.", "Hồ Tahoe", v.v. Một tên thực thể có thể chỉ đơn giản là một chuỗi ký tự hoặc có thể bao gồm một vài thành phần của tên thực thể, chẳng hạn như "Jim", "Bob", "Walton", và "con", "Health Level Seven" và "Inc.", "Hồ" và "Tahoe".

Bảng 28. Tóm tắt đặc tính Tên thực thể

Tên	Kiểu	Mô tả
use (sử dụng)	SET<CS>	Một tập hợp mã thông báo cho một hệ thống hoặc người sử dụng nên lựa chọn tên nào trong một tập hợp các tên để phục vụ một mục đích cụ thể.
validTime (Thời gian có hiệu lực)	IVL<TS>	Một khoảng thời gian xác định thời gian trong đó tên đang hoặc đã được dùng cho thực thể. Điều này phù hợp với thực tế là con người thay đổi tên cho người, địa điểm hoặc sự vật.
formatted (được định dạng)	ST	Một giá trị chuỗi ký tự với tên thực thể được định dạng có dấu cách khoảng trắng hợp lý. Đây chỉ là một đặc tính ngữ nghĩa để định nghĩa chức năng của một số kiểu thành phần của tên. Cần nhớ rằng các đặc tính ngữ nghĩa không có tất cả các ngữ nghĩa luồng điều khiển. Phần được định dạng có thể được triển khai như một "thủ tục" sẽ "trả về" tên được định dạng, nhưng nó thường không phải là một biến để có thể gán cho nó một tên được định dạng. Tuy nhiên, HL7 không định nghĩa các ứng dụng mà chỉ định nghĩa ngữ nghĩa của các giá trị dữ liệu được trao đổi. Vì vậy, mô hình ngữ nghĩa rút ra từ các khái niệm như "thủ tục", "trả", và "gán" nhưng chỉ đề cập đến về đặc tính và giá trị.

Tên thực thể được khái niệm hóa là văn bản có bổ sung các đánh dấu (mark-up) logic. Các thành phần của tên xuất hiện theo trật tự tự nhiên mà chúng sẽ được hiển thị, trái ngược với trật tự do thành phần của tên quyết định. Trật tự của các thành phần của tên là một đặc điểm quan trọng giúp thay thế nhu cầu phải có một đặc tính "tên hiển thị" riêng biệt. Các ứng dụng có thể thay đổi trật tự các thành phần của tên đó để giải thích cho trật tự thông thường các thành phần của tên của người sử dụng chúng.

Cách thức này giống với kiểu đánh dấu văn bản trong HTML hay XML (nhưng không bị giới hạn về mặt kỹ thuật ở các biểu diễn định dạng XML).

Tên thực thể về bản chất là chuỗi các thành phần có thứ tự của tên thực thể, nhưng có bổ sung thêm mã "sử dụng" và khoảng thời gian hiệu lực cho thông tin về việc tên được sử dụng khi nào và làm sao để lựa chọn trong số nhiều bí danh có thể có hiệu lực vào cùng một thời điểm.

```
type EntityName alias EN specializes LIST<ENXP> {
  SET<CS>  use;
  IVL<TS>  validTime;
  BL       equal(ANY x);
  ST       formatted;
};
```

2.23.1 Mã sử dụng: SET<CS>

Định nghĩa: Một tập hợp mã thông báo cho một hệ thống hoặc người sử dụng nên lựa chọn tên nào trong một tập hợp các tên để phục vụ một mục đích cụ thể.

Bảng 29. Miền Sử dụng tên thực thể

mã	tên	định nghĩa
C	License (Giấy phép)	Tên như ghi trên giấy phép, hồ sơ, giấy chứng nhận, v.v. (chỉ dùng nếu khác tên hợp pháp).
I	Indigenous/Tribal (Bản địa/Bộ lạc)	Ví dụ, Chief Red Cloud (Tù trưởng Mây đỏ)
L	Legal (Hợp pháp)	Tên thường gọi/tên thông thường/tên thường sử dụng
P	pseudonym (biệt hiệu)	Tên tự đặt mà một người đang hoặc đã dùng.
A	Artist/Stage (Nghệ sỹ/Sân khấu)	Bao gồm bút danh của nhà văn, nghệ danh, v.v.

mã	tên	định nghĩa
R	Religious (Tôn giáo)	Ví dụ, Sơ Mary Francis, Tu sĩ John
SRCH	search (tìm kiếm)	Tên dự định dùng để tìm kiếm hoặc so khớp.
PHON	phonetic (ngữ âm)	Tên đánh vần theo ngữ âm.
SNDX	Soundex	Tên đánh vần theo thuật toán SoundEx.
ABC	Alphabetic (Tên ghi theo bảng chữ cái)	Phiên âm tên theo bảng chữ cái (tiếng Nhật: chữ romaji)
SYL	Syllabic (Tên ghi theo âm tiết)	Phiên âm tên theo âm tiết (ví dụ: chữ kana của Nhật, chữ hangul của Hàn Quốc)
IDE	Ideographic (Tên ghi theo chữ tượng hình)	Biểu diễn tên bằng chữ tượng hình (ví dụ, chữ kanji của Nhật, các ký tự tiếng Trung)

Một tên không có mã sử dụng cụ thể có thể là tên mặc định hữu ích cho bất kỳ mục đích nào, nhưng một tên có mã sử dụng cụ thể sẽ được ưa chuộng hơn cho một mục đích tương ứng.

2.23.2 Thời gian hợp lệ: IVL<TS>

Định nghĩa: Một khoảng thời gian xác định thời gian trong đó tên đang hoặc đã được dùng cho thực thể. Điều này phù hợp với thực tế là con người thay đổi tên cho người, địa điểm hoặc sự vật.

Kiểu dữ liệu *EN* tuân theo phần mở rộng kiểu dữ liệu hạng mục lịch sử (HXIT).

2.23.3 Tương đương: BL, thừa kế từ ANY

Hai giá trị tên được coi là bình đẳng nếu cả hai chứa các phần của tên giống nhau, không phụ thuộc vào trật tự. Mã sử dụng và thời gian có hiệu lực bị loại khỏi kiểm tra bình đẳng.

```
invariant(EN x, y)
    where x.nonNull.and(y.nonNull) {
  x.equal(y).equal((
    forall(ENXP p) where x.contains(p) {
      y.contains(p);
      }).and.(
        forall(ENXP p) where x.contains(p) {
          y.contains(p);
          }));
```

2.23.4 Định dạng tên thực thể: ST

Định nghĩa: Một giá trị chuỗi ký tự tên thực thể được định dạng có dấu cách hợp lý. Đây chỉ là một đặc tính ngữ nghĩa để định nghĩa chức năng của một số kiểu thành phần của tên.

Mục đích chính của kiểu dữ liệu *EN* là để nắm giữ tên của người, địa điểm, và đồ vật (thực thể), để một người có thể xưng hô hay nhắc tới các thực thể này khi nói hay viết. Con người sẽ thấy tên ở dạng bản in ấn, ví dụ như trên một nhãn bưu điện. Kiểu dữ liệu *EN* vì vậy định nghĩa các quy tắc chính xác về cách thức định dạng các dữ liệu của nó.[42]

Tên thực thể là danh sách có thứ tự các thành phần của tên thực thể. Mỗi thành phần của tên thực thể được in theo thứ tự của danh sách từ trái sang phải (hoặc theo chiều đọc của bất kỳ ngôn ngữ cụ thể nào). Mọi thành phần của tên thực thể (ngoại trừ những thành phần được đánh dấu là "vô hình") đều được in. Hầu hết các thành phần của tên thực thể đều được đóng khung bởi khoảng trắng. Sáu quy tắc dưới đây điều chỉnh việc thiết lập khoảng trắng.

1. Khoảng trắng không bao giờ tích lũy, nghĩa là hai khoảng trắng liên tiếp tương đương một khoảng trắng.
2. Dạng chữ có thể chứa các khoảng trắng rõ ràng, và cũng áp dụng các quy tắc rút gọn khoảng trắng tương tự.
3. Ngoại trừ các phần *tiền tố, hậu tố* và *thành phần phân tách* của tên, mọi thành phần của tên đều được bao quanh bởi khoảng trắng ẩn. Khoảng trắng hiện phía trước và sau không có ý nghĩa trong tất cả các thành phần của tên.

4. Các phần thành phần phân tách của tên không được bao quanh bởi bất kỳ khoảng trắng ẩn nào. Khoảng trắng hiện phía trước và sau là quan trọng trong các phần thành phần phân tách của tên.
5. Các phần tiền tố của tên chỉ có khoảng trắng ẩn phía trước nhưng không có khoảng trắng ẩn phía sau. Khoảng trắng xuất hiện phía sau là quan trọng trong các phần tiền tố của tên.
6. Các phần hậu tố của tên chỉ có khoảng trắng ẩn phía sau nhưng không có khoảng trắng ẩn phía trước. Khoảng trắng xuất hiện phía trước là quan trọng trong các phần hậu tố của tên.

Điều này có nghĩa là tất cả các phần của tên thực thể thường được bao quanh bởi các khoảng trắng, nhưng khoảng trắng không bao giờ tích lũy. Thành phần phân tách không bao giờ được bao quanh bởi các khoảng trắng ẩn, tiền tố không có khoảng trắng ẩn đứng sau và phụ tố không có khoảng trắng ẩn đứng trước. Mọi khoảng trắng của các phần của tên trước hoặc sau đứng xung quanh các phần của tên đặc biệt đó đều bị loại bỏ, dù nó là ẩn hay hiện.

2.23.5 Ví dụ

Một mã hóa rất đơn giản cho tên "Adam A. Everyman" là:

Ví dụ 7:

```
<name>
  <given>Adam</given>
  <given>A.</given>
  <family>Everyman</family>
</name>
```

Không cần đề cập đến từ hạn định đặc biệt nào nếu chúng không được biết hoặc không liên quan. Ví dụ tiếp theo cho thấy việc sử dụng rộng rãi nhiều tên, tiền tố, hậu tố, cho các bằng cấp có tính học thuật, danh xưng quý tộc, *vorvoegsels* ("van"), và định danh nghề nghiệp.

Ví dụ 8:

```
<name>
  <prefix qualifier="AC">Dr. phil. </prefix>
  <given>Regina</given>
  <given>Johanna</given>
  <given>Maria</given>
  <prefix qualifier="NB">Gr&auml;fin </prefix>
  <family qualifier="BR">Hochheim</family>-<family qualifier="SP">Weilenfels</family>
```

```
    <suffix qualifier="PR">NCFSA</suffix>
</name>
```

Ví dụ tiếp theo là một tên tổ chức, "Health Level Seven, Inc." ở dạng chuỗi đơn giản:

Ví dụ 9:

```
<name>Health Level Seven, Inc.</name>
```

và như một tên được phân tích đầy đủ

Ví dụ 10:

```
<name>Health Level Seven, <suffix qualifier="LS">Inc.</suffix></name>
```

Ví dụ dưới đây thể hiện một tên tiếng Nhật ở ba dạng: tượng hình (Kanji), âm tiết (Hiragana), và chữ cái (Romaji).

Ví dụ 11:

```
<name use="IDE">
  <family>木村</family>
  <given>通男</given>
</name>
<name use="SYL">
  <family>きむら</family>
  <given>みちお</given>
</name>
<name use="ABC">
  <family>KIMURA</family>
  <given>MICHIO</given>
</name>
```

2.24 Tên thông thường (TN) chuyên biệt hóa EN

Định nghĩa: Một mục hẹp của tên thực thể, mà thực ra là một chuỗi đơn giản dùng để chỉ tên đơn giản của sự vật và địa điểm

Kiểu dữ liệu *TN* là một EN chỉ chứa một phần của tên mà không có bất kỳ kiểu phần của tên hay từ hạn định nào. *TN* và phần của tên duy nhất của nó vì vậy tương ứng với một chuỗi ký tự đơn giản. Sự tương ứng này được biểu diễn bằng một chuyển đổi giảm đã được định nghĩa thành kiểu dữ liệu ST và một chuyển đổi tăng từ ST.

```
type TrivialName alias TN specializes EN {
  demotion   ST;
```

```
  promotion  TN (ST x);
};

invariant(TN x) where x.nonNull {
  x.head.nonNull;
  x.tail.isEmpty;
  x.formatted.equal(x.head);
};

invariant(ST x) {
  ((TN)x).head.equal(x);
};
```

Tên thông thường thường được sử dụng cho địa điểm hoặc sự vật, như *Hồ Erie* hoặc *Sân bay quốc gia Washington-Reagan*:

Ví dụ 12:

```
<name>Lake Erie</name>
<name>Washington-Reagan National Airport</name>
```

2.25 Tên người (PN) chuyên biệt hóa EN

Định nghĩa: Kiểu dữ liệu EN được sử dụng khi Thực thể được đặt tên là một Người bao gồm một chuỗi các phần của tên, như tên hay họ, tiền tố, hậu tố, v.v. Một phần của tên là một mục hẹp của phần tên thực thể mà chỉ chấp nhận các từ hạn định phần tên thực thể dùng được cho tên người. Do cấu trúc của tên thực thể chủ yếu được quyết định bởi các yêu cầu về tên người, mục hẹp này là rất nhỏ.

Do hầu hết các chức năng của tên thực thể là để hỗ trợ tên người, tên người (PN) chỉ là một mục hẹp rất nhỏ trên từ hạn định phần của tên thực thể.

```
type PersonName alias PN specializes EN;

invariant(PN this) {
  forall(ENXP part)
      where this.contains(part) {
    part.qualifier.contains("LS").not;
    }
};
```

2.26 Tên tổ chức (ON) chuyên biệt hóa EN

Định nghĩa: Kiểu dữ liệu EN được sử dụng khi Thực thể được đặt tên là một Tổ chức bao gồm một chuỗi các thành phần của tên.

Là tên cho một tổ chức, như "Health Level Seven, Inc." Tên tổ chức chỉ bao gồm các thành phần không định kiểu của tên, các tiền tố, hậu tố và thành phần phân tách.

```
type OrganizationName alias ON specializes EN;


invariant(ON this) {
  forall(ENXP part)
      where this.contains(part) {
    part.type.implies("FAM").not;
    part.type.implies("GIV").not;
    }
};
```

2.26.1 Ví dụ

Dưới đây là tên tổ chức, "Health Level Seven, Inc." ở dạng chuỗi đơn giản:

Ví dụ 13:

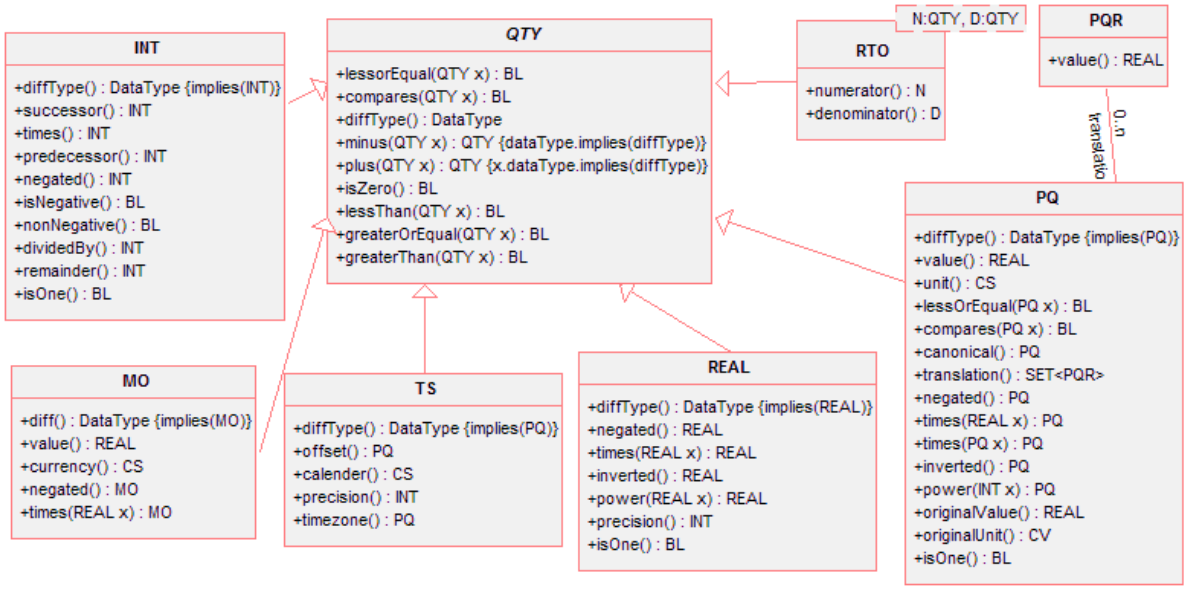
```
<name>Health Level Seven, Inc.</name>
```

Và với tình trạng pháp lý "Inc." như một phần phân biệt của tên:

Ví dụ 14:

```
<name>Health Level Seven, <suffix qualifier="LS">Inc.</suffix></name>
```

Các kiểu dữ liệu đại lượng



2.27 Đại lượng kiểu trừu tượng (QTY) chuyên biệt hóa ANY

Định nghĩa: Kiểu dữ liệu đại lượng là một khái quát hóa trừu tượng cho tất cả các kiểu dữ liệu (1) có tập hợp giá trị có quan hệ thứ tự (nhỏ hơn hay bằng) và (2) trong đó sự chênh lệch được định nghĩa trong các tập giá trị con được sắp xếp toàn phần của toàn bộ kiểu dữ liệu. Trừu tượng hóa kiểu đại lượng cần thiết khi định nghĩa một số kiểu khác, như khoảng và phân bổ xác suất.

```
abstract type Quantity alias QTY specializes ANY {
  BL    lessOrEqual(QTY x);
  BL    compares(QTY x);
  TYPE  diffType;
  QTY   minus(QTY x);
  QTY   plus(QTY x);
  BL    isZero;
  BL    lessThan(QTY x);
  BL    greaterOrEqual(QTY x);
  BL    greaterThan(QTY x);
};
```

2.27.1 Sắp xếp thứ tự: nhỏ hơn hoặc bằng: BL

Định nghĩa: Một vị từ thể hiện quan hệ thứ tự có tính phản xạ, bất đối xứng và bắc cầu giữa đại lượng này và đại lượng khác.

Quan hệ *nhỏ hơn hay bằng* được định nghĩa cho bất kỳ thành phần được sắp xếp toàn phần nào của kiểu dữ liệu đại lượng. Một phần được sắp xếp toàn phần là một tập con các giá trị đã được định nghĩa của kiểu dữ liệu trong đó tất cả các phần tử đều có thứ tự được định nghĩa (ví dụ, các số nguyên và số thực được sắp xếp toàn phần).

Ngược lại, một tập hợp được sắp xếp một phần là tập hợp trong đó một vài, chứ không phải tất cả các cặp phần tử đều là có thể so sánh được thông qua quan hệ thứ tự (ví dụ, cấu trúc cây hay tập hợp các đại lượng vật lý là tập hợp được sắp xếp một phần). Hai giá trị dữ liệu *x* và *y* của một kiểu được sắp xếp là có thể so sánh được (*x*.compares(*y*)) (*x*.so sánh(*y*)) nếu quan hệ nhỏ hơn hay bằng là đúng dù theo cách nào ($x \leq y$ hoặc $y \leq x$).

Một quan hệ thứ tự một phần tạo ra các tập con được sắp xếp toàn phần mà sự kết hợp của tất cả các tập con đó tạo thành toàn bộ tập hợp (ví dụ, tập hợp toàn bộ chiều dài là một tập con được sắp xếp toàn phần của tập hợp tất cả các đại lượng vật lý).

Ví dụ, một cấu trúc cây được sắp xếp một phần, trong đó phần gốc được coi là nhỏ hơn hay bằng một lá, nhưng có thể không có thứ tự giữa các lá. Ngoài ra, các đại lượng vật lý cũng được sắp xếp một phần, do thứ tự chỉ tồn tại giữa các đại lượng có cùng đơn vị (ví dụ, giữa hai độ dài, nhưng không thể giữa độ dài và thời gian). Một tập con được sắp xếp toàn phần của một cây là đường dẫn kết nối một cách bắc cầu lá với gốc. Kích thước vật lý của thời gian là một tập con được sắp xếp toàn bộ của các đại lượng vật lý.

```
invariant (QTY x, y, z)
  where x.nonNull.and(y.nonNull).and(z.nonNull) {
    x.lessOrEqual(x);       /* reflexive */
    x.lessOrEqual(y).implies(y.lessOrEqual(x)).not;/* asymmetric */
    x.lessOrEqual(y).and(y.lessOrEqual(z))
     .implies(x.lessOrEqual(z))       /* transitive */
};
```

2.27.2 Tương đương: BL, thừa kế từ ANY

Định nghĩa: Bình đẳng là một quan hệ có tính phản xạ, đối xứng, và bắc cầu giữa hai giá trị dữ liệu bất kỳ. Chỉ các giá trị hợp lệ mới có thể bình đẳng, các giá trị null không bao giờ tương đương (ngay cả khi chúng có cùng trạng thái null).

```
invariant(ANY x, y, z)
    where x.nonNull.and(y.nonNull).and(z.nonNull) {
  x.equal(x);             /* reflexivity */
  x.equal(y).equal(y.equal(x));        /* symmetry */
  x.equal(y).and(y.equal(z)).implies(x.equal(z)) /* transitivity */
  x.equal(y).implies(x.dataType.equal(y.dataType);
};
```

Phải định nghĩa bình đẳng được xác định thế nào cho từng kiểu dữ liệu. Nếu không có gì khác được nêu ra, hai giá trị dữ liệu là bình đẳng nếu không thể phân biệt

chúng, nghĩa là, nếu chúng không có khác biệt gì trong đặc tính ngữ nghĩa của chúng. Một kiểu dữ liệu có thể "ghi đè hoặc thay thế" định nghĩa chung này về bình đẳng, bằng cách chỉ ra mối quan hệ bình đẳng của riêng nó. Có thể sử dụng việc ghi đè mối quan hệ bình đẳng này để loại trừ các đặc tính ngữ nghĩa khỏi kiểm tra bình đẳng. Nếu một kiểu dữ liệu loại trừ các đặc tính ngữ nghĩa khỏi định nghĩa của nó về bình đẳng, điều này ngụ ý rằng có một vài đặc tính nhất định (hay các khía cạnh của đặc tính), mà không phải một phần của kiểm tra bình đẳng, không cần thiết đối với ý nghĩa của giá trị.

Ví dụ, đại lượng vật lý có hai đặc tính ngữ nghĩa (1) một số thực và (2) một đơn vị được mã hóa của phép đo. Tuy nhiên, kiểm tra bình đẳng phải giải thích được thực tế là, ví dụ, 1 mét bằng 100 centimét; bình đẳng độc lập của hai đặc tính ngữ nghĩa là một tiêu chí quá mạnh cho kiểm tra bình đẳng. Vì vậy, đại lượng vật lý phải thay thế định nghĩa bình đẳng.

2.27.3 Tính có thể so sánh: BL

Định nghĩa: Một vị từ chỉ ra liệu có thể so sánh giá trị này và toán hạng để xem cái nào lớn hơn.

Hai đại lượng là có thể so sánh nếu chúng đều là các phần tử của cùng một phần được sắp xếp toàn phần trong không gian giá trị của các kiểu dữ liệu của chúng. Định nghĩa này dựa trên quan hệ nhỏ hơn hay bằng.

```
invariant (QTY x, y, z)
    where x.nonNull.and(y.nonNull) {
  x.compares(y).equal(x.lessOrEqual(y).or(y.lessOrEqual(x)));
};
```

2.27.4 Kiểu dữ liệu hiệu: TYPE

Định nghĩa: Kiểu của hiệu của 2 giá trị của một kiểu dữ liệu QTY cụ thể.

```
invariant(QTY x) {
  x.diffType.implies(QTY)
};
```

Kiểu sẽ là một kiểu dữ liệu nào đó tiếp tục chuyên biệt hóa QTY

2.27.5 Hiệu: QTY

Định nghĩa: Một đại lượng thể hiện "khoảng cách" của đại lượng này so với đại lượng toán hạng, mà phải là so sánh được. Kiểu dữ liệu của đại lượng hiệu có liên hệ tới các đại lượng toán hạng nhưng không cần phải giống nhau.

```
invariant(QTY x, y) {
  x.minus(y).implies(x.diffType);
```

```
};
```

Kết quả của minus (trừ) có kiểu dữ liệu được trả về bởi đặc tính diffType (Kiểu hiệu) của thực thể.

Hiệu được định nghĩa trong một tập hợp được sắp xếp nếu việc phát biểu rằng Δ là hiệu giữa giá trị *x* và *y* là có ý nghĩa về mặt ngữ nghĩa. Hiệu Δ này phải có ý nghĩa một cách độc lập với các giá trị *x* và *y*. Sự độc lập này tồn tại nếu đối với tất cả các giá trị *u* có thể rút ra, một cách có ý nghĩa, một giá trị *v* sao cho Δ cũng là hiệu giữa *u* và *v*. Việc đánh giá thế nào là *có ý nghĩa* không thể được định nghĩa một cách chính thức.[43]

Phép *trừ* có kiểu dữ liệu có thể biểu thị hiệu của hai giá trị mà có quan hệ thứ tự được định nghĩa (nghĩa là hai phần tử của cùng một tập con được sắp xếp toàn phần). Ví dụ, kiểu dữ liệu hiệu của số nguyên là số nguyên, nhưng kiểu hiệu của thời điểm là một đại lượng vật lý theo đơn vị thời gian. Kiểu dữ liệu hiệu là một kiểu dữ liệu được sắp xếp toàn phần.

Hiệu của hai giá trị *x* trừ *y* phải được định nghĩa cho tất cả *x* và *y* trong cùng một tập con được sắp xếp toàn phần của tập hợp giá trị của kiểu dữ liệu. 0 là hiệu của một giá trị và chính nó.

```
invariant(QTY x, y)
    where x.compares(y) {
  x.minus(y).nonNull;
  x.minus(x).isZero;
};
```

Nếu x và y là không thể so sánh, thì hiệu có giá trị là Null

```
invariant(QTY x, y)
    where x.compares(y).Not {
  x.minus(y).notApplicable;
};
```

2.27.6 Phép cộng: QTY

Định nghĩa: Tổng của đại lượng này và toán hạng của nó. Toán hạng phải thuộc kiểu dữ liệu có thể biểu thị hiệu của hai giá trị của kiểu dữ liệu của đại lượng này.

```
invariant(QTY x, y)
    where x.compares(y) {
  x.plus(y.minus(x)).equal(y);
  y.dataType.implies(x.diffType);
};
```

Câu hỏi: mối quan hệ giữa {y.dataType.implies(x.diffType)} và {x.compares(y)} ({y.Kiểu dữ liệu.ngụ ý(x.Kiểu hiệu)} và {x.so sánh(y)}) là gì?

Nếu *y* không phải là kiểu hợp lệ cho hiệu của hai giá trị của kiểu *x*, thì kết quả của phép toán sẽ có giá trị là NULL.

```
invariant(QTY x, y)
    where y.dataType.implies(x.diffType).not {
  x.plus(y).notApplicable;
};
```

2.27.7 Đại lượng 0: BL

Định nghĩa: Phần tử trung lập trong các phép trừ và cộng, nghĩa là, nếu một đại lượng là 0, thì việc cộng vào, hay trừ đi bất kỳ đại lượng có thể so sánh nào khác sẽ cho kết quả là đại lượng đó.

```
invariant(QTY x) {
  x.minus(x).isZero;
};
```

2.27.8 Sắp xếp thứ tự: nhỏ hơn: BL

Định nghĩa: Một vị từ thể hiện quan hệ thứ tự có tính bất đối xứng và bắc cầu, giữa đại lượng này và một đại lượng khác. Việc sắp xếp thứ tự cũng tương tự như quan hệ nhỏ hơn hay bằng, nhưng không có tính phản xạ.

```
invariant (QTY x, y, z)
    where x.nonNull.and(y.nonNull) {
  x.lessThan(y).equal(x.lessOrEqual(y)
          .and(x.equal(y).not));
};
```

2.27.9 Sắp xếp thứ tự: lớn hơn hoặc bằng: BL

Định nghĩa: Một vị từ thể hiện quan hệ thứ tự có tính phản xạ, bất đối xứng và bắc cầu, giữa đại lượng này và một đại lượng khác. Đây là thứ tự nghịch đảo của nhỏ hơn hay bằng.

```
invariant (QTY x, y, z)
    where x.nonNull.and(y.nonNull) {
  x.greaterOrEqual(y).equal(y.lessOrEqual(x));
};
```

2.27.10 Sắp xếp thứ tự: lớn hơn: BL

Định nghĩa: Một vị từ thể hiện quan hệ thứ tự có tính bất đối xứng và bắc cầu, giữa đại lượng này và một đại lượng khác. Đây là nghịch đảo của nhỏ hơn.

```
invariant (QTY x, y, z)
    where x.nonNull.and(y.nonNull) {
  x.greaterThan(y).equal(y.lessThan(x));
};
```

2.28 Số nguyên (INT) chuyên biệt hóa QTY

Định nghĩa: Các số nguyên (-1,0,1,2, 100, 3398129, v.v.) là các số chính xác mà là kết quả của việc đếm và liệt kê. Các số nguyên là rời rạc, tập hợp các số nguyên là vô hạn nhưng đếm được. Không có giới hạn bất kỳ nào được áp đặt cho khoảng số nguyên. Hai trạng thái null/rỗng được xác định cho dương vô cùng và âm vô cùng.

```
type IntegerNumber alias INT specializes QTY {
        INT  successor;
        INT  times(INT x);
        INT  predecessor;
        INT  negated;
        BL   isNegative;
        BL   nonNegative;
        INT  dividedBy(INT x);
        INT  remainder(INT x);
        BL   isOne;
 literal  ST;
};
```

Do kiểu dữ liệu số nguyên bao gồm tất cả các ngữ nghĩa của khái niệm số nguyên toán học, các phép toán cơ bản cộng và nhân được định nghĩa. Các phép toán này được định nghĩa ở đây như các phép toán đặc trưng theo ý nghĩa của tiêu chuẩn ISO 11404, và bởi vì các phép toán này là cần thiết trong các phần khác của đặc tả này, cụ thể là ngữ nghĩa của dạng chữ.

Định nghĩa đệ quy truyền thống của phép cộng và phép nhân là của Grassmann, và sử dụng khái niệm phần tử tiếp sau.[44]

```
invariant(INT x, o, i)
    where x.nonNull.and(o.isZero) {
  x.lessThan(x.successor);
  x.plus(o).equal(x);
  x.plus(y.successor).equal(x.plus(y).successor);
  x.times(o).equal(o);
  x.times(y.successor).equal(x.times(y)).plus(x);
};
```

2.28.1 Phần tử tiếp sau: INT

Định nghĩa: Giá trị *INT* lớn hơn giá trị *INT* này nhưng không có giá trị *INT* nào tồn tại giữa giá trị này và phần tử tiếp sau của nó.

```
invariant(INT x, y)
    where x.successor(y) {
  x.lessThan(y).and.not(exists(INT z) {
    x.lessThan(z);
    z.lessThan(y);
    });
};
```

2.28.2 Kiểu dữ liệu hiệu: TYPE, thừa kế từ QTY

```
invariant(INT x) {
  x.diffType.implies(INT);
};
```

Hiệu giữa hai giá trị *INT* cũng là một giá trị *INT*.

2.28.3 Phép cộng: INT, thừa kế từ QTY

```
invariant(INT x, y, o)
    where x.nonNull.and(y.nonNull).and(o.isZero) {
  x.plus(o).equal(x);
  x.plus(y.successor).equal(x.plus(y).successor);
};
```

2.28.4 Phép nhân: INT

Định nghĩa: Kết quả của việc nhân số nguyên này với toán hạng, tương đương với việc lặp đi lặp lại phép cộng với số nguyên này.

```
invariant(INT x, y, i, o)
    where x.compares(y).and(o.isZero).and(i.isOne) {
  x.times(o).equal(o);
  x.times(i).equal(x);
  x.times(y.successor).equal(x.times(y)).plus(x);
};
```

2.28.5 Phần tử trước: INT

Định nghĩa: Nghịch đảo của phần tử tiếp sau.

```
invariant(INT x, y)
    where x.successor(y) {
```

```
    x.successor.predecessor.equal(x);
};
```

2.28.6 Phủ định: INT

Định nghĩa: Phần tử nghịch đảo của giá trị *INT*, một giá trị *INT* khác, mà, khi được cộng với giá trị đó cho kết quả bằng 0 (phần tử trung lập).

```
invariant(INT x)
    where x.nonNull {
  x.plus(x.negated).isZero;
};
```

2.28.7 Không âm: BL

Định nghĩa: Một vị từ chỉ ra liệu đại lượng 0 *INT* (phần tử trung lập) là nhỏ hơn hay bằng giá trị *INT* này.

```
invariant(INT x, o)
     where x.nonNull.and(o.isZero) {
  x.nonNegative.equal(o.lessOrEqual(x));
};
```

2.28.8 Âm: BL

Định nghĩa: Một vị từ chỉ ra liệu giá trị *INT* này có nhỏ hơn 0 (không không-âm).

```
invariant(INT x)
    where x.nonNull {
  x.isNegative.equal(x.nonNegative.not);
};
```

2.28.9 Chia số nguyên: INT

Định nghĩa: Phép chia số nguyên của số nguyên này (số bị chia) cho một số nguyên khác (số chia) là số nguyên, số lần số chia khớp với số bị chia.

```
invariant(INT dividend, divisor, o, i)
    where divisor.isZero.not.and(o.isZero) {
  dividend.isZero.implies(dividend.dividedBy(divisor).equal(o));
  dividend.isZero.not.implies(dividend.dividedBy(divisor).equal(
absolute(dividend).minus(absolute(divisor)).dividedBy(absolute(divisor))
                .successor.times(sign(dividend)
                .times(sign(divisor)))));
};
```

2.28.10 Số dư: INT

Định nghĩa: Số dư của phép chia số nguyên.

```
invariant(INT x, y)
    where x.nonNull.and(y.nonNull) {
  x.reminder(y).equal(x.minus(x.dividedBy(z).times(y)));
};
```

Định nghĩa này của số dư khớp với các ngôn ngữ lập trình C và Java.

2.28.11 Phần tử trung lập của phép nhân: BL

Định nghĩa: Một vị từ cho biết liệu giá trị này có phải là số 1, tức là phần tử trung lập của phép nhân. Có chính xác một số nguyên có đặc tính này.

```
invariant(INT x, y)
    where x.nonNull.and(y.nonNull) {
  x.isOne.and(y.isOne).implies(x.equal(y));
  x.isOne.and(y.isZero).implies(x.equal(y).not);
};
```

2.28.12 Dạng chữ

Dạng chữ của một số nguyên là một số thập phân đơn giản, tức là một chuỗi các chữ số thập phân.

```
INT.literal ST {
  INT digit : "0"       { $.isZero; }
            | "1"       { $.equal(0.successor); }
            | "2"       { $.equal(1.successor); }
            | "3"       { $.equal(2.successor); }
            | "4"       { $.equal(3.successor); }
            | "5"       { $.equal(4.successor); }
            | "6"       { $.equal(5.successor); }
            | "7"       { $.equal(6.successor); }
            | "8"       { $.equal(7.successor); }
            | "9"       { $.equal(8.successor); };

  INT uint  : digit     { $.equal($1); }
            | uint digit { $.equal($1.times(9.successor).plus($2)); };

  INT       : uint      { $.equal($1); }
            | "+" uint  { $.equal($2); }
```

```
        | "-" uint    { $.equal($2.negated); };
};
```

2.29 Số thực (REAL) chuyên biệt hóa QTY

Định nghĩa: Các phân số, thường được dùng khi đo đạc, ước tính, hoặc tính toán các đại lượng từ các số thực khác. Cách biểu diễn tiêu biểu là số thập phân, trong đó số lượng các số thập phân quan trọng được gọi là độ chính xác.

Từ "số thực" được sử dụng trong đặc tả này với ý rằng các giá trị phân số được nhắc tới nhưng không nhất thiết có ý chỉ tập hợp đầy đủ các số thực toán học sẽ bao gồm các số vô tỷ như ρ, số Euler, v.v.[45]

LƯU Ý: Đặc tả này định nghĩa kiểu dữ liệu số thực theo nghĩa rộng nhất có thể. Tuy nhiên, nó không ngụ ý rằng bất kỳ ITS hay triển khai nào tuân theo đặc tả này phải có khả năng biểu diễn toàn dải số thức, vốn là điều không thể trong bất kỳ triển khai hữu hạn nào. Các trường hợp sử dụng hiện nay của HL7 đối với kiểu dữ liệu số thực là các đại lượng được đo đạc, ước tính và số tiền. Có thể xử lý các trường hợp sử dụng này với một không gian giá trị thực hạn hẹp, các số hữu tỷ, và thậm chí chỉ các số thập phân rất hạn chế (số nguyên chia tỷ lệ). Tuy nhiên, chúng tôi khai báo các biểu diễn không gian giá trị thực là dấu phẩy động, số hữu tỷ, số nguyên chia tỷ lệ, hay chuỗi chữ số, còn các giới hạn khác nhau của chúng nằm ngoài phạm vi của đặc tả này.

Đặc tả này có hai lựa chọn cho một kiểu dữ liệu số. Lựa chọn được đưa ra như sau: Bất kỳ thuộc tính số nào đều là số thực nếu không biết chắc rằng nó là số nguyên. Một số là số nguyên nếu nó *luôn luôn* được đếm, thường biểu diễn một số thứ tự. Nếu có những trường hợp sử dụng trong đó một số như vậy được ước tính hay tính trung bình, nó không nhất thiết là một số nguyên và do đó nên sử dụng kiểu dữ liệu số thực.

```
type RealNumber alias REAL specializes QTY {
        REAL  negated;
        REAL  times(REAL x);
        REAL  inverted;
        BL    isOne;
        REAL  power(REAL x);
  literal    ST;
        INT   precision;
  demotion   INT;
  promotion  REAL  (INT x);
  promotion  PQ;
  promotion  RTO;
};
```

Các phép toán đại số được xác định ở đây như các phép toán đặc trưng theo ý nghĩa của tiêu chuẩn ISO 11404, và bởi vì các phép toán này là cần thiết trong các phần khác của đặc tả này.

Không giống các số nguyên, ngữ nghĩa số thực không được xây dựng theo suy luận quy nạp mà chỉ được mô tả trực quan bằng các tiên đề của chúng về các đặc tính đại số của chúng. Các tiên đề về tính hoàn chỉnh được cố ý bỏ qua để không đưa ra tuyên bố nào về số vô tỷ.

2.29.1 Tính có thể so sánh: BL, thừa kế từ QTY

Tập hợp giá trị của *REAL* được sắp xếp toàn bộ.

```
invariant(REAL x, y)
    where x.nonNull.and(y.nonNull) {
  x.compares(y);
};
```

2.29.2 Kiểu dữ liệu hiệu: TYPE, thừa kế từ QTY

```
invariant(REAL x) {
  x.diffType.implies(REAL);
};
```

Hiệu giữa hai giá trị *REAL (số thực)* cũng là một giá trị *REAL*.

2.29.3 Phép cộng: QTY, thừa kế từ QTY

```
invariant(REAL x, y, z, o)
    where x.nonNull.and(y.nonNull).and(z.nonNull).and(o.isZero) {
  x.plus(o).equal(x);              /* neutral element */
  x.plus(y).plus(z).equal(x.plus(y.plus(z)));    /* associative */
  x.plus(y).equal(y.plus(x));                    /* commutative */
  z.lessOrEqual(x).and(z.lessOrEqual(y)
              .implies(z.lessOrEqual(x.plus(y));
  x.lessOrEqual(y).implies(x.plus(z)
              .lessOrEqual(y.plus(z)));
};
```

2.29.4 Phủ định (Phần tử nghịch đảo của phép cộng): REAL

Định nghĩa: Một giá trị *REAL*, mà khi được cộng với một giá trị *REAL* khác cho kết quả bằng 0 (phần tử trung lập của phép cộng).

```
invariant(REAL x)
    where x.nonNull {
  x.plus(x.negated).isZero;
```

```
};
```

2.29.5 Phần tử trung lập của phép nhân: BL

Định nghĩa: Một vị từ cho biết liệu giá trị này có phải là số 1, tức là phần tử trung lập của phép nhân. Có chính xác một số thực có đặc tính này.

```
invariant(REAL x, y)
    where x.nonNull.and(y.nonNull) {
  x.isOne.and(y.isOne).implies(x.equal(y));
  x.isOne.and(y.isZero).implies(x.equal(y).not);
};
```

2.29.6 Phép nhân: REAL

Định nghĩa: Một phép toán trong *REAL* hình thành một nhóm giao hoán (nhóm aben) và có liên quan tới phép cộng theo luật phân phối.

```
invariant(REAL x, y, z, i, o)
    where x.nonNull.and(y.nonNull).and(z.nonNull)
        .and(i.isOne).and(o.isZero) {
  x.times(o).equal(o);
  x.times(i).equal(x);              /* neutral element */
  x.times(y).times(z).equal(x.times(y.times(z))); /* associative */
  x.times(y).equal(y.times(x));             /* commutative */
  x.times(y.plus(z)).equal(x.times(y).plus(x.times(z));/* distributive */
o.lessOrEqual(x).and(o.lessOrEqual(y).implies(o.lessOrEqual(x.times(y));
};
```

2.29.7 Phần tử nghịch đảo của phép nhân: REAL

Định nghĩa: Một giá trị *REAL*, mà khi được nhân với một giá trị *REAL* khác cho kết quả bằng 1 (phần tử trung lập của phép nhân). 0 (phần tử trung lập của phép cộng) không có phần tử nghịch đảo.

```
invariant(REAL x, i)
    where x.isZero.not.and(i.isOne) {
  x.times(x.inverted).equal(i);
};
```

2.29.8 Đồng cấu của INT vào REAL: INT

Các kiểu dữ liệu INT và *REAL* được liên hệ bởi một đồng cấu liên kết mọi giá trị trong INT với một giá trị trong *REAL* theo đó các đặc tính đại số của INT được bảo toàn. Điều này có nghĩa là, có thể chuyển đổi tăng một số nguyên thành một số thực và

có thể chuyển đổi giảm một số thực thành một số nguyên bằng cách làm tròn phần phân số.

```
invariant(INT n, m)
    where n.nonNull.and(m.nonNull) {
  ((REAL)n.plus(m)).equal(((REAL)n).plus((REAL)m));
  ((REAL)n.times(m)).equal(((REAL)n).times((REAL)m));
};
```

2.29.9 Lũy thừa: REAL

Định nghĩa: Cơ sở của lũy thừa là lặp đi lặp lại phép nhân với một số thực, và được mở rộng tới các số mũ hợp lý như phép nghịch đảo.

Chúng tôi chỉ liệt kê một số đặc tính phổ biến của lũy thừa.

```
invariant(REAL x, y, z, o, i)
    where x.nonNull.and(y.nonNull).and(z.nonNull)
        .and(o.isZero).and.(i.isOne) {
  forall(INT n)
      where n.nonNull {
    n.greaterThan(o).implies(
      x.power(n).equal(x.times(x.power(n.predecessor))));
    n.lessThan(o).implies(
      x.power(n).equal(x.power(n.negated).inverted);
    }
    x.power(o).equal(i);
    x.power(i).equal(x);
    x.power(y).power(z).equal(x.power(y.times(z)));
    x.power(y).times(x.power(z)).equal(x.power(y.plus(z)));
    x.power(y).inverted.equal(x.power(y.negated));
    x.power(y).power(y.inverted).equal(x);
};
```

2.29.10 Dạng chữ

Dạng chữ của một số thực là một chuỗi các số thập phân với các dấu "+" hoặc "-" tùy chọn đứng trước, và dấu thập phân tùy chọn, và ký hiệu số mũ tùy chọn sử dụng chữ "e" không phân biệt viết hoa hay viết thường giữa phần định trị và số mũ (logarit). Số lượng các số quan trọng phải tuân thủ đặc tính độ chính xác.

```
REAL.literal ST {
```

```
  REAL    : mantissa            { $.equal($1); }
          | mantissa /[eE]/ INT      {$.equal($1.times(10.power($3)); };

  REAL mantissa  : /0*/ 0       { $.isZero; $.precision.equal(1); }
                 | /0*/ "." /0*/ { $.isZero;
                                   $.precision.equal($3.length.successor); }
                 | /0*/ "." /0*/ fractional   { $.equal($4);
                                                $.precision.equal($4.precision); }
                 | integer     { $.equal($1); }
                 | integer "." fractional  { $.equal($1.plus($2));
                $.precision.equal($1.precision.plus($3.precision)); };

  REAL integer    : uintval        { $.equal($2); }
                  | "+" uintval     { $.equal($1.times($2)); }
                  | "-" uintval  { $.equal($1.times($2).negated); };

  REAL uintval    : /0*/ uint               { $.equal($2); };

  REAL uint       : digit      { $.equal($1);$.precision.equal(1); }
                  | uint digit { $.equal($1.times(10).plus($2));
                    $.precision.equal($1.precision.successor); };

  REAL fractional : digit      { $.equal($1.times(10.inverted));
                                             $.precision.equal(1); }
                  | digit fractional
                                        {$.equal($1.plus($2.times(10.inverted));
                              $.precision.equal($1.precision.successor); };
  INT digit       : /[0-9]/                 { $.equal($1); }
};
```

Ví dụ cho dạng chữ số thực của hai nghìn là 2000, 2000., 2e3, 2.0e+3, +2.0e+3.

Lưu ý rằng dạng chữ không chứa thông tin kiểu dữ liệu. Ví dụ, "2000" là một biểu diễn hợp lệ cho cả một số thực và một số nguyên. Không có dấu thập phân phía sau nào được sử dụng để loại bỏ mọi sự không rõ nghĩa khỏi các số nguyên. ITS sử dụng dạng chữ này phải phục hồi thông tin kiểu dữ liệu từ các nguồn khác.

2.29.11 Độ chính xác của dạng thập phân: INT

Định nghĩa: Số lượng các số quan trọng của biểu diễn thập phân.

Độ chính xác được định nghĩa một cách chính thức dựa trên dạng chữ

Thuộc tính độ chính xác chỉ là độ chính xác của biểu diễn số thập phân, *không phải là tính đúng đắn của giá trị số thực*.

Mục đích của đặc tính độ chính xác cho kiểu dữ liệu số thực là để có thể nắm bắt một cách trung thực toàn bộ thông tin được trình bày cho con người trong một con số. Số lượng số thập phân được trình bày chuyển tải thông tin về tính không chắc chắn (nghĩa là, độ chính xác và tính đúng đắn) của một giá trị được đo đạc.

LƯU Ý: Độ chính xác của một biểu diễn không phụ thuộc vào tính không chắc chắn (độ chính xác và tính đúng đắn) của kết quả đo đạc. Nếu tính không chắc chắn của một kết quả đo đạc là quan trọng, nên định rõ các giá trị không chắc chắn là PPD

Các quy tắc để xác định các số quan trọng là như sau:

1. Tất cả các số khác 0 đều là quan trọng.
2. Tất cả các số 0 nằm về bên phải của một số quan trọng đều là quan trọng.
3. Khi tất cả các chữ số trong một số đều là 0, chữ số 0 liền kề bên trái dấu thập phân là quan trọng (và do có quy tắc 2, tất cả các số 0 tiếp sau đó cũng là quan trọng.)

LƯU Ý: Các quy tắc về tính quan trọng này hơi khác so với các quy tắc thông thường hơn được dạy ở trường. Đáng chú ý là các số 0 phía sau nằm trước dấu thập phân luôn được coi là quan trọng ở đây. Ở những tài liệu khác, ví dụ, không rõ trong số 2000 liệu các số 0 có quan trọng hay không. Sự khác biệt này so với thường lệ nhằm mục đích đảm bảo việc thông tin một cách rõ ràng.

Bảng 30. Ví dụ về tính chính xác của dạng chữ số thực.

Dạng chữ	**Số lượng chữ số quan trọng**
2000	có 4 chữ số quan trọng.
2e3	có 1 chữ số quan trọng, được sử dụng khi nói một cách tự nhiên "2000" nhưng độ chính xác chỉ là 1.
0.001	có 1 chữ số quan trọng.
1e-3	có 1 chữ số quan trọng, được sử dụng khi nói một cách tự nhiên "0.001" nhưng độ chính xác chỉ là 1.
0	có 1 chữ số quan trọng.
0.0	có 2 chữ số quan trọng.

Dạng chữ	Số lượng chữ số quan trọng
000.0	có 2 chữ số quan trọng.
0.00	có 3 chữ số quan trọng.
4.10	có 3 chữ số quan trọng.
4.09	có 3 chữ số quan trọng.
4.1	có 2 chữ số quan trọng.

Độ chính xác của biểu diễn *nên* khớp với tính không chắc chắn của giá trị. Tuy nhiên, độ chính xác của biểu diễn và tính không chắc chắn của giá trị là hai khái niệm độc lập riêng biệt. Tham khảo PPD<REAL> để biết thêm chi tiết về các số thực không chắc chắn.

Ví dụ, "0.123" có 3 chữ số quan trọng *trong biểu diễn*, nhưng *tính không chắc chắn của giá trị này* có thể ở trong bất kỳ số nào được thể hiện hay không được thể hiện, nghĩa là, tính không chắc chắn có thể là 0.123±0.0005, 0.123±0.005 hay 0.123±0.00005, v.v. Lưu ý rằng các biểu diễn ITS *nên* điều chỉnh độ chính xác của phép biểu diễn với tính không chắc chắn của giá trị. Tuy nhiên, do độ chính xác trong chuỗi chữ số lẻ đến 0.5 số ít quan trọng nhất, trong khi độ không chắc chắn có thể ở bất kỳ đâu giữa các "đường lưới" này, 0.123±0.005 cũng sẽ là một biểu diễn đầy đủ cho giá trị giữa 0.118 và 0.128.

LƯU Ý: Trên một công nghệ triển khai dựa trên ký tự, ITS không cần biểu diễn độ chính xác như một thuộc tính rõ ràng nếu các số được biểu diễn như các chuỗi số thập phân. Trong trường hợp đó, ITS phải tuân theo các quy tắc của một quy định rõ ràng về các chữ số quan trọng. Một biểu diễn số không được tạo ra nhiều hay ít các chữ số quan trọng hơn so với số ban đầu. Có thể kiểm tra mức phù hợp thông qua quy trình mã hóa — giải mã — mã hóa.

2.30 Tỷ lệ (RTO) chuyên biệt hóa QTY

Định nghĩa: Một đại lượng được cấu tạo như thương số của một đại lượng tử số chia cho một đại lượng mẫu số. Các thừa số chung trong tử số và mẫu số không được tự động triệt tiêu. Kiểu dữ liệu *RTO* hỗ trợ độ chuẩn (ví dụ “1:128”) và các đại lượng khác từ phòng thí nghiệm mà thực sự biểu diễn các tỷ lệ. Tỷ lệ không đơn giản là “các con số có cấu trúc”, đặc biệt các giá trị đo huyết áp (ví dụ “120/60”) không phải là tỷ lệ. Trong nhiều trường hợp, nên sử dụng kiểu dữ liệu REAL thay cho *RTO*.

Tỷ lệ khác số hữu tỷ, nghĩa là, đối với tỷ lệ các thừa số chung trong tử số và mẫu số không bao giờ triệt tiêu. Tỷ lệ của hai số thực hoặc nguyên không được tự động rút gọn về một số thực.

Bảng 31. Tóm tắt đặc tính của tỷ lệ

Tên	Kiểu	Mô tả
numerator (tử số)	N	Đại lượng bị chia trong tỷ lệ. Mặc định là số nguyên 1 (một).
denominator (mẫu số)	D	Đại lượng chia tử số trong tỷ lệ. Mặc định là số nguyên 1 (một). Mẫu số không được là 0.

LƯU Ý: Kiểu dữ liệu này được định nghĩa không phải để biểu diễn chung chung các số hữu tỷ. Nó chỉ được sử dụng khi các thừa số chung trong tử và mẫu không được triệt tiêu. Điều này hiếm khi xảy ra. Đối với các giá trị quan sát, hầu như chỉ có tỷ lệ với độ chuẩn.

```
type Ratio<QTY N, QTY D> alias RTO specializes QTY {
  N       numerator;
  D       denominator;
  demotion  REAL;
  demotion  PQ;
};
```

Giá trị mặc định cho cả tử số và mẫu số là số nguyên 1 (một). Mẫu số không được là 0.

LƯU Ý: Kiểu dữ liệu này được định nghĩa như một kiểu dữ liệu chung (xem Kiểu dữ liệu chung (§ 1.9.5)) nhưng được thảo luận trong bối cảnh của các kiểu dữ liệu khác có liên quan tới đại lượng. Lí do để định nghĩa *RTO* như một kiểu dữ liệu chung là để nó có thể được ràng buộc một cách chính xác về việc tử số và mẫu số nên có kiểu gì.

2.30.1 Tử số: N

Định nghĩa: Đại lượng bị chia trong tỷ lệ. Mặc định là số nguyên 1 (một).

2.30.2 Mẫu số: D

Định nghĩa: Đại lượng chia tử số trong tỷ lệ. Mặc định là số nguyên 1 (một). Mẫu số không được là 0.

```
invariant(RTO x)
    where x.nonNull {
```

```
    x.denominator.isZero.not;
};
```

2.30.3 Dạng chữ

Dạng chữ của tỷ lệ tồn tại đối với tất cả các tỷ lệ trong đó cả tử số và mẫu số đều có dạng chữ. Một tỷ lệ đơn giản là dạng chữ tử số, một dấu hai chấm làm dấu phân cách, tiếp theo là dạng chữ mẫu số. Khi thiếu dấu hai chấm và mẫu số, số nguyên 1 được mặc định là mẫu số.

```
RTO.literal ST {
  RTO : QTY        { $.numerator.equal($1);
                     $.denominator.equal((INT)1); };
    | QTY ":" QTY  { $.numerator.equal($1);
                     $.denominator.equal($3); };
};
```

Ví dụ, giá trị độ chuẩn kháng thể kháng virus rubella 1:64 có thể được biểu diễn bằng dạng chữ "1:64".

2.31 Đại lượng vật lý (PQ) chuyên biệt hóa QTY

Định nghĩa: Một đại lượng có đơn vị thể hiện một kết quả đo đạc.

Bảng 32. Tóm tắt đặc tính của Đại lượng vật lý

Tên	Kiểu	Mô tả
value (giá trị)	REAL	Độ lớn của đại lượng được đo xét theo đơn vị.
unit (đơn vị)	CS	Đơn vị của phép đo được xác định trong Mã thống nhất cho đơn vị đo lường (UCUM) [http://aurora.rg.iupui.edu/UCUM].
translation (chuyển ngữ)	SET<PQR>	Một biểu diễn thay thế cho cùng một đại lượng vật lý được thể hiện với một đơn vị khác, của một hệ thống mã đơn vị khác và có thể với một giá trị khác.
canonical (chính tắc)	PQ	Một đại lượng vật lý được thể hiện theo đơn vị chính tắc. Trong một hệ thống đơn vị cho trước bất kỳ, mỗi kích thước vật lý đều có thể được gán một đơn vị chính tắc. Định nghĩa đơn vị chính tắc không phải là đối tượng của quy chuẩn này, chỉ khẳng định rằng có tồn tại một đơn vị chính tắc như vậy (và nó có thể được tùy chọn) cho mỗi đại lượng vật lý. Một đại lượng vật lý trừu tượng bình

Tên	Kiểu	Mô tả
		đẳng với dạng chính tắc của nó.
diffType (Kiểu hiệu)	TYPE	Kiểu của hiệu của 2 giá trị của một kiểu dữ liệu QTY cụ thể.
toPQ (tới PQ)	REAL	

```
type PhysicalQuantity alias PQ specializes QTY {
        REAL     value;
        CS       unit;
        BL       equal(ANY x)
        BL       lessOrEqual(PQ x);
        BL       compares(PQ x);
        PQ       canonical;
        SET<PQR>  translation;

        PQ       negated;
        PQ       times(REAL x);
        PQ       times(PQ x);
        PQ       inverted;
        PQ       power(INT x);
        BL       isOne;

  literal   ST;
  demotion  REAL;

        REAL     originalValue;
        CV       originalUnit;
};
```

2.31.1 Giá trị độ lớn: REAL

Định nghĩa: Độ lớn của đại lượng được đo xét theo đơn vị.

2.31.2 Đơn vị đo: CS

Định nghĩa: Đơn vị của phép đo được xác định trong Mã thống nhất cho đơn vị đo (UCUM) [http://aurora.rg.iupui.edu/UCUM].

LƯU Ý: Tính bình đẳng của các đại lượng vật lý không yêu cầu giá trị và đơn vị phải bình đẳng một cách độc lập. Giá trị và đơn vị chỉ là cách chúng ta biểu diễn đại lượng vật lý. Ví dụ, 1 m bằng với 100 cm. Mặc dù đơn vị là khác nhau và giá trị là khác nhau, các đại lượng vật lý vẫn bình đẳng! Vì vậy, không bao giờ nên mong đợi một đơn vị cụ thể cho một đại lượng vật lý mà thay vào đó cung cấp chuyển đổi tự động giữa các đơn vị có thể so sánh khác nhau.

2.31.3 Chuyển ngôn ngữ: SET<PQR>

Định nghĩa: Một biểu diễn thay thế cho cùng một đại lượng vật lý được thể hiện với một đơn vị khác, của một hệ thống mã đơn vị khác và có thể với một giá trị khác.

Về mặt ngữ nghĩa, các đại lượng vật lý là kết quả của hành động đo lường. Mặc dù các đại lượng vật lý được biểu diễn như các cặp giá trị và đơn vị, về mặt ngữ nghĩa, một đại lượng vật lý là nhiều hơn thế. Để biết được liệu hai đại lượng vật lý có bình đẳng, việc so sánh tính bình đẳng của hai giá trị và đơn vị của chúng một cách độc lập là không đủ. Ví dụ, 100 cm bằng với 1 m mặc dù cả giá trị lẫn đơn vị đều không bình đẳng. Để định nghĩa bình đẳng, chúng tôi giới thiệu khái niệm dạng chính tắc.

2.31.4 Dạng chính tắc: PQ

Định nghĩa: Một đại lượng vật lý được thể hiện theo đơn vị chính tắc. Trong một hệ thống đơn vị cho trước bất kỳ, mỗi kích thước vật lý đều có thể được gán một đơn vị chính tắc. Định nghĩa đơn vị chính tắc không phải là đối tượng của quy chuẩn này, chỉ khẳng định rằng có tồn tại một đơn vị chính tắc như vậy (và nó có thể được tùy chọn) cho mỗi đại lượng vật lý. Một đại lượng vật lý trừu tượng bình đẳng với dạng chính tắc của nó.

```
invariant(PQ x, y)
    where x.nonNull.and(y.nonNull) {
  x.canonical.equal(x);
};
```

Ví dụ, đối với một hệ thống đơn vị dựa trên Hệ đo lường quốc tế (SI), có thể định nghĩa dạng chính tắc là (a) sản phẩm chỉ của các đơn vị cơ sở; (b) không có tiền tố; trong đó (c) chỉ phép nhân và số mũ được sử dụng (không có phép chia); và (d) trong đó bảy đơn vị cơ sở xuất hiện theo một thứ tự được định nghĩa (ví dụ, m, s, g...) Vì vậy, 1 mm Hg sẽ được thể hiện là 133322 m^{-1} s^{-2}. Như có thể thấy, các quy tắc về cách xây dựng dạng chính tắc của đơn vị có thể rất phức tạp. Tuy nhiên, đối với đặc tả ngữ nghĩa, việc dạng chính tắc được xây dựng như thế nào, hay dạng chính tắc cụ thể

nào được lựa chọn, là không quan trọng, chỉ cần một dạng chính tắc *nào đó có thể* được định nghĩa.

2.31.5 Tương đương: BL, thừa kế từ ANY

Hai đại lượng vật lý là bình đẳng nếu mỗi giá trị của chúng và mỗi đơn vị của chúng của các dạng chính tắc của chúng là bình đẳng.

```
invariant(PQ x, y)
    where x.nonNull.and(y.nonNull) {
  x.equal(y).equal(x.canonical.value
        .equal(y.canonical.value).and(
          x.canonical.unit.equal(y.canonical.unit)));
};
```

2.31.6 Tính có thể so sánh: BL, thừa kế từ QTY

Hai đại lượng vật lý so sánh với nhau (và có thứ tự và hiệu) nếu các đơn vị của các dạng chính tắc của chúng là bình đẳng.

```
invariant(PQ x, y)
    where x.nonNull.and(y.nonNull) {
  x.compares(y).equal(x.canonical.unit.equal(y.canonical.unit));
};
```

2.31.7 Kiểu dữ liệu hiệu: TYPE, thừa kế từ QTY

```
invariant(PQ x) {
  x.diffType.implies(PQ);
};
```

Hiệu của 2 Đại lượng vật lý là một Đại lượng vật lý khác với cùng đơn vị

```
invariant(PQ x, y)
    where x.compares(y) {
  x.minus(y).canonical.unit.implies(x.canonical.unit);
};
```

2.31.8 Phần tử trung lập của phép nhân: BL

Định nghĩa: Một vị từ chỉ ra liệu giá trị này có phải là số 1, tức là phần tử trung lập của phép nhân. Có chính xác một đại lượng vật lý có đặc tính này và được gọi là *phần tử đơn vị*.

```
invariant(PQ x, y)
    where x.nonNull.and(y.nonNull) {
  x.isOne.and(y.isOne).implies(x.equal(y));
  x.isOne.and(y.isZero).implies(x.equal(y).not);
```

```
};
```

2.31.9 Phép nhân: PQ

Định nghĩa: Tích của hai đại lượng vật lý là tích của giá trị của chúng nhân với tích của đơn vị của chúng.

```
invariant(PQ x, y, z, i, o)
    where x.nonNull.and(y.nonNull).and(z.nonNull)
          .and(o.isZero).and(i.isOne) {
  x.times(o).equal(o);
  x.times(i).equal(x);            /* neutral element */
  x.times(y).times(z).equal(
        x.times(y.times(z)));   /* associative */
  x.times(y).equal(y.times(x));    /* commutative */
  o.lessOrEqual(x).and(o.lessOrEqual(y).implies(o.lessOrEqual(x.times(y));
};
```

2.31.10 Phần tử nghịch đảo của phép nhân: PQ

Định nghĩa: Một giá trị *PQ*, mà, khi nhân với một giá trị *PQ* khác cho kết quả là 1 (phần tử trung lập của phép nhân). Số 0 (phần tử trung lập của phép cộng) không có phần tử nghịch đảo. Thương của hai đại lượng có thể so sánh là có thể so sánh với phần tử đơn vị (đơn vị 1).

```
invariant(PQ this, that, one)
    where this.nonNull.and(that.nonNull).and(one.isOne) {
  this.times(this.inverted).equal(one);
  this.compares(that).implies(this.times(that.inverted).equal(one));
};
```

2.31.11 Phép nhân số thực: PQ

Định nghĩa: Phép nhân với một số thực tạo ra một đại lượng có chia tỷ lệ. Có thể so sánh một đại lượng có chia tỷ lệ với đại lượng gốc của nó.

Nếu hai đại lượng Q_1 và Q_2 so sánh với nhau, thì tồn tại một số thực *r* sao cho $r\mathbf{1} = Q_1 / Q_2$.

```
invariant(PQ x; REAL r)
    where x.nonNull.and(r.nonNull) {
  x.times(r).value.equal(x.value.times(r));
  x.times(r).compares(x);
};
```

2.31.12 Phép biến đổi của REAL sang PQ: REAL

Một giá trị REAL có thể được chuyển đổi thành một giá trị *PQ* với phần tử đơn vị, nghĩa là đơn vị **1** (một). Tương tự, một đại lượng vật lý so sánh phần tử đơn vị có thể được chuyển đổi thành một số thực.

```
invariant(PQ x, unity)
    where x.nonNull.and.unity.isOne.and(x.compares(unity)) {
  unity.times((REAL)x).equal(x);
};
```

2.31.13 Lũy thừa: PQ

Định nghĩa: Một đại lượng vật lý có thể được nâng lên thành lũy thừa nguyên.

```
invariant (PQ x, i; INT n, o)
    where x.nonNull.and(i.isOne).and(n.nonNull.and(o.isZero) {
  x.power(o).equal(i);
  n.greaterThan(o).implies(
    x.power(n).equal(x.times(x.power(n.predecessor))));
  n.lessThan(o).implies(
    x.power(n).equal(x.power(n.negated).inverted);
};
```

2.31.14 Phép cộng: PQ

Định nghĩa: Hai đại lượng vật lý mà so sánh với nhau có thể được cộng.

```
invariant (PQ x, y)
    where x.compares(y) {
  x.canonical.plus(y.canonical).value.equal(
    x.canonical.value.plus(y.canonical.value));
};
```

2.31.15 Dạng chữ

Dạng chữ cho một đại lượng vật lý là một dạng chữ số thực theo sau bởi khoảng trắng tùy chọn và một chuỗi ký tự biểu diễn một mã hợp lệ trong Mã thống nhất cho đơn vị đo lường (UCUM) [http://aurora.rg.iupui.edu/UCUM].

```
PQ.literal ST {
  PQ      : REAL unit    { $.value.equal($1);
                           $.unit.equal($2); }
  CS  unit : ST          { $.value.equal($1);
                  $.codeSystem.equal(2.16.840.1.113883.6.8); };
};
```

Ví dụ, 20 phút là "20 min" (20p).

2.32 Biểu diễn đại lượng vật lý (PQR) chuyên biệt hóa CV

Định nghĩa: Một phần mở rộng của kiểu dữ liệu giá trị được mã hóa, biểu diễn một đại lượng vật lý nhờ dùng một đơn vị từ hệ thống mã bất kỳ. Được sử dụng để thể hiện cách biểu diễn thay thế cho một đại lượng vật lý.

Bảng 33. Tóm tắt đặc tính của Biểu diễn đại lượng vật lý

Tên	Kiểu	Mô tả
value (giá trị)	REAL	Độ lớn của giá trị đo xét theo đơn vị do mã này xác định.
code (mã)	ST	Ký hiệu mã đơn giản được hệ thống mã định nghĩa. Ví dụ, "784.0" là ký hiệu mã của mã ICD-9 "784.0" cho bệnh đau đầu.
codeSystem (Hệ thống mã)	UID	Xác định hệ thống mã định nghĩa mã.
codeSystemName (tên hệ thống mã)	ST	Tên thông thường của hệ thống mã hóa.
codeSystemVersion (Phiên bản hệ thống mã)	ST	Nếu áp dụng được, thì đây là một bộ mô tả phiên bản được xác định riêng cho hệ thống mã đang xét.
displayName (Tên hiển thị)	ST	Tên hoặc tiêu đề của mã, hệ thống gửi trình bày giá trị mã dưới tên đó cho người sử dụng.
originalText (Văn bản gốc)	ED	Văn bản hoặc cụm từ dùng làm cơ sở để mã hóa

```
type PhysicalQuantityRepresentation alias PQR specializes CV {
  REAL  value;
};
```

2.32.1 Giá trị: REAL

Định nghĩa: Độ lớn của giá trị đo xét theo đơn vị do mã này xác định.

2.32.2 Mã: ST, thừa kế từ CV

Định nghĩa: Ký hiệu mã đơn giản được hệ thống mã định nghĩa. Ví dụ, "784.0" là ký hiệu mã của mã ICD-9 "784.0" cho bệnh đau đầu.

Một giá trị không ngoại lệ có một đặc tính *mã* non-NULL với giá trị của nó là một chuỗi ký tự mà là một ký hiệu được định nghĩa bởi hệ thống mã hóa do Hệ thống mã nhận diện. Ngược lại, một giá trị không có giá trị cho đặc tính mã, hoặc có giá trị không phải từ hệ thống mã hóa được trích dẫn, là một giá trị ngoại lệ (NULL thuộc sắc thái *khác*)

```
invariant(CD x)
    where x.nonNull {
  x.code.nonNull;
};
```

2.32.3 Hệ thống mã: UID, thừa kế từ CV

Định nghĩa: Xác định hệ thống mã định nghĩa mã.

Hệ thống mã phải được gọi bằng một UID, để có một tham chiếu rõ ràng tới các mã HL7 tiêu chuẩn, các hệ thống mã tiêu chuẩn khác, cũng như các mã địa phương. HL7 phải gán một UID cho từng bảng mã của nó cũng như cho các hệ thống mã hóa tiêu chuẩn bên ngoài được sử dụng cùng HL7. Các trang địa phương phải sử dụng Định danh đối tượng ISO (OID) của chúng để xây dựng một định danh hệ thống mã hóa địa phương duy nhất toàn cầu

Dưới nhánh của HL7, 2.16.840.1.113883, các nhánh phụ 5 và 6 lần lượt chứa định danh tiêu chuẩn HL7 và định danh hệ thống mã bên ngoài. Ủy ban Kỹ thuật Từ vựng HL7 duy trì hai nhánh phụ này.

Một giá trị không ngoại lệ (nghĩa là một giá trị có đặc tính mã non-null) có một *Hệ thống mã* non-NULL xác định hệ thống khái niệm định nghĩa mã. Nói cách khác, bất cứ khi nào có mã thì cũng có một hệ thống mã.

LƯU Ý: Mặc dù mọi giá trị non-NULL đều có một hệ thống mã được định nghĩa, trong một vài trường hợp, biểu diễn của ITS cho giá trị không cần đề cập rõ ràng tới hệ thống mã. Ví dụ, khi bối cảnh quy định bắt buộc có một và chỉ một hệ thống mã được sử dụng, việc chỉ rõ hệ thống mã sẽ trở nên thừa thãi. Tuy nhiên, trong trường hợp đó *Hệ thống mã* chấp nhận giá trị mặc định của bối cảnh cụ thể đó và có giá trị là không NULL.

```
invariant(CD x)
    where x.code.nonNull {
  x.codeSystem.nonNull;
};
```

Một ngoại lệ của nullFlavor (trạng thái null) *khác* cho biết không thể mã hóa một khái niệm trong hệ thống mã hóa được xác định. Vì vậy, đối với các ngoại lệ về mã hóa này, hệ thống mã không chứa khái niệm thích hợp phải được cung cấp trong *Hệ thống mã.*

Một vài miền mã được chuẩn bị điều kiện để chứa một phần của bất kỳ hệ thống mã hóa địa phương phù hợp nào không chỉ đơn thuần diễn giải hệ thống mã hóa tiêu chuẩn (*được mã hóa với khả năng mở rộng*, kiểu dữ liệu CWE.) Nếu một trường dữ liệu đủ điều kiện CWE thực sự chứa một mã địa phương như vậy, hệ thống mã hóa phải xác định hệ thống mã hóa địa phương mà từ đó mã địa phương được lấy. Tuy nhiên, đối với các miền CWE mã địa phương là một thành viên hợp lệ của miền, để các mã địa phương trong miền CWE không tạo ra lỗi cũng không tạo ra một giá trị ngoại lệ (NULL/khác) theo ý nghĩa của đặc tả này.

LƯU Ý: Mặc dù mọi giá trị non-NULL đều có một hệ thống mã được định nghĩa, trong một vài trường hợp, biểu diễn của ITS cho giá trị không cần đề cập rõ ràng tới hệ thống mã. Ví dụ, khi bối cảnh quy định bắt buộc có một và chỉ một hệ thống mã được sử dụng, việc chỉ rõ hệ thống mã sẽ trở nên thừa thãi. Tuy nhiên, trong trường hợp đó *Hệ thống mã* chấp nhận giá trị mặc định của bối cảnh cụ thể đó và có giá trị là không NULL

```
invariant(CD x)
    where x.other {
  x.code.other;
  x.codeSystem.nonNull;
};
```

2.32.4 Tên hệ thống mã: ST, thừa kế từ CV

Định nghĩa: Tên thông thường của hệ thống mã hóa.

Tên hệ thống mã không có giá trị tính toán. Mục đích của tên hệ thống mã là để hỗ trợ một người diễn dịch không được hỗ trợ của một giá trị mã trong việc diễn dịch Hệ thống mã. Chúng tôi đề xuất — tuy nhiên không thực sự bắt buộc — rằng ITS hỗ trợ *Tên hệ thống mã* để chú giải UID sao cho con người có thể hiểu được.

Các hệ thống HL7 không được dựa về mặt chức năng vào *codeSystemName.codeSystemName* không bao giờ có thể thay đổi ý nghĩa của Hệ thống mã và không thể tồn tại mà không có Hệ thống mã.

```
invariant(CD x) {
  x.codeSystemName.nonNull.implies(x.codeSystem.nonNull);
};
```

2.32.5 Phiên bản hệ thống mã: ST

Định nghĩa: Nếu áp dụng được, thì đây là một bộ mô tả phiên bản được xác định riêng cho hệ thống mã đang được đề cập đến.

HL7 phải xác định các chuỗi ký tự phiên bản này được hình thành như thế nào cho từng hệ thống mã bên ngoài. Nếu HL7 chưa xác định các chuỗi phiên bản được

hình thành như thế nào cho một hệ thống mã hóa cụ thể, thì chỉ định phiên bản không có ý nghĩa được định nghĩa cho hệ thống mã hóa đó.

Các phiên bản khác nhau của một hệ thống mã phải tương thích. Bất cứ khi nào một hệ thống mã thay đổi theo hướng không tương thích, nó sẽ tạo ra một hệ thống mã mới, không đơn giản là một phiên bản khác, bất kể nhà xuất bản từ vựng gọi nó là gì.

Ví dụ, nhà xuất bản của ICD-9 và ICD-10 gọi các hệ thống mã này lần lượt là "hiệu chỉnh 9" và "hiệu chỉnh 10". Tuy nhiên, ICD-10 là thiết kế lại hoàn toàn của mã ICD, chứ không phải một phiên bản tương thích ngược. Vì vậy, để phục vụ mục đích của đặc tả kiểu dữ liệu này, ICD-9 và ICD-10 là các hệ thống mã khác biệt, chứ không chỉ là các phiên bản khác biệt. Ngược lại, khi LOINC cập nhật từ hiệu chỉnh "1.0j" lên "1.0k", HL7 sẽ coi đây chỉ là một phiên bản khác của LOINC, do các hiệu chỉnh LOINC có tính tương thích ngược.

```
invariant(CD x) {
  x.codeSystemVersion.nonNull.implies(x.codeSystem.nonNull);
};
```

2.32.6 Tên hiển thị: ST, thừa kế từ CV

Định nghĩa: Tên hoặc tiêu đề của mã, hệ thống gửi trình bày giá trị mã dưới tên đó cho người sử dụng.

Tên hiển thị được đưa vào vừa như một hình thức thiện chí với một người diễn dịch không được hỗ trợ của một giá trị mã, vừa như một dạng văn bản cho tên được sử dụng để hiển thị khái niệm với người sử dụng. Tên hiển thị không có ý nghĩa chức năng; nó không bao giờ có thể tồn tại mà không có mã; và nó không bao giờ có thể thay đổi ý nghĩa của mã.

LƯU Ý: HL7 cung cấp "tên in ấn" trong các miền từ vựng được định nghĩa trước của mình. Các giá trị này phù hợp để sử dụng trong Tên hiển thị.

LƯU Ý: Tên hiển thị không thể thay đổi ý nghĩa của giá trị mã. Vì vậy, không nên trình bày tên hiển thị với người sử dụng trên một hệ thống ứng dụng tiếp nhận mà không biết chắc rằng tên hiển thị đó biểu diễn một cách thỏa đáng khái niệm được giá trị mã nhắc tới. Việc thông tin liên lạc không được chỉ dựa vào tên hiển thị. Mục đích chính của tên hiển thị là để hỗ trợ việc chỉnh lỗi của các đơn vị dữ liệu giao thức HL7 (ví dụ, bản tin).

```
invariant(CD x) {
  x.displayName.nonNull.implies(x.code.nonNull);
};
```

2.32.7 Văn bản gốc: ED, thừa kế từ CV

Định nghĩa: Văn bản hoặc cụm từ dùng làm cơ sở để mã hóa

Văn bản gốc tồn tại trong một kịch bản khi người khởi tạo thông tin không gán mã, nhưng sau này mã được gán bởi một người tạo mã (mã hóa sau). Trong quá trình tạo ra một bộ mô tả khái niệm, văn bản gốc do đó có thể tồn tại mà không có mã.

LƯU Ý: Mặc dù mã hóa sau thường được thực hiện từ thông tin văn bản tự do, như tài liệu, hình quét (scan) hay chính tả, dữ liệu đa phương tiện rõ ràng không được chấp nhận là văn bản gốc. Ngoài ra, đặc tính văn bản gốc không phải là một liên kết tới toàn bộ tài liệu nguồn. Liên kết giữa các sản phẩm thông tin y tế khác nhau (ví dụ, tài liệu và kết quả được mã hóa) nằm ngoài phạm vi của đặc tả này và được duy trì ở phần khác trong các tiêu chuẩn HL7. Văn bản gốc là một đoạn trích các thông tin liên quan trong các nguồn ban đầu, chứ không phải một con trỏ văn bản hay tái tạo chính xác. Vì vậy, văn bản gốc được biểu diễn dưới dạng văn bản đơn thuần.

Các giá trị của kiểu có thể có đặc tính văn bản gốc non-NULL dù cho nó có mã NULL. Bất kỳ giá trị nào có mã NULL đều biểu thị một ngoại lệ mã hóa. Trong trường hợp này, *Văn bản gốc* là tên hay mô tả của khái niệm không được mã hóa. Các giá trị ngoại lệ như vậy cũng có thể chứa các chuyển ngữ. Các chuyển ngữ này mã hóa trực tiếp khái niệm được mô tả trong *Văn bản gốc*.

Một giá trị *CD* có thể được chuyển đổi giảm thành một giá trị ST chỉ biểu diễn *Văn bản gốc* của giá trị *CD*.

```
invariant(CD x)
    where x.originalText.nonNull {
  ((ST)x).equal(x.originalText);
};
```

2.33 Số tiền (MO) chuyên biệt hóa QTY

Định nghĩa: Một giá trị *MO* là một đại lượng biểu diễn số tiền trong một loại tiền tệ nào đó. Loại tiền tệ là đơn vị trong đó số tiền được thể hiện tại các vùng kinh tế khác nhau. Trong khi số tiền là loại đại lượng duy nhất (tiền), thì tỷ giá hối đoái giữa các đơn vị tiền khác nhau lại biến thiên. Đây là sự khác biệt chính giữa kiểu dữ liệu PQ và *MO*, và là lý do tại sao đơn vị tiền tệ lại không phải là đơn vị vật lý.

Bảng 34. Tóm tắt đặc tính của Số tiền

Tên	Kiểu	Mô tả
value (giá trị)	REAL	Độ lớn của *MO* xét theo loại tiền tệ.
currency (tiền tệ)	CS	Đơn vị tiền tệ như được định nghĩa trong ISO 4217.

Tên	Kiểu	Mô tả
diffType (Kiểu hiệu)	TYPE	Kiểu của hiệu của 2 giá trị của một kiểu dữ liệu QTY cụ thể.

```
type MonetaryAmount alias MO specializes QTY {
        REAL  value;
        CS    currency;
        MO    negated;
        MO    times(REAL x);
  literal ST;
};
```

2.33.1 Giá trị: REAL

Định nghĩa: Độ lớn của giá trị *MO* xét theo loại tiền tệ

LƯU Ý: Các giá trị *MO* thường chính xác tới 0.01 (một cent, penny, paisa, v.v.) Đối với số lượng lớn, điều quan trọng là không lưu các giá trị *MO* trong các thanh ghi dấu phẩy động, vì có thể làm giảm độ chính xác. Tuy nhiên, đặc tả này không định nghĩa việc lưu trữ trong REAL là các số dấu phẩy tĩnh hay động.

REAL.precision là độ chính xác của việc biểu diễn thập phân, không phải độ chính xác của giá trị. Kiểu dữ liệu REAL không có khái niệm về tính không chắc chắn hay tính đúng đắn. Ví dụ, "1,99 USD" (độ chính xác 3) nhân 7 bằng "13,93 USD" (độ chính xác 4) và không nên làm tròn thành "13,9" để giữ cho độ chính xác không đổi.

2.33.2 Tiền tệ: CS

Định nghĩa: Đơn vị tiền tệ như được định nghĩa trong ISO 4217

Bảng 35. Miền tiền tệ

mã	tên	định nghĩa
ARS	Argentine Peso	Peso Argentina, đơn vị tiền tệ của Argentina
AUD	Australian Dollar (Đô-la Úc)	Đô-la Úc, đơn vị tiền tệ của Australia
BRL	Brazilian Real	Real Braxin, đơn vị tiền tệ của Braxin

mã	tên	định nghĩa
CAD	Canadian Dollar (Đô-la Canada)	Đô-la Canada, đơn vị tiền tệ của Canada
CHF	Swiss Franc (Franc Thụy Sỹ)	Franc Thụy Sỹ, đơn vị tiền tệ của Thụy Sỹ
CLF	Unidades de Formento	Unidades de Formento, đơn vị tiền tệ của Chi-lê
CNY	Yuan Renminbi (Nhân dân tệ)	Nhân dân tệ, đơn vị tiền tệ của Trung Quốc
DEM	Deutsche Mark (Mark Đức)	Mark Đức, đơn vị tiền tệ của Đức
ESP	Spanish Peseta (Peseta Tây Ban Nha)	Peseta Tây Ban Nha, đơn vị tiền tệ của Tây Ban Nha
EUR	Euro	Euro, đơn vị tiền tệ của Liên minh châu Âu
FIM	Markka	Markka, đơn vị tiền tệ của Phần Lan
FRF	French Franc (Franc Pháp)	Franc Pháp, đơn vị tiền tệ của Pháp
GBP	Pound Sterling (Bảng Anh)	Bảng Anh, đơn vị tiền tệ của Vương quốc Anh
ILS	Shekel	Shekel, đơn vị tiền tệ của Israel
INR	Indian Rupee (Rupi Ấn Độ)	Rupi Ấn Độ, đơn vị tiền tệ của Ấn Độ

mã	tên	định nghĩa
JPY	Yen (Yên)	Yên, đơn vị tiền tệ của Nhật Bản
KRW	Won	Won, đơn vị tiền tệ của Hàn Quốc
MXN	Mexican Nuevo Peso (Nuevo Peso Mêhicô)	Nuevo Peso Mêhicô, đơn vị tiền tệ của Mêhicô
NLG	Netherlands Guilder (Guilder Hà Lan)	Guilder Hà Lan, đơn vị tiền tệ của Hà Lan
NZD	New Zealand Dollar (Đô-la New Zealand)	Đô-la New Zealand, đơn vị tiền tệ của New Zealand
PHP	Philippine Peso (Peso Philippine)	Peso Philippine, đơn vị tiền tệ của Philippine
RUR	Russian Ruble (Rúp Nga)	Rúp Nga, đơn vị tiền tệ của Liên bang Nga
THB	Baht (Bạt)	Bạt, đơn vị tiền tệ của Thái Lan
TRL	Lira (Lia)	Lia, đơn vị tiền tệ của Thổ Nhĩ Kỳ
TWD	Taiwan Dollar (Đô-la Đài Loan)	Đô-la Đài Loan, đơn vị tiền tệ của Đài Loan
USD	US Dollar (Đô-la Mỹ)	Đô-la Mỹ, đơn vị tiền tệ của Hoa Kỳ
ZAR	Rand	Ran, đơn vị tiền tệ của Nam Phi

mã	tên	định nghĩa
	(Ran)	

Bảng này chỉ thể hiện một tập con đại diện của các mã được ISO 4217 định nghĩa. Tất cả các mã từ ISO 4127 đều là các giá trị hợp lệ cho thuộc tính này.

2.33.3 Tương đương: BL, thừa kế từ ANY

Hai giá trị *MO* là bình đẳng nếu cả giá trị và loại tiền tệ là bình đẳng.

```
invariant(MO x, y)
    where x.nonNull.and(y.nonNull) {
  x.equal(y).equal(x.value.equal(y.value)
        .and(x.unit.equal(y.unit)));
};
```

2.33.4 Tính có thể so sánh: BL, thừa kế từ QTY

Hai giá trị *MO* có thể được so sánh với nhau (và có thứ tự và hiệu) nếu loại tiền tệ của chúng là bình đẳng

Nếu loại tiền tệ của chúng không giống nhau, thì không thể so sánh giá trị. Chuyển đổi giữa các loại tiền tệ nằm ngoài phạm vi của đặc tả này. Trong thực tiễn, tỷ giá hối đoái là rất khác nhau không chỉ trong các khoảng thời gian dài và ngắn, mà còn phụ thuộc vào địa điểm và khả năng tiếp cận với thị trường thương mại tiền tệ.

```
invariant(MO x, y)
    where x.nonNull.and(y.nonNull) {
  x.compares.equal(x.currency.equal(y.currency));
};
```

2.33.5 Kiểu dữ liệu hiệu: TYPE, thừa kế từ QTY

```
invariant(INT x) {
  x.diffType.implies(MO);
};
```

Hiệu của 2 giá trị *MO* là một *MO* khác.

2.33.6 Phép cộng: MO

Định nghĩa: Có thể cộng hai giá trị *MO* nếu loại tiền tệ của chúng là bình đẳng.

```
invariant (MO x, y)
    where x.currency.equal(y.currency) {
  x.plus(y).currency.equal(x.currency);
```

```
    x.plus(y).value.equal(x.value.plus(y.value));
};
```

2.33.7 Phép nhân số thực: MO

Định nghĩa: Phép nhân với một giá trị REAL tạo ra một đại lượng có chia tỷ lệ. Một đại lượng có chia tỷ lệ có thể so sánh với đại lượng gốc của nó.

```
invariant(MO x; REAL r)
    where x.nonNull.and(r.nonNull) {
  x.times(r).value.equal(x.value.times(r));
  x.times(r).currency.equal(x.currency);
};
```

2.33.8 Dạng chữ

Dạng chữ cho một giá trị *MO* bao gồm một chuỗi ký tự mã loại tiền tệ, khoảng trắng tùy chọn, và một số lượng dạng chữ REAL.

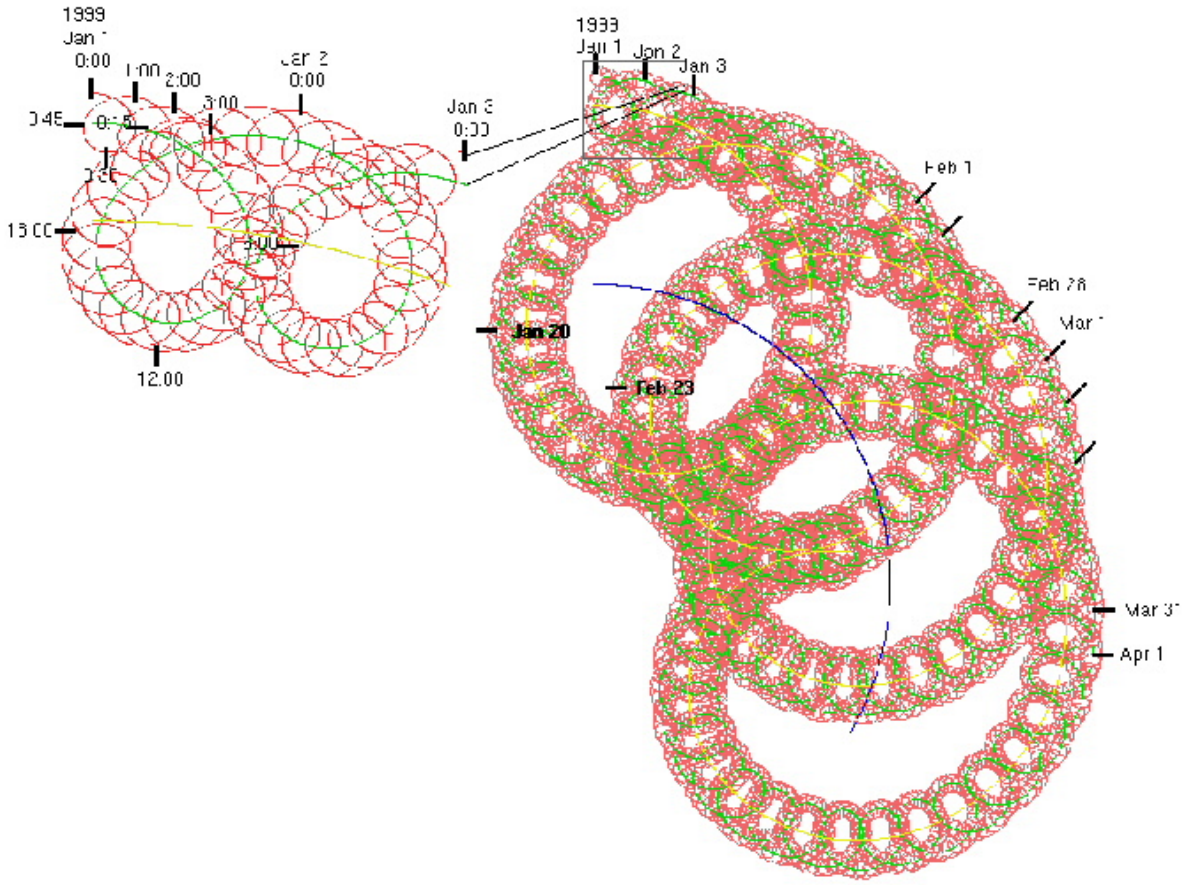
```
MO.literal ST {
  MO          : currency value  { $.currency.equal($1); }
                                $.value.equal($2);
  CS  currency : ST             { $.currency.value.equal($1);
                                $.currency.codeSystem
                                .equal(2.16.840.1.113883.6.9); }
  REAL value   : REAL           { $.value.equal($1); }
};
```

Ví dụ, “USD 189.95” là một biểu diễn cho 189.95 USD

2.34 Lịch (CAL) chuyên biệt hóa SET<CLCY>

Định nghĩa: Lịch là khái niệm đo thời gian theo nhiều chu kỳ. Các chu kỳ đó là năm, tháng, ngày, giờ, phút, giây, và tuần. Một vài trong số các chu kỳ này được đồng bộ hóa và một số khác thì không (ví dụ, tuần và tháng không được đồng bộ hóa).

Sau khi "lăn trục thời gian" vào các chu kỳ này, lịch biểu diễn một thời điểm như một chuỗi các lần đếm số nguyên các chu kỳ, ví dụ, cho năm, tháng, ngày, giờ, v.v. Lịch bắt nguồn từ một điểm bắt đầu truyền thống nào đó, được gọi là "thời đại".



Lịch "lăn" trục thời gian vào một mặt cuốn phức tạp dựa theo các giai đoạn của lịch là năm (xanh dương), tháng (vàng), ngày (xanh lá), giờ (đỏ), v.v. Không cần căn chỉnh các chu kỳ, ví dụ, tuần (không được thể hiện) không được căn chỉnh với tháng.[46]

Lịch được định nghĩa là một tập hợp các chu kỳ lịch, và có một tên và một mã. Phần đầu của Lịch là Chu kỳ lịch lớn nhất xuất hiện nhiều nhất trong biểu thức lịch. Thời đại là khởi đầu của lịch đó, nghĩa là thời điểm khi tất cả các chu kỳ lịch là 0.

```
private type Calendar alias CAL specializes SET<CLCY>  {
  CS    name;
  CLCY  head;
  TS    epoch;
};

invariant(CAL c)
    where c.nonNull {
  c.name.nonNull;
  c.contains(c.head);
};
```

Định nghĩa lịch có thể được thể hiện như trong Bảng 36 cho lịch Dương hiện đại. Bảng định nghĩa lịch liệt kê một chu kỳ lịch trong mỗi hàng. Các đơn vị lịch là

phụ thuộc lẫn nhau và được định nghĩa trong cột giá trị. Cột chuỗi thứ tự thể hiện mối quan hệ thông qua đặc tính tiếp theo. Các cột khác như trong định nghĩa chu kỳ lịch chính thức.[47]

Bảng 36. Miền Chu kỳ lịch

tên	mã 1	mã 2	bộ đếm	chữ số	bắt đầu	điều kiện
năm	Y	CY	1	4	0	
tháng của năm	M	MY	2	2	1	
tháng (liên tục)		CM			0	
tuần (liên tục)	W	CW			0	
tuần của năm		WY		2	1	
ngày của tháng	D	DM	3	2	1	
ngày (liên tục)		CD			0	
ngày của năm		DY		3	1	
ngày của tuần (bắt đầu từ thứ Hai)	J	DW		1	1	
giờ của ngày	H	HD	4	2	0	
giờ (liên tục)		CH			0	
phút của giờ	N	NH	5	2	0	
phút (liên tục)		CN			0	
giây của phút	S	SN	6	2	0	

tên	mã 1	mã 2	bộ đếm	chữ số	bắt đầu	điều kiện
giây (liên tục)		CS			0	

2.35 Chu kỳ lịch (CLCY) chuyên biệt hóa ANY

Định nghĩa: Một chu kỳ lịch định nghĩa một nhóm các chữ số thập phân trong biểu thức lịch. Ví dụ về chu kỳ lịch là năm, tháng, ngày, giờ, phút, giây, và tuần.

Một chu kỳ lịch có một tên và hai mã, một mã một chữ cái và một mã hai chữ cái. Đặc tính *n chữ số* là số chữ số thập phân có trong biểu thức lịch. Đặc tính *bắt đầu* xác định lúc bắt đầu đếm (nghĩa là, tại 0 hay 1). Đặc tính *tiếp theo* là chu kỳ thấp hơn tiếp theo trong thứ tự của biểu thức lịch. Đặc tính lớn nhất (*t*) là số chu kỳ lớn nhất tại thời gian *t* (lớn nhất phụ thuộc vào thời gian *t* để giải thích cho năm nhuận và giây nhuận). Đặc tính giá-trị (*t*) là số nguyên chu kỳ được thể hiện trong biểu thức lịch của thời gian *t*. Đặc tính tổng(*t, n*) là tổng các chu kỳ lịch *n* cộng với thời gian *t*.

```
private type CalendarCycle alias CALCY specializes ANY {
  CE     name;
  INT    ndigits;
  INT    start;
  CALCY  next;
  INT    max(TS);
  TS     sum(TS t, REAL r);
  INT    value(TS t);
};

invariant(CALCY c)
    where c.nonNull {
  c.name.nonNull;
  c.start.equal(0).or(c.start.equal(1));
  c.digits.greaterThan(0);
};
```

2.36 Thời điểm (TS) chuyên biệt hóa QTY

Định nghĩa: Một đại lượng xác định một điểm trên trục thời gian tự nhiên. Thời điểm thường được biểu diễn ở dạng một biểu thức lịch.

Tuy nhiên, về mặt ngữ nghĩa, thời gian không phụ thuộc lịch và được mô tả tốt nhất bằng mối quan hệ của nó với thời gian trôi qua (được đo như một đại lượng vật lý

bằng đơn vị thời gian). Một giá trị *TS* cộng thời gian trôi qua cho ra một *TS* khác. Ngược lại, một giá trị *TS* trừ một *TS* khác cho ra thời gian trôi qua.

Do không ai biết thời gian bắt đầu khi nào, một giá trị *TS* được khái niệm hóa là lượng thời gian đã trôi qua từ một điểm 0 tùy ý nào đó, được gọi là một thời đại. Vì không có một điểm 0 tuyệt đối trên trục thời gian; thời gian tự nhiên là một đại lượng hiệu-thang tỷ lệ, trong đó chỉ có các hiệu được định nghĩa mà không có tỷ lệ. (Ví dụ, không có *TS* nào là — nói một cách tuyệt đối — "muộn gấp hai lần" một *TS* khác).

Với một điểm 0 tùy ý nào đó cho trước, có thể biểu diễn bất kỳ thời điểm nào như một thời gian trôi qua được đo từ khoảng chênh thời gian đó. Một điểm 0 tùy ý như vậy được gọi là một thời đại. Dạng thời đại-khoảng chênh thời gian này được sử dụng như một biểu diễn ngữ nghĩa ở đây, mà không ngụ ý rằng một hệ thống bất kỳ sẽ phải triển khai *TS* theo cách đó. Các hệ thống không cần tính khoảng cách giữa các giá trị *TS* sẽ không cần bất kỳ biểu diễn nào khác ngoài dạng chữ biểu thức lịch.

```
type PointInTime alias TS specializes QTY {
        PQ   offset;
        CS   calendar;
        INT  precision;
        PQ   timezone;
        BL   equal(ANY x);
 literal  ST;
};
```

2.36.1 Khoảng chênh thời gian từ thời đại: PQ

Định nghĩa: Thời gian trôi qua kể từ thời đại không đổi bất kỳ, được đo như một đại lượng vật lý theo đơn vị thời gian (nghĩa là, có thể so sánh với một giây).

```
invariant(TS x)
     where x.nonNull {
  x.offset.compares(1 s);
};
```

Đặc tả này không cần định nghĩa thời đại chính tắc; ngữ nghĩa cho một thời đại bất kỳ là như nhau, miễn rằng thời đại là không đổi.

LƯU Ý: *khoảng chênh thời gian* có thể được coi như một đặc tính thuần ngữ nghĩa, tức là không được biểu diễn bằng cách nào khác ngoài dạng chữ lịch. Tuy nhiên, ITS cũng có thể chọn định nghĩa một thời đại không đổi và biểu diễn các giá trị *TS* như khoảng chênh thời gian trôi qua liên quan đến thời đại đó. Tuy nhiên, ITS sử dụng biểu diễn thời đại-khoảng chênh thời gian sẽ vẫn cần phải thông tin về mã lịch và độ chính xác của một biểu diễn lịch một khi các lịch khác được hỗ trợ.

2.36.2 Tương đương: BL, thừa kế từ QTY

Hai giá trị *TS* là bình đẳng khi và chỉ khi các khoảng chênh thời gian của chúng (liên quan đến cùng một thời đại) là bình đẳng.

```
invariant(TS x, y)
    where x.nonNull.and(y.nonNull) {
  x.equal(y).equal(x.offset.equal(y.offset));
};
```

2.36.3 Lịch: CS

Định nghĩa: Một mã chỉ ra lịch được sử dụng trong biểu diễn dạng chữ của kiểu dữ liệu *TS* này.[48]

Bảng 37. Miền Lịch

mã	tên	định nghĩa
GREG	Gregorian (Lịch Dương)	Lịch Dương là lịch thông dụng ở hầu hết các quốc gia chịu ảnh hưởng của Thiên chúa giáo từ khoảng năm 1582. Lịch này thay thế lịch Julius.

Mục đích của đặc tính này chủ yếu là để chuyển tải một cách trung thành nội dung đã được nhập hay nhìn thấy bởi người dùng trong hệ thống đã hình thành một giá trị *TS* như vậy. *lịch* cũng đưa ra lời khuyên cho bất kỳ hệ thống nào muốn biểu diễn một giá trị *TS* thành dạng chữ về việc nên dùng lịch nào. Tuy nhiên, đây chỉ là lời khuyên; bất kỳ hệ thống nào biểu diễn các giá trị *TS* cho người sử dụng đều có thể chọn lịch và dạng chữ mà người dùng của nó yêu cầu thay vì lịch được đề cập trong *lịch*. Vì vậy, *lịch* không phải là không đổi trong giao tiếp giữa các hệ thống, lịch không phải là một phần trong kiểm tra bình đẳng.

Để phục vụ mục đích định nghĩa mối quan hệ giữa biểu thức lịch và dạng thời đại-khoảng chênh thời gian, hai kiểu dữ liệu cá nhân, CAL và CLCY được định nghĩa. Các kiểu dữ liệu lịch này tồn tại chỉ để định nghĩa đặc tả này. Không thể sử dụng các kiểu dữ liệu cá nhân này bên ngoài đặc tả này.

2.36.4 Độ chính xác của dạng chữ lịch: INT

Định nghĩa: Số chữ số quan trọng của biểu diễn biểu thức lịch.

độ chính xác được định nghĩa chính thức dựa trên dạng chữ

độ chính xác chỉ là độ chính xác của biểu diễn chữ số thập phân, *không phải là tính đúng đắn của giá trị TS.*

Mục đích của *độ chính xác* là để có thể nắm bắt một cách trung thực toàn bộ thông tin được trình bày cho con người trong một biểu thức lịch. Số lượng chữ số được

trình bày chuyển tải thông tin về tính không chắc chắn (nghĩa là, độ chính xác và tính đúng đắn) của một giá trị *TS* được đo đạc.

LƯU Ý: Độ chính xác của một biểu diễn không phụ thuộc vào tính không chắc chắn (độ chính xác và tính đúng đắn) của kết quả đo đạc. Nếu tính không chắc chắn của một kết quả đo đạc là quan trọng, nên định rõ các giá trị không chắc chắn là PPD

độ chính xác phụ thuộc vào lịch. Điều này có nghĩa là một *độ chính xác* cụ thể liên quan tới một lịch sẽ không giữ nguyên trong một lịch khác với các giai đoạn khác.

Ví dụ, "20000403" có 8 chữ số quan trọng *trong biểu diễn*, nhưng *tính không chắc chắn của giá trị này* có thể nằm ở bất kỳ chữ số nào được thể hiện hay không được thể hiện, nghĩa là, tính không chắc chắn có thể ở ngày, tuần, hay giờ. Lưu ý rằng các biểu diễn bên ngoài *nên* điều chỉnh độ chính xác của phép biểu diễn với tính không chắc chắn của giá trị. Tuy nhiên, do độ chính xác trong chuỗi chữ số phụ thuộc vào lịch và lẻ tới các giai đoạn lịch, tính không chắc chắn có thể không nằm trong mạng lưới đó (ví dụ, 2000040317 là một biểu diễn đầy đủ cho các giá trị giữa 2000040305 và 2000040405.)

LƯU Ý: ITS dựa trên ký tự không cần biểu diễn *độ chính xác* như một thuộc tính rõ ràng nếu các giá trị *TS* được biểu diễn như các biểu thức lịch dạng chữ. Việc biểu diễn giá trị *TS* không được tạo ra nhiều hay ít các chữ số quan trọng hơn so với trong giá trị ban đầu. Có thể kiểm tra mức phù hợp thông qua quy trình mã hóa — giải mã — mã hóa.

2.36.5 Khoảng chênh thời gian giữa các múi giờ: PQ

Định nghĩa: Mức chênh lệch giữa giờ địa phương trong múi giờ đó và Thời gian phối hợp toàn cầu (UTC, trước đây được gọi là giờ GMT). Múi giờ là một giá trị PQ theo đơn vị thời gian (nghĩa là, có thể so sánh với một giây.) Một giá trị múi giờ bằng 0 xác định UTC. Giá trị múi giờ không cho phép kết luận về kinh độ địa lý hay tên múi giờ thông thường.

Ví dụ, 200005121800-0500 có thể là giờ chuẩn miền đông (EST) ở thành phố Indianapolis, bang Indiana, hay giờ tiết kiệm ánh sáng ban ngày (CDT) ở thành phố Decatur, bang Illinoise. Ngoài ra, ở các quốc gia khác với vĩ độ khác, múi giờ có thể được đặt tên khác.

```
invariant(TS x, y)
    where x.nonNull.and(y.nonNull) {
  x.timezone.compares(1 s);
};
```

Khi *múi giờ* có giá trị NULL (không biết), "giờ địa phương" được sử dụng. Tuy nhiên, "giờ địa phương" luôn chỉ có tính địa phương với một nơi nào đó và không có kiến thức về nơi đó, múi giờ là không được biết. Vì vậy, không thể chuyển đổi giờ địa

phương thành UTC. Nên chỉ rõ *múi giờ* cho tất cả các giá trị *TS* nhằm tránh tổn thất đáng kể về độ chính xác khi so sánh các *TS*. Hiệu của hai giờ địa phương trong đó địa phương là không biết có sai số là ±12 giờ.

Trong bối cảnh dữ liệu hành chính, một vài giá trị thời gian không chứa múi giờ. Ví dụ, đối với một ngày sinh trong dữ liệu hành chính, sẽ là không chính xác khi chỉ ra *múi giờ*, do điều này có thể thực sự làm thay đổi ngày sinh khi chuyển đổi sang các múi giờ khác. Đối với các dữ liệu hành chính như vậy, múi giờ có giá trị là NULL (không áp dụng được).

2.36.6 Kiểu dữ liệu hiệu: TYPE, thừa kế từ QTY

```
invariant(TS x) {
  x.diffType.implies(PQ);
};
```

Mức chênh lệch giữa hai giá trị *TS* là một giá trị PQ theo đơn vị thời gian.

2.36.7 Phép cộng: TS, thừa kế từ QTY

Định nghĩa: Một giá trị *TS* cộng một thời gian đã trôi qua (tức là, giá trị PQ theo đơn vị thời gian) là một giá trị *TS*.

```
invariant(TS x, PQ t)
    where x.nonNull.and(t.compares(1 s)) {
  x.plus(t).offset.equal(x.offset.plus(t));
};
```

2.36.8 Hiệu: QTY, thừa kế từ QTY

Định nghĩa: Hiệu giữa hai giá trị *TS* là một thời gian đã trôi qua.

```
invariant(TS x)
    where x.nonNull {
  x.minus(y).offset.equal(
    x.offset.plus(y.offset.negated));
};
```

2.36.9 Dạng chữ

Dạng chữ giá trị *TS* là các biểu thức lịch đơn giản, như được định nghĩa bởi bảng định nghĩa lịch. Theo mặc định, lịch tây (lịch Dương) phải được sử dụng (Bảng 36).

Đối với lịch Dương mặc định, dạng chữ biểu thức lịch của đặc tả này phù hợp với tiêu chuẩn ISO 8601 hạn chế được định nghĩa trong ISO 8824 (ASN.1) theo khoảng 32 (thời gian suy rộng) và với kiểu dữ liệu TS của HL7 Phiên bản 2.

Dạng chữ biểu thức lịch là các chuỗi số nguyên được sắp xếp theo cột "bộ đếm" của Bảng 36. Các giai đoạn với số thứ tự thấp hơn đứng về bên trái các giai đoạn có số thứ tự cao hơn. Không thể có các giai đoạn không được gán số thứ tự trong biểu thức lịch cho giá trị *TS*.

Cột "chữ số" của Bảng 36 xác định chính xác số chữ số cho số bộ đếm cho giai đoạn bất kỳ.

Vì vậy, Bảng 36 xác định rằng biểu thức lịch tây bắt đầu với năm có 4 chữ số (bắt đầu đếm từ 0); tiếp đến là tháng của năm với 2 chữ số (bắt đầu đếm từ một); tiếp đến là ngày của tháng với 2 chữ số (bắt đầu từ một); tiếp đến là giờ của ngày với hai chữ số (bắt đầu từ 0); và cứ như thế. Ví dụ, "200004010315" là biểu thức hợp lệ cho ngày 1 tháng Tư năm 2000, 3:15 sáng.

Một biểu thức lịch có thể có độ chính xác biến thiên, khi bỏ qua các phần từ phía phải.

Ví dụ, "20000401" chỉ chính xác về ngày của tháng.

Giai đoạn lịch được định nghĩa ít nhất (tức là giây) có thể được viết như một giá trị REAL, với số lượng các chữ số nguyên được chỉ rõ, tiếp sau bởi dấu thập phân và một số bất kỳ các chữ số phân số.

Ví dụ, "20000401031520.34" có nghĩa là ngày 1 tháng tư năm 2000, 3:15 và 20.34 giây.

Khi có các lịch khác được sử dụng trong tương lai, một tiền tố "GREG:" có thể được đặt trước biểu thức lịch tây (Dương) để phân biệt rõ ràng với các lịch khác. Mỗi lịch phải có một tiền tố riêng. Tuy nhiên, lịch tây là mặc định nếu không có tiền tố nào xuất hiện.

Trong lịch Dương hiện đại (và tất cả các lịch trong đó thời gian của ngày dựa trên UTC), biểu thức lịch có thể chứa một hậu tố múi giờ. Hậu tố múi giờ bắt đầu bằng một dấu cộng (+) hay trừ (-) tiếp sau bởi các chữ số cho các chu kỳ giờ và phút. UTC được đặt tại khoảng chênh thời gian "+00" hoặc "-00"; hậu tố "Z" của tiêu chuẩn ISO 8601 và ISO 8824 cho UTC là không được phép.

```
TS.literal ST {
  TS   : cal timestamp($1)          { $.equal($2); }
       | timestamp(GREG)            { $.equal($1); };

  TS timestamp(Calendar C)
       : cycles(C.head, C.epoch) zone(C)  {$.equal($1.minus($2));
                                  $.timezone.equal($2); }
       | cycles(C.head, C.epoch)          { $.equal($1);
```

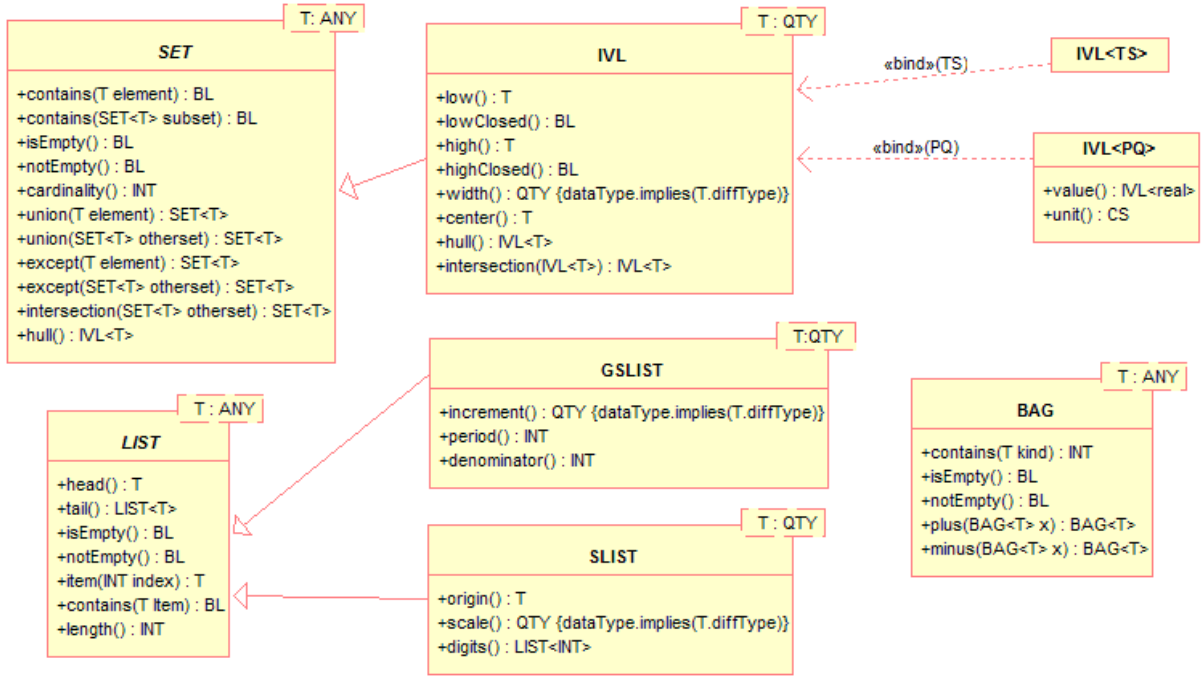
```
                                $.timezone.unknown; };
    Calendar cal
        : /[a-zA-Z_][a-zA-Z0-9_]*:/          { $.equal($1); };

    TS cycles(CalendarCycle c, TS t)
        : cycle(c, t) cycles(c.next, $1)     {$.equal($2); }
        | cycle(c, t) "." REAL.fractional    {$.equal(c.sum($1, $3));
                $.precision.equal(t.precision.plus($3.precision)); }
        | cycle(c, t)                        { $.equal($1); };

    TS cycle(CalendarCycle c, TS t)
        : /[0-9]{c.ndigits}/     { $.equal(c.sum(t, $1));
                                   $.precision.equal(t.precision.plus(c.ndigits)); };

    PQ zone(Calendar C)
        : "+" cycles(C.zonehead, C.epoch){$.equal($2.minus(C.epoch)); }
        | "-" cycles(C.zonehead, C.epoch){$.equal(C.epoch.minus($2)); };
};
```

3. Kiểu dữ liệu Tập hợp (Collection) chung



Mục này định nghĩa các kiểu dữ liệu có thể "thu thập" các giá trị dữ liệu khác, Tập hợp, Chuỗi, Bộ sưu tập và Khoảng.[49] Các kiểu bộ sưu tập này được định nghĩa là

các kiểu (tham số) chung. Khái niệm kiểu chung được mô tả trong Kiểu dữ liệu chung (§ 1.9.5).

3.1 Tập hợp (SET) chuyên biệt hóa ANY

Định nghĩa: Một giá trị chứa các giá trị riêng biệt khác không theo một thứ tự cụ thể nào.

```
template<ANY T>
type Set<T> alias SET<T> specializes ANY {
        BL      contains(T element);
        BL      isEmpty;
        BL      notEmpty;
        BL      contains(SET<T> subset);
        INT     cardinality;
        SET<T>  union(SET<T> otherset);
        SET<T>  union(T element);
        SET<T>  except(T element);
        SET<T>  except(SET<T> otherset);
        SET<T>  intersection(SET<T> otherset);
 literal    ST;
 promotion  SET<T>  (T x);
        IVL<T>  hull;
};
```

3.1.1 Chứa phần tử: BL

Định nghĩa: Một quan hệ giữa tập hợp với các phần tử của nó, đúng nếu giá trị đã cho là một phần tử của tập hợp.

Đây là đặc tính ngữ nghĩa nguyên thủy của một tập hợp, dựa trên đó tất cả các đặc tính khác được định nghĩa.

Một tập hợp có thể chỉ chứa các phần tử non-NULL riêng biệt. Các giá trị ngoại lệ (các giá trị NULL) không thể là phần tử của một tập hợp.

```
invariant(SET<T> s, T n)
     where s.nonNull.and(n.isNull) {
  s.contains(n).not;
};
```

3.1.2 Chứa tập con: BL

Định nghĩa: Mối quan hệ giữa một tập hợp với các tập con của nó, trong đó mỗi phần tử của tập con cũng là một phần tử của tập cha.

```
invariant(SET<T> superset, subset)
    where superset.nonNull.and(subset.nonNull)
      superset.contains(subset).equal(
    forall(T element) where subset.contains(element) {
      superset.contains(element);
      });
};
```

Điều này ngụ ý rằng tập rỗng là một tập con của mọi tập hợp bao gồm chính nó.

3.1.3 Không rỗng: BL

Định nghĩa: Một vị từ chỉ ra rằng tập hợp này có chứa phần tử.

```
invariant(SET<T> set)
    where set.nonNull {
  set.notEmpty.equal(exists(T element) {
    set.contains(element);
    });
};
```

3.1.4 Tập trống: BL

Định nghĩa: Một vị từ chỉ ra rằng tập hợp này không có phần tử (phủ định của không trống). Tập trống là một giá trị tập hợp hợp lệ, *không phải là* một giá trị ngoại lệ (NULL).

```
invariant(SET<T> set)
    where set.nonNull {
  set.isEmpty.equal(notEmpty.not);
};
```

3.1.5 Số lượng phần tử: INT

Định nghĩa: Số phần tử của một tập hợp là số phần tử riêng biệt trong tập hợp.

```
invariant(SET<T> set)
    where set.nonNull {
  exists(T element) where set.contains(element) {
    set.cardinality.equal(set.except(element)
            .cardinality.successor);
      };
};
```

Định nghĩa số phần tử là không đủ do nó không đồng quy cho các tập hợp vô hạn không đếm được (REAL, PQ, v.v.) và nó không chấm dứt đối với các tập hợp vô hạn. Thêm vào đó, định nghĩa kiểu số nguyên trong đặc tả này là không hoàn thiện đối với các trường hợp này, do nó không giải thích vô cực. Cuối cùng, giá trị số phần tử là m ột(*aleph*$_0$) của các tập hợp vô hạn đếm được (ví dụ, INT) v ới(*aleph*$_1$), số phần tử của các tập hợp không đếm được (ví dụ, REAL, PQ).

3.1.6 Hợp: SET<T>

Định nghĩa: Hợp của hai tập hợp (tập hợp thành phần) là một tập hợp trong đó mỗi phần tử trong số các phần tử của hợp cũng là một phần tử của một trong hai tập hợp thành phần.

```
invariant(SET<T> x, y, z)
    where x.nonNull.and(y.nonNull).and(z.nonNull) {
  x.union(y).equal(z).equal(forall(T e) {
    z.contains(e).equal(x.contains(e).or(y.contains(e)));
    });
};
```

3.1.7 Gộp phần tử: SET<T>

Định nghĩa: Một kết hợp của một tập hợp và một phần tử.

```
invariant(SET<T> set, singletonset, T element)
    where set.nonNull.and(element.nonNull)
          .and(singletonset.cardinality.isOne)
          .and(singletonset.contains(element)) {
  set.union(element).equal(set.union(singleton));
};
```

3.1.8 Hiệu tập hợp: SET<T>

Định nghĩa: Hiệu của tập hợp này và tập hợp trừ của nó là tập hợp chứa tất cả các phần tử của tập hợp này mà không phải là phần tử của tập hợp trừ.

```
invariant(SET<T> x, y, z)
    where x.nonNull.and(y.nonNull).and(z.nonNull) {
  x.except(y).equal(z).equal(forall(T e) {
    z.contains(e).equal(x.contains(e).and(y.contains(e).not));
    });
};
```

3.1.9 Loại phần tử: SET<T>

Định nghĩa: Hiệu của tập hợp này và một giá trị phần tử là tập hợp chứa tất cả các phần tử của tập hợp này ngoại trừ giá trị phần tử loại trừ. Nếu giá trị phần tử đó không nằm trong tập hợp này, hiệu sẽ bằng với tập hợp này.

```
invariant(SET<T> x, z; T d)
    where z.nonNull.and(z.nonNull).and(d.nonNull) {
  x.except(d).equal(z).equal(forall(T e) {
    z.contains(e).equal(x.contains(e).and(d.equal(e).not));
    });
};
```

3.1.10 Phép giao: SET<T>

Định nghĩa: Giao của hai tập hợp là một tập hợp chứa tất cả và chỉ những phần tử nằm trong cả hai tập hợp toán hạng.

```
invariant(SET<T> x, y, z)
    where x.nonNull.and(y.nonNull).and(z.nonNull) {
  x.intersection(y).equal(z).equal(forall(T e) {
    z.contains(e).equal(x.contains(e).and(y.contains(e)));
    });
};
```

3.1.11 Dạng chữ

Khi kiểu phần tử T có một dạng chữ, tập hợp các phần tử T có một dạng chữ, trong đó các phần tử của tập hợp được liệt kê trong dấu ngoặc cong và được phân tách bởi ký tự chấm phẩy.

```
SET<T>.literal ST {
  SET<T>          : "{" elements "}"   { $.equal($2); };
  SET<T> elements : elements ";" T     { $.except($2).equal($1); }
                  | T                  { $.contains($1);
                                         $.except($1).isEmpty; };
};
```

LƯU Ý: Dạng chữ cho các tập hợp này chỉ thiết thực cho các tập hợp đếm được tương đối nhỏ; tuy nhiên, điều này không có nghĩa là tất cả các tập hợp đều là những liệt kê tương đối nhỏ các phần tử.

Bảng 38. Ví dụ

dạng chữ	ý nghĩa

dạng chữ	ý nghĩa
{1; 3; 5; 7; 19}	một tập hợp các số nguyên hoặc số thực
{3; 1; 5; 19; 7}	một tập hợp tương tự các số nguyên hoặc số thực
{1.2 m; 2.67 m; 17.8 m}	một tập hợp các đại lượng vật lý rời rạc
{táo; cam; chuối}	một tập hợp các chuỗi ký tự

LƯU Ý: ITS dựa trên ký tự *nên* chọn một dạng chữ khác cho các tập hợp nếu Công nghệ triển khai có một dạng chữ bản địa cho các bộ sưu tập này.

3.1.12 Chuyển đổi tăng các giá trị phần tử thành các tập hợp: SET<T>

Một giá trị dữ liệu thuộc kiểu T có thể được chuyển đổi tăng thành một tập hợp nhỏ của T với giá trị dữ liệu đó là phần tử duy nhất của nó.

```
invariant(T x) {
  ((SET<T>)x).contains(x);
  ((SET<T>)x).except(x).isEmpty;
};
```

3.1.13 Bao lồi của tập hợp được sắp xếp toàn phần: IVL<T>

Các tập hợp đại lượng có thể là các tập hợp được sắp xếp toàn phần khi có một quan hệ thứ tự được định nghĩa giữa hai phần tử bất kỳ trong tập hợp. Lưu ý rằng "tập hợp được sắp xếp" không đồng nghĩa với Danh sách (LIST). Ví dụ, tập hợp {3; 2; 4; 88; 1} là một tập hợp được sắp xếp. Thứ tự của các phần tử trong biểu diễn tập hợp vẫn không liên quan, nhưng các phần tử có thể được so sánh để thiết lập thứ tự (1; 2; 4; 88).

Các tập hợp được sắp xếp toàn phần có bao lồi. Bao lồi của một tập hợp được sắp xếp toàn phần *S* là khoảng nhỏ nhất mà là tập cha của *S*. Đây là khái niệm sau này sẽ trở nên quan trọng.

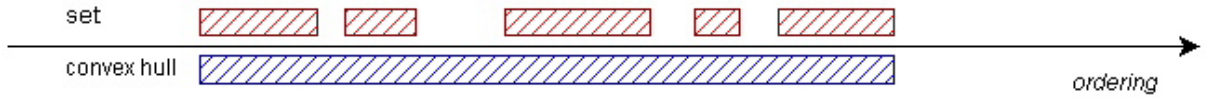
```
type Set<QTY> alias SET<QTY> {
  BL          totallyOrdered;
  IVL<T>  hull;
};

invariant(SET<QTY> s)
    where s.nonNull {
  s.totallyOrdered.equal(forall(QTY x, y)
```

```
        where s.contains(x).and(s.contains(y)) {
      x.compares(y); });
};

invariant(SET<QTY> s)
     where s.totallyOrdered {
  s.hull.contains(s);
  forall(T e)
       where s.contains(e) {
     s.hull.low.lessOrEqual(e);
     e.lessOrEqual(s.hull.high);
     };
};
```

Lưu ý rằng bao được định nghĩa khi và chỉ khi tập hợp *thực* là một tập hợp được sắp xếp toàn phần. Kiểu dữ liệu của bản thân các phần tử không cần phải được sắp xếp toàn phần. Ví dụ, kiểu dữ liệu PQ chỉ được sắp xếp một phần (do chỉ có thể so sánh các đại lượng cùng loại), nhưng một tập SET<PQ> vẫn có thể được sắp xếp toàn phần (nếu nó chỉ chứa các đại lượng có thể so sánh). Ví dụ, bao lồi của {4 s, 20 s, 55 s} là [4 s;55 s]; bao lồi của {"táo"; "cam"; "chuối"} không được xác định do giữa các phần tử không có mối quan hệ thứ tự; và bao lồi của {2 m; 4 m; 8 s} cũng không được xác định, vì nó không được sắp xếp *toàn phần* (giây không so sánh được với mét).



3.2 Danh sách (LIST) chuyên biệt hóa ANY

Định nghĩa: Một giá trị chứa các giá trị rời rạc (nhưng không nhất thiết riêng biệt) khác theo một trật tự đã xác định.

```
template<ANY T>
type Sequence<T> alias LIST<T> specializes ANY {
        T       head;
        LIST<T>  tail;
        BL      isEmpty;
        BL      notEmpty;
        T       item(INT index);
        BL      contains(T item);
        INT     length;
 literal   ST;
```

```
  promotion  LIST<T>  (T x);
};
```

Một chuỗi có thể chứa các hạng mục là các giá trị NULL.

3.2.1 Hạng mục đầu: T

Định nghĩa: Hạng mục đầu tiên trong chuỗi này. *Đầu* là một đặc tính định nghĩa cho ngữ nghĩa của chuỗi.

3.2.2 Hạng mục đuôi: LIST<T>

Định nghĩa: Chuỗi tiếp sau hạng mục đầu tiên trong chuỗi này. *Đuôi* là một đặc tính định nghĩa cho ngữ nghĩa của chuỗi.

3.2.3 Chuỗi thứ tự trống: BL

Định nghĩa: Một vị từ sẽ là *đúng* nếu chuỗi thứ tự này là một chuỗi trống, nghĩa là, nếu nó không chứa hạng mục nào.

Lưu ý sự khác biệt giữa chuỗi trống và NULL: một chuỗi trống là một chuỗi hợp lệ, không phải là một giá trị null.

```
invariant(LIST<T> x)
    where x.isEmpty {
  x.head.isNull;
  x.tail.isNull;
};
```

Lưu ý rằng đầu và đuôi là NULL chỉ là một điều kiện cần nhưng không đủ để xác định một danh sách rỗng, do một chuỗi có thể chứa các hạng mục là các giá trị NULL, điều kiện này có thể có nghĩa là danh sách này chỉ có một hạng mục đầu và hạng mục đó lại là NULL.

3.2.4 Chuỗi thứ tự không trống: BL

Định nghĩa: Một vị từ sẽ là *đúng* nếu chuỗi thứ tự này không rỗng. Phủ định của isEmpty (là trống)

```
invariant(LIST<T> x)
    where x.nonNull {
  x.notEmpty.equal(x.isEmpty.not);
};
```

3.2.5 Hạng mục theo chỉ mục: T

Định nghĩa: Hạng mục tại vị trí theo thứ tự cho trước (chỉ mục) trong chuỗi thứ tự. Chỉ mục 0 chỉ phần tử đầu tiên (đầu) của chuỗi.

```
invariant(LIST<T> list; INT index)
    where list.nonNull.and(index.nonNegative) {
  list.isEmpty.implies(list.item(index).isNull);
  list.notEmpty.and(index.isZero)
     .implies(list.item(index).equal(list.head));
  list.notEmpty.and(index.nonZero)
     .implies(list.item(index).equal(
       list.tail.item(index.predecessor)));
};
```

3.2.6 Chứa hạng mục: BL

Định nghĩa: Một vị từ sẽ là *đúng* nếu chuỗi thứ tự này chứa giá trị hạng mục cho trước.

```
invariant(LIST<T> list; T item)
    where list.nonNull {
  list.isEmpty.implies(list.contains(item).not);
  list.nonEmpty.and(item.nonNull).implies(list.contains(item).equal(
    list.head.equal(item).or(list.tail.contains(item))));
  list.notEmpty.and(item.isNull).implies(list.contains(item).equal(
     list.head.isNull.or(list.tail.contains(item)));
};
```

3.2.7 Độ dài: INT

Định nghĩa: Số phần tử trong chuỗi thứ tự. Các phần tử NULL được tính như các phần tử chuỗi thông thường.

```
invariant(LIST<T> list)
    where x.nonNull {
  list.isEmpty.equal(list.length.isZero);
  list.notEmpty.equal(list.length.equal(
    list.tail.length.successor));
};
```

3.2.8 Tương đương: BL, thừa kế từ ANY

Hai danh sách là bình đẳng khi và chỉ khi chúng đều trống, hoặc khi cả đầu và đuôi của chúng là bình đẳng.

```
invariant(LIST<T> x, y)
    where x.nonNull.and(y.nonNull) {
  x.isEmpty.and(y.isEmpty).implies(x.equal(y));
```

```
    x.notEmpty.and(y.notEmpty).and(x.head.nonNull)
      .implies(x.equal(y).equal(
        x.head.equal(y.head).and(x.tail.equal(y.tail))));
    x.notEmpty.and(y.notEmpty).and(x.head.isNull)
          .implies(x.equal(y).equal(
            y.heas.isNull.and(x.tail.equal(y.tail))));
  };
```

3.2.9 Dạng chữ

Khi kiểu phần tử T có một dạng chữ, chuỗi thứ tự *LIST* có một dạng chữ. Các phần tử danh sách được liệt kê, phân tách nhờ dấu chấm phẩy, và đóng trong ngoặc đơn.

```
  LIST<T>.literal ST {
    LIST<T> : "(" elements ")"       { $.equal($2); }
          | "(" ")"                  { $.isEmpty; };
    LIST<T> elements
          : T ";" elements           { $.head.equal($1);
                                       $.tail.equal($3); }
          | T                        { $.head.equal($1);
                                       $.tail.isEmpty; };
  };
```

Bảng 39. Ví dụ

dạng chữ	ý nghĩa
(1; 3; 5; 7; 19)	một chuỗi các số nguyên hoặc số thực
(3; 1; 5; 19; 7)	một chuỗi khác các số nguyên hoặc số thực
(1.2 m; 17.8 m; 2.67 m)	một chuỗi các đại lượng vật lý rời rạc
(táo; cam; chuối)	một chuỗi các chuỗi ký tự

LƯU Ý: ITS dựa trên ký tự *nên* chọn một dạng chữ khác cho các chuỗi nếu Công nghệ triển khai có một dạng chữ bản địa cho các bộ sưu tập này.

3.2.10 Chuyển đổi tăng các giá trị hạng mục thành chuỗi: LIST<T>

Một giá trị dữ liệu thuộc kiểu T có thể được chuyển đổi tăng thành một chuỗi nhỏ của T với giá trị dữ liệu đó là hạng mục duy nhất của nó.

```
  invariant(T x) {
```

```
    ((LIST<T>)x).head.equal(x);
    ((LIST<T>)x).tail.isEmpty;
};
```

3.3 Chuỗi được tạo (GLIST) chuyên biệt hóa LIST

Định nghĩa: Một chuỗi tuần hoàn hay đơn điệu các giá trị được tạo ra từ một vài tham số, thay vì được liệt kê. Được dùng để xác định các điểm lấy mẫu thường xuyên cho tín hiệu y sinh.

```
type GeneratedSequence<QTY T> alias GLIST specializes LIST<T> {
  T    head;
  QTY  increment;
  INT  period;
  INT  denominator;
};
```

Bảng 40. Tóm tắt đặc tính của Chuỗi được tạo

Tên	Kiểu	Mô tả
head (đầu)	T	Hạng mục đầu tiên trong chuỗi này. *Đầu* là một đặc tính định nghĩa cho ngữ nghĩa của chuỗi.
increment (số gia tăng)	QTY	Mức chênh lệch giữa một giá trị và giá trị khác trước đó của nó. Ví dụ, để tạo chuỗi (1; 4; 7; 10; 13; ...) số gia tăng là 3; tương tự để tạo chuỗi (1; 1; 4; 4; 7; 7; 10; 10; 13; 13; ...) số gia tăng cũng là 3.
period (chu kỳ)	INT	Nếu non-NULL, chỉ ra rằng chuỗi luân phiên, nghĩa là, sau nhiều số gia tăng, các giá trị hạng mục chuỗi quay lại để bắt đầu từ giá trị hạng mục chuỗi ban đầu. Ví dụ, chuỗi (1; 2; 3; 1; 2; 3; 1; 2; 3; ...) có chu kỳ 3; tương tự chuỗi (1; 1; 2; 2; 3; 3; 1; 1; 2; 2; 3; 3; ...) cũng có chu kỳ 3.
denominator (mẫu số)	INT	Số nguyên mà chỉ mục cho chuỗi được chia cho, thực ra là số lần chuỗi tạo ra cùng một giá trị hạng mục chuỗi trước khi gia tăng lên giá trị hạng mục chuỗi tiếp theo. Ví dụ, để tạo chuỗi (1; 1; 1; 2; 2; 2; 3; 3; 3; ...) mẫu số là 3.

Hạng mục tại một chỉ mục nhất định trong danh sách được tính toán bằng cách thực hiện một phép chia số nguyên đối với chỉ mục (i) và mẫu số (d) rồi sau đó lấy số dư của giá trị đó với chu kỳ (p). Nhân giá trị này với số gia (Δx) và cộng với đầu (x_0.)

$$x_i = x_0 + \Delta x \times (i/d) \bmod p$$

```
invariant(GLIST<T> list, INT index)
    where list.nonNull.and(index.nonNull) {
  list.period.nonNull.implies(list.item(index).equal(
    list.head.plus(item.dividedBy(list.increment.denominator)
      .remainder(list.period)).times(increment)));
  list.period.isNull.implies(list.item(index).equal(
    list.head.plus(item.dividedBy(list.increment.denominator))
      .times(increment)));
};
```

3.3.1 Hạng mục đầu: T, thừa kế từ LIST

Đây là giá trị bắt đầu của danh sách được tạo.

3.3.2 Số gia tăng: QTY

Định nghĩa: Mức chênh lệch giữa một giá trị và giá trị khác trước đó của nó. Ví dụ, để tạo chuỗi (1; 4; 7; 10; 13; ...) số gia là 3; tương tự để tạo chuỗi (1; 1; 4; 4; 7; 7; 10; 10; 13; 13; ...) số gia cũng là 3.

```
invariant(GLIST<T> x) {
  x.increment.dataType.implies(T.diffType);
};
```

3.3.3 Đếm bước chu kỳ: INT

Định nghĩa: Nếu non-NULL, chỉ ra rằng chuỗi luân phiên, nghĩa là, sau nhiều số gia, các giá trị hạng mục chuỗi quay lại để bắt đầu từ giá trị hạng mục chuỗi ban đầu. Ví dụ, chuỗi (1; 2; 3; 1; 2; 3; 1; 2; 3; ...) có chu kỳ 3; tương tự chuỗi (1; 1; 2; 2; 3; 3; 1; 1; 2; 2; 3; 3; ...) cũng có chu kỳ 3.

Chu kỳ cho phép lặp đi lặp lại việc lấy mẫu cùng một không gian mẫu. "Dạng sóng" của bộ tạo sóng tuần hoàn này luôn luôn là "răng cưa", cũng giống như hàm x của máy hiện sóng.[50]

3.3.4 Mẫu số: INT

Định nghĩa: Số nguyên mà chỉ mục cho chuỗi được chia cho, thực ra là số lần chuỗi tạo ra cùng một giá trị hạng mục chuỗi trước khi gia tăng lên giá trị hạng mục chuỗi tiếp theo. Ví dụ, để tạo chuỗi (1; 1; 1; 2; 2; 2; 3; 3; 3; ...) mẫu số là 3.

Việc sử dụng mẫu số là để cho phép nhiều chuỗi được tạo quét định kỳ một không gian đa chiều. Ví dụ, một màn hình TV (trừu tượng) sử dụng 2 bộ tạo sóng như vậy cho các cột và hàng điểm ảnh. Chẳng hạn, nếu có 200 dòng quét và 320 cột mành, bộ tạo sóng cột sẽ có mẫu số 1 và bộ tạo sóng dòng sẽ có mẫu số 320.

Bảng 41. Ví dụ về Chuỗi được tạo

head **đầu**	**increment** **số gia**	**denominator** **mẫu số**	**period** **chu kỳ**	**ý nghĩa**
0	1	1	∞	Chuỗi định danh trong đó mỗi hạng mục bằng với chỉ mục của nó.
198706052000	2 giờ	1	∞	Chuỗi bắt đầu từ ngày 5 tháng sáu năm 1987 vào lúc 7 giờ tối và gia tăng sau mỗi hai giờ: 9 giờ tối, 11 giờ đêm, 1 giờ sáng (ngày 6 tháng sáu), 3 giờ sáng, 5 giờ sáng, và cứ tiếp tục như thế.
0 V	1 mV	1	100	Sóng x của một máy ghi sóng số, quét giữa 0 và 100 mV trong 100 bước 1 mV. Các dữ liệu này không cho biết tần suất do chúng ta không biết có bao nhiêu thời gian trôi qua giữa mỗi bước của chỉ mục.
2002072920300	100 us	1	∞	Một cơ sở thời gian từ ngày 29 tháng sáu năm 2002 vào lúc 8:30 tối với 100 us giữa mỗi bước của chỉ mục. Nếu được kết hợp với bộ tạo sóng trước làm chiều lấy mẫu thứ hai, sẽ mô tả cơ sở thời gian x của máy ghi sóng số là 1 mV trên mỗi 100 us. Ở mức 100 bước mỗi chu kỳ, chu kỳ là 10 ms, tương đương với tần số 100 Hz.
0 V	1 mV	100	100	Kết hợp bộ tạo sóng này với hai bộ tạo sóng trước có thể mô tả một không gian lấy mẫu ba chiều với

head đầu	increment số gia	denominator mẫu số	period chu kỳ	ý nghĩa
				hai điện áp và thời gian. Bộ tạo sóng này cũng có bước 1 mV và có 100 bước mỗi chu kỳ, tuy nhiên, nó chỉ bước sau mỗi 100 số gia chỉ mục, vì vậy, bộ tạo sóng điện áp đầu tiên tạo một chu kỳ trọn vẹn trước khi bộ tạo sóng này được gia tăng. Có thể coi hai điện áp là "hàng" và "cột" của một "khung lấy mẫu". Với bộ tạo sóng trước là cơ sở thời gian, sẽ tạo ra quét khung lấy mẫu 100 mV × 100 mV với tốc độ khung là 1 Hz.

3.4 Chuỗi lấy mẫu (SLIST) chuyên biệt hóa LIST

Định nghĩa: Một chuỗi các giá trị lấy mẫu được chia tỷ lệ và chuyển ngữ từ một danh sách các giá trị số nguyên. Được dùng để chỉ rõ các tín hiệu y sinh lấy mẫu.

```
type SampledSequence<QTY T> alias SLIST specializes LIST<T> {
  T        origin;
  QTY      scale;
  LIST<INT>  digits;
};
```

Bảng 42. Tóm tắt đặc tính của Chuỗi lấy mẫu

Tên	Kiểu	Mô tả
origin (gốc)	T	Gốc của thang tỷ lệ giá trị hạng mục danh sách, nghĩa là, đại lượng vật lý mà một chữ số 0 trong chuỗi sẽ biểu diễn.
scale (thang tỷ lệ)	QTY	Đại lượng tỷ lệ-thang tỷ lệ được tính từ chuỗi chữ số.
digits (chữ số)	LIST<INT>	Một chuỗi các chữ số thô cho các giá trị mẫu. Đây thường là kết quả thô của một bộ chuyển đổi tương tự/số.

Hạng mục tại một chỉ mục nhất định (i) trong danh sách được tính toán bằng cách nhân hạng mục tại cùng chỉ mục trong chuỗi chữ số (d_i) với tỷ lệ (s) rồi cộng giá trị đó với gốc (x_o).

$x_i = x_o + s \times d_i$

```
invariant(SLIST<T> list, INT index)
    where list.nonNull.and(index.nonNegative) {
  list.item(index).equal(
    list.scale.times(digits.item(index))
      .plus(list.origin));
};
```

3.4.1 Gốc thang tỷ lệ: T

Định nghĩa: Gốc của thang tỷ lệ giá trị hạng mục danh sách, nghĩa là, đại lượng vật lý mà một chữ số 0 trong chuỗi sẽ biểu diễn.

3.4.2 Hệ số thang tỷ lệ: QTY

Định nghĩa: Đại lượng tỷ lệ-thang tỷ lệ được tính từ chuỗi chữ số.

```
invariant(SLIST<T> x) {
  x.scale.dataType.implies(T.diffType);
};
```

3.4.3 Chữ số lấy mẫu: LIST<INT>

Định nghĩa: Một chuỗi các chữ số thô cho các giá trị mẫu. Đây thường là kết quả thô của một bộ chuyển đổi tương tự/số.

3.5 Bộ sưu tập (BAG) chuyên biệt hóa ANY

Định nghĩa: Một bộ sưu tập không có thứ tự các giá trị, trong đó mỗi giá trị có thể được chứa nhiều hơn một lần trong bộ sưu tập.

```
template<ANY T>
type Bag<T> alias BAG<T> specializes ANY {
        INT     contains(T kind);
        BL      isEmpty;
        BL      notEmpty;
        BAG<T>  plus(BAG<T> x);
        BAG<T>  minus(BAG<T> x);
  promotion  BAG<T>  (T x);
};
```

LƯU Ý: Một *TÚI (Bộ sưu tập)* có thể được biểu diễn theo hai cách. Hoặc dưới dạng một liệt kê đơn giản các phần tử, bao gồm cả các phần tử lặp lại, hoặc dưới dạng một "túi nén" theo đó nội dung của *TÚI* được liệt kê theo các cặp giá trị phần tử và số. Một biểu đồ thể hiện tần số tuyệt đối là một *TÚI* được biểu diễn ở dạng nén. *TÚI* do đó rất hữu ích trong việc truyền tải thông tin các mẫu dữ liệu thống kê thô.

3.5.1 Chứa hạng mục: INT

Định nghĩa: Số hạng mục trong túi này với giá trị hạng mục cho trước.

Đây là đặc tính ngữ nghĩa nguyên thủy của một *TÚI*, dựa trên đó tất cả các đặc tính khác được định nghĩa.

```
invariant(BAG<T> bag; T item)
    where bag.nonNull.and(item.nonNull) {
  bag.contains(item).nonNegative;
  bag.isEmpty.equal(bag.contains(item).isZero);
};
```

3.5.2 Không trống: BL

Định nghĩa: Một vị từ chỉ ra rằng *TÚI* này có chứa hạng mục.

```
invariant(BAG<T> bag)
    where bag.nonNull {
  bag.notEmpty.equal(exists(T item) {
    bag.contains(item);
    });
};
```

3.5.3 Bộ sưu tập trống: BL

Định nghĩa: Một vị từ chỉ ra rằng *TÚI* này không có phần tử (phủ định của vị từ isEmpty. *TÚI* trống là một giá trị hợp lệ, *không phải* một giá trị ngoại lệ (NULL).

```
invariant(BAG<T> bag)
    where bag.nonNull {
  bag.isEmpty.equal(notEmpty.not);
};
```

3.5.4 Phép cộng: BAG<T>

Định nghĩa: Một *TÚI* chứa tất cả các hạng mục của các *TÚI* toán hạng, nghĩa là số hạng mục của mỗi giá trị hạng mục được thêm vào.

```
invariant(BAG<T> x, y, z)
    where x.nonNull.and(y.nonNull) {
```

```
  x.plus(y).equal(z).equal(
    forall(T e)
        where e.nonNull {
      z.contains(e).equal(x.contains(e)
              .plus(y.contains(e)));
      });
};
```

3.5.5 Phép trừ: BAG<T>

Định nghĩa: Một *TÚI* chứa tất cả các hạng mục của *TÚI* này (số bị trừ) bị giảm bớt bởi các hạng mục trong *TÚI* khác (số trừ). *TÚI* không thể chứa số thiếu hụt. Khi số trừ chứa nhiều hạng mục của một giá trị hơn so với số bị trừ, hiệu chứa 0 hạng mục của giá trị đó.

```
invariant(BAG<T> x, y, z)
    where x.nonNull.and(y.nonNull) {
  x.minus(y).equal(z).equal(
    forall(T e)
        where e.nonNull {
      exists(INT n)
          where n.equal(x.contains(e).minus(y.contains(e)) {
      n.nonNegative.equal(z.contains(e));
      n.isNegative.equal(z.contains(e).isZero);
      };
    });
};
```

3.5.6 Chuyển đổi tăng giá trị hạng mục cho Bộ sưu tập: BAG<T>

Một giá trị dữ liệu thuộc kiểu T có thể được chuyển đổi tăng thành một *TÚI* nhỏ kiểu T với giá trị dữ liệu đó là hạng mục duy nhất của nó.

```
invariant(T x) {
  ((BAG<T>)x).contains(x).equal(1);
  forall(T y) {
    ((BAG<T>)x).contains(y)
          .implies(x.equal(y)) };
};
```

3.6 Khoảng (IVL) chuyên biệt hóa SET

Định nghĩa: Một tập hợp các giá trị liên tiếp của một kiểu dữ liệu cơ bản có thứ tự.

Bất kỳ kiểu dữ liệu nào có thứ tự đều có thể là cơ sở của một *IVL*; việc kiểu cơ bản là rời rạc hay liên tục là không quan trọng. Nếu kiểu dữ liệu cơ bản chỉ được sắp xếp thứ tự một phần, tất cả các phần tử của *IVL* phải là phần tử của một tập con được sắp xếp thứ tự toàn phần của một kiểu dữ liệu được sắp xếp thứ tự một phần.

Ví dụ, giá trị PQ được coi là có thứ tự. Tuy nhiên, thứ tự của các giá trị PQ chỉ là một phần; một thứ tự toàn phần chỉ được định nghĩa giữa các đại lượng có thể so sánh (các đại lượng có cùng đơn vị vật lý). Trong khi tồn tại các giá trị *IVL* giữa 2 và 4 mét, không có giá trị *IVL* nào giữa 2 mét và 4 giây.

IVL là các TẬP HỢP và có tất cả các đặc tính của TẬP HỢP. Tuy nhiên, hợp và hiệu của TẬP HỢP có thể không phải là TẬP HỢP, do các phần tử của các TẬP HỢP hợp và hiệu này có thể không liền kề. Giao của các TẬP HỢP luôn luôn là TẬP HỢP.

```
template<QTY T>
type Interval<T> alias IVL<T> specializes SET<T> {
        T      low;
        BL     lowClosed;
        T      high;
        BL     highClosed;
        QTY    width;
        T      center;
        IVL<T>  hull;
        IVL<T>  hull(IVL<T> x);
  literal   ST;
  promotion  IVL<T>  (T x);
  demotion   T;
};
```

3.6.1 Ranh giới thấp : T

Định nghĩa: Đây là giới hạn thấp.

```
invariant(IVL<T> x; T e)
    where x.nonNull.and(x.contains(e)) {
  x.low.lessOrEqual(e);
};
```

3.6.2 Ranh giới cao : T

Định nghĩa: Đây là giới hạn cao.

```
invariant(IVL<T> x; T e)
    where x.nonNull.and(x.contains(e)) {
  e.lessOrEqual(x.high);
};
```

3.6.3 Độ rộng : QTY

Định nghĩa: Mức chênh lệch giữa ranh giới cao và ranh giới thấp. Mục đích của việc phân biệt *độ rộng* là để xử lý tất cả các trường hợp thông tin không đầy đủ về mặt đối xứng. Trong bất kỳ biểu diễn TẬP HỢP nào, chỉ cần nêu hai trong số ba đặc tính cao, thấp, và *độ rộng*, còn đặc tính thứ ba có thể suy ra.

Khi biết cả hai ranh giới, có thể suy được *độ rộng* bằng cách lấy cao trừ thấp. Khi biết một ranh giới và *độ rộng*, thì cũng biết được ranh giới còn lại. Khi không biết ranh giới nào, vẫn có thể biết *độ rộng*. Ví dụ, có thể biết rằng một hoạt động kéo dài khoảng 30 phút, nhưng có thể chưa biết hoạt động đó bắt đầu khi nào.

Lưu ý rằng kiểu dữ liệu của *độ rộng* không phải lúc nào cũng giống nhau về ranh giới. Đối với các đại lượng thang tỷ lệ (REAL, PQ, MO) nó là như nhau. Đối với các đại lượng thang hiệu (ví dụ, TS) nó là kiểu dữ liệu của hiệu (ví dụ, PQ theo đơn vị thời gian cho TS). Đối với các phần tử rời rạc (INT) độ rộng có thể là một REAL chỉ số phần tử trong khoảng chia cho 2.

```
invariant(IVL<T> x) {
  x.low.lessOrEqual(x.high);
  x.width.equal(x.high.minus(x.low));
};

invariant(IVL<T> x) {
  x.width.dataType.implies(T.diffType);
};
```

3.6.4 Giá trị trung tâm : T

Định nghĩa: Trung bình cộng của giá trị *IVL* (thấp cộng cao chia cho 2). Mục đích của việc phân biệt *trung tâm* như một đặc tính ngữ nghĩa là để chuyển đổi các giá trị *IVL* thành và từ các giá trị điểm.

Lưu ý rằng *trung tâm* không phải lúc nào cũng tồn tại cho mọi giá trị *IVL*. Đáng chú ý là các giá trị *IVL* vô hạn trên một phía không có *trung tâm*. Tương tự, các giá trị *IVL* của các kiểu cơ bản rời rạc với một số chẵn các phần tử cũng không có *trung tâm*.

Nếu không biết một (hoặc cả hai) ranh giới của một *IVL*, vẫn có thể khẳng định được *trung tâm*. Trong thực tế, trường hợp sử dụng chính của *trung tâm* là để được khẳng định khi không biết ranh giới.

```
invariant(IVL<T> x)
    where x.low.nonNull.and(x.high.nonNull) {
  x.center.equal(x.low.plus(x.width.times(0.5))));
};

invariant(IVL<T> x)
    where x.low.isNull.or(x.high.isNull) {
  x.center.notApplicable;
};
```

3.6.5 Ranh giới thấp đóng : BL

Định nghĩa: Xác định liệu ranh giới thấp có được gộp vào *IVL* (đóng) hay bị loại trừ khỏi *IVL* (mở).

```
invariant(IVL<T> x)
    where x.nonNull {
  x.low.nonNull.implies(x.lowClosed.equal(x.contains(x.low)));
  x.low.isNull.implies(x.lowClosed.not);
};
```

3.6.6 Ranh giới cao đóng : BL

Định nghĩa: Xác định liệu ranh giới cao có được gộp vào *IVL* (đóng) hay bị loại trừ khỏi *IVL* (mở).

```
invariant(IVL<T> x)
    where x.nonNull {
  x.high.nonNull.implies(x.highClosed.equal(x.contains(x.high)));
  x.high.isNull.implies(x.highClosed.not);
};
```

3.6.7 Dạng chữ

Dạng chữ cho *IVL* được định nghĩa sao cho trực quan nhất ở mức có thể với con người. Năm dạng khác nhau được định nghĩa:[51]

1. dạng khoảng sử dụng ngoặc vuông, ví dụ, "[3.5; 5.5[";
2. dạng gạch ngang, ví dụ, "3.5-5.5";
3. dạng "bộ so sánh", sử dụng các ký hiệu toán tử tương quan, ví dụ, "<5.5";
4. dạng trung tâm-độ rộng, ví dụ, "4.5[2.0[";

5. dạng chỉ có độ rộng sử dụng ngoặc vuông, ví dụ, "[2.0[".

```
IVL<T>.literal ST {
  IVL<T> range : interval            { $.equal($1); }
               | dash                { $.equal($1); }
               | comparator          { $.equal($1); }
               | center_width        { $.equal($1); }
               | width               { $.equal($1); };

  IVL<T> interval
               : open T ";" T close;   { $.low.equal($2);
                                         $.high.equal($4);
                                         $.lowClosed.equal($1);
                                         $.highClosed.equal($5); };
  BL open      : "["                   { $.equal(true); }
               | "]"                   { $.equal(false); };
  BL close     : "]"                   { $.equal(true); }
               | "["                   { $.equal(false); };
  IVL<T> width
               : open T.diffType close { $.width.equal($2);
                                         $.lowClosed.equal($1);
                                         $.highClosed.equal($3); };
  IVL<T> center_width
               : T width     { $.center.equal($1);
                               $.width.equal($2.width);
                               $.lowClosed.equal($2.lowClosed);
                               $.highClosed.equal($2.highClosed); };
  IVL<T> dash  : T "-" T;          { $.low.equal($2);
                                     $.high.equal($4);
                                     $.lowClosed.equal(true);
                                     $.highClosed.equal(true); };
  IVL<TS> comparator
               : "<"  T            { $.high.equal(T);
                                     $.high.closed(false);
                                     $.low.negativelyInfinite; }
               | ">"  T            { $.low.equal(T);
                                     $.low.closed(false);
```

```
                    $.high.positivelyInfinite; }
        | "<=" T      { $.high.equal(T);
                    $.high.closed(true);
                    $.low.negativelyInfinite; }
        | ">=" T      { $.low.equal(T);
                    $.low.closed(true);
                    $.high.positivelyInfinite; };
};
```

3.6.8 Chuyển đổi tăng các giá trị phần tử thành khoảng : IVL<T>

Một đại lượng thuộc kiểu T có thể được chuyển đổi tăng thành một *IVL* nhỏ trong đó thấp và cao là bình đẳng và ranh giới là đóng.

```
invariant(T x) {
  ((IVL<T>)x).low.equal(x);
  ((IVL<T>)x).high.equal(x);
  ((IVL<T>)x).highClosed;
  ((IVL<T>)x).lowClosed;
};
```

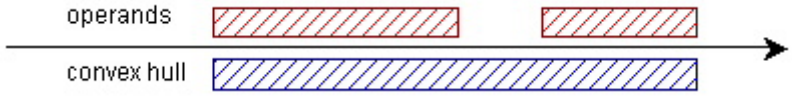
3.6.9 Chuyển đổi giảm khoảng thành giá trị phần tử đại diện : T

Một *IVL* có thể được chuyển đổi giảm thành một đại lượng đơn giản của kiểu T có tính đại diện cho toàn bộ *IVL*. Nếu cả hai ranh giới là hữu hạn, đây là trung tâm. Nếu một ranh giới là vô hạn, giá trị đại diện là ranh giới còn lại. Nếu cả hai ranh giới là vô hạn, không thể áp dụng chuyển đổi thành một giá trị điểm.

```
invariant(IVL<T> x)
    where x.nonNull {
  x.low.nonNull.and(x.high.nonNull).implies(((T)x).equal(x.center));
  x.high.nonNull.and(x.low.isNull).implies(((T)x).equal(x.high));
  x.low.nonNull.and(x.high.isNull).implies(((T)x).equal(x.low));
  x.low.isNull.and(x.high.isNull).implies(((T)x).notApplicable);
};
```

3.6.10 Bao lồi : IVL<T>, thừa kế từ SET

Định nghĩa: Một bao lồi hay "bao khoảng" của hai giá trị *IVL* là *IVL* tối thiểu mà là tập cha của các toán hạng của nó.



Bao lồi của hai Khoảng (operands = các toán hạng, convex hull = bao lồi)

```
invariant(IVL<T> h, IVL<T> i, j)
    where h.equal(i.hull(j)) {
  i.low.lessOrEqual(j.low).implies(h.low.equal(i.low));
  j.low.lessOrEqual(i.low).implies(h.low.equal(j.low));
  i.high.lessOrEqual(j.high).implies(h.high.equal(j.high));
  j.high.lessOrEqual(i.high).implies(h.high.equal(i.high));
};
```

3.7 Khoảng các đại lượng vật lý (IVL<PQ>) chuyên biệt hóa IVL

Định nghĩa: Một tập hợp các giá trị đại lượng vật lý liên tiếp.

Một khoảng các đại lượng vật lý được xây dựng từ kiểu khoảng chung. Tuy nhiên, nhận thức được rằng có thể tính đơn vị từ các ranh giới, chúng tôi thêm vào ngữ nghĩa bổ sung và một dạng chữ riêng biệt. Quan điểm bổ sung về một khoảng các đại lượng vật lý là một khoảng số thực với một đơn vị.

```
type Interval<PQ> alias IVL<PQ> {
  IVL<REAL> value;
  CS  unit;
};
```

Đơn vị áp dụng cho cả ranh giới thấp và cao.

```
invariant(IVL<PQ> x)
    where x.nonNull {
  x.value.nonNull;
  x.low.value.equal(x.value.low);
  x.low.unit.equal(x.unit);
  x.lowClosed.equal(x.value.lowClosed);
  x.high.value.equal(x.value.high);
  x.high.unit.equal(x.unit);
  x.highClosed.equal(x.value.highClosed);
};
```

Dạng chữ đặc biệt đơn giản là một khoảng số thực, một không gian và đơn vị.

```
IVL<PQ>.literal ST {
  IVL<PQ>  : IVL<REAL> " " unit  { $.value($1);
                                   $.unit.equal($3); }
           | IVL<REAL>     { $.equal($1); };
  CS unit  : ST            { $.value.equal($1);
```

```
        $.codeSystem(2.16.840.1.113883.3.2); };
    };
```

Ví dụ: "[0;5] mmol/L" hay "<20 mg/dL" là các dạng chữ hợp lệ của khoảng đại lượng vật lý. Dạng khoảng chung, ví dụ, "[50 nm; 2 m]" cũng được chấp nhận.

3.8 Khoảng thời điểm (IVL<TS>) chuyên biệt hóa IVL

Định nghĩa: Một tập hợp các giá trị dấu thời gian liên tiếp.

Kiểu dữ liệu khoảng chung cũng định nghĩa khoảng thời điểm. Tuy nhiên, có một số cân nhắc đặc biệt về biểu diễn dạng chữ và chuyển đổi khoảng thời điểm, sẽ được chỉ ra trong mục này.

```
type Interval<TS> alias IVL<TS> {
  literal    ST;
  promotion  IVL<TS> (TS x);
};
```

3.8.1 Chuyển đổi tăng các giá trị thời điểm thành khoảng : IVL<TS>, thừa kế từ IVL

Một TS có thể được chuyển đổi tăng thành một IVL<TS> theo đó ranh giới thấp là bản thân giá trị TS, và độ rộng được suy ra từ độ chính xác của TS và thời gian của giai đoạn lịch ít quan trọng nhất được xác định. Ranh giới cao là mở. Ví dụ, dạng chữ TS "200009" được chuyển đổi thành một IVL<TS> với ranh giới thấp là 200009 và độ rộng là 30 ngày, vốn là khoảng "[200009;200010[".

3.8.2 Dạng chữ

Dạng chữ cho khoảng thời điểm là ngoại lệ.

- "Dạng gạch ngang" không được chấp nhận đối với các khoảng thời điểm
- Thay vào đó một "dạng bao" được định nghĩa

Để tránh các xung đột về cú pháp với múi giờ và các hồ sơ sử dụng hơi khác nhau của tiêu chuẩn ISO 8601 xảy ra trên một vài nền tảng ITS, **dạng gạch ngang của khoảng không được chấp nhận đối với kiểu dữ liệu** IVL<TS>. Dạng khoảng sử dụng ngoặc vuông được ưa chuộng hơn.

Ví dụ: ngày 12 tháng năm năm 1987 từ 8 đến 9:30 tối là "[198705122000;198705122130]".

LƯU Ý: Độ chính xác của một ranh giới khoảng được nêu là không phù hợp cho khoảng. Một người có thể hiểu sai rằng khoảng "[19870901;19870930]" tương đương toàn bộ tháng Chín năm 1987 cho tới hết ngày 30 tháng Chín. Tuy nhiên, không phải như vậy! Cách thích hợp để biểu thị toàn bộ chu kỳ lịch (ví dụ, giờ, ngày, tháng,

năm, v.v.) trong phép biểu diễn khoảng là sử dụng một ranh giới mở. Ví dụ, toàn bộ tháng Chín năm 1987 được biểu diễn thành "[198709;198710[".[52]

"Dạng bao" của dạng chữ được định nghĩa là bao lồi (xem IVL.hull) của chuyển đổi tăng khoảng từ hai dấu thời gian.

```
  IVL<TS> hull : TS ".." TS     { $.equal(((IVL<TS>)$1)
                                  .hull((IVL<TS>)$3));
};
```

Ví dụ, "19870901..19870930" là dạng chữ hợp lệ sử dụng dạng bao. Giá trị tương đương với dạng khoảng "[19870901;19871001[".[53]

Dạng bao tiếp tục chấp nhận viết tắt, trong đó dạng chữ dấu thời gian cao hơn không cần nhắc lại các chữ số nằm về phía trái mà cũng xuất hiện trong dạng chữ dấu thời gian thấp hơn. Hai dấu thời gian được canh phải và các chữ số về bên trái được sao chép từ dạng chữ dấu thời gian thấp hơn sang cao hơn. Đây là một thao tác chuỗi đơn giản và không được định nghĩa chính thức ở đây.

Ví dụ: từ ngày 12 tháng Năm năm 1987 tới ngày 23 tháng Năm năm 1987 là "19870512..23". Tuy nhiên, lưu ý rằng từ ngày 12 tháng Năm năm 1987 tới ngày 2 tháng Sáu năm 1987 là "19870512..0602", mà không phải là "20000512..02".

4. Phần mở rộng kiểu chung

Phần mở rộng kiểu chung là các kiểu chung với một kiểu tham số, và là kiểu mở rộng (chuyên biệt hóa) kiểu tham số của nó. Trong ngôn ngữ định nghĩa kiểu dữ liệu chính thức, phần mở rộng kiểu chung tuân theo mẫu sau: **mẫu**<ANY T> **kiểu***TênPhầnMởRộngKiểuChung***chuyênbiệthóa** T { ... }; Các phần mở rộng kiểu chung này thừa kế hầu hết đặc tính của kiểu cơ bản của chúng và bổ sung một số đặc điểm cụ thể. Phần mở rộng kiểu chung là một chuyên biệt hóa của kiểu cơ bản, vì vậy một giá trị của kiểu dữ liệu phần mở rộng có thể được sử dụng thay cho kiểu dữ liệu cơ bản của nó.

LƯU Ý: Các giá trị của kiểu được mở rộng có thể được thay thế cho kiểu cơ bản của chúng. Tuy nhiên, ITS có thể đưa ra một số ràng buộc về việc có thể chứa phần mở rộng nào. Cụ thể, không cần phải định nghĩa các phần mở rộng cho những thành phần mang giá trị của các đặc tính giá trị dữ liệu. Vì vậy, trong khi bất kỳ giá trị dữ liệu nào cũng đều có thể được chú thích bên ngoài đặc tả kiểu dữ liệu, ITS có thể không cung cấp cách thức để chú thích cho giá trị của một đặc tính giá trị dữ liệu. **Vào thời điểm này, HL7 không cho phép sử dụng phần mở rộng kiểu chung, ngoại trừ khi được cho phép một cách rõ ràng (trong đặc tả này hay một đặc tả HL7 khác) cho những trường hợp sử dụng trong đó chức năng cao cấp này là quan trọng. Trong những trường hợp này, các phần mở rộng kiểu chung phải được phản ánh**

cụ thể và rõ ràng trong sơ đồ RIM, MIM, RMIM và HMD của HL7 (như có thể áp dụng), như là kết quả của nội dung đã được Ủy ban kỹ thuật bỏ phiếu.[54]

4.1 Hạng mục lịch sử (HXIT) chuyên biệt hóa T

Định nghĩa: Một phần mở rộng kiểu dữ liệu chung mà gắn một dải thời gian cho giá trị dữ liệu bất kỳ của kiểu dữ liệu bất kỳ. Dải thời gian là thời gian trong đó thông tin được biểu diễn bởi giá trị (đã) là có hiệu lực.

Nếu kiểu cơ bản T không sở hữu đặc tính Thời gian có hiệu lực, HXIT sẽ bổ sung đặc tính đó vào kiểu cơ bản. Tuy nhiên, nếu kiểu cơ bản T có đặc tính thời gian có hiệu lực (hiện chỉ có kiểu dữ liệu EN), đặc tính đó sẽ được liên kết với đặc tính thời gian có hiệu lực của HXIT[55]

```
template<ANY T>
type HistoryItem<T> alias HXIT<T> specializes T {
  IVL<TS> validTime;
};
```

4.1.1 Thời gian có hiệu lực : IVL<TS>

Định nghĩa: Khoảng thời gian trong đó thông tin cho trước đã là, là hoặc được kỳ vọng sẽ là có hiệu lực. Khoảng có thể mở hoặc đóng vô hạn hay không xác định về cả hai phía.

4.2 Lịch sử (HIST) chuyên biệt hóa SET<HXIT>

Định nghĩa: Một tập hợp các giá trị dữ liệu có đặc tính thời gian có hiệu lực và nhờ đó phù hợp với kiểu dữ liệu HXIT. Thông tin lịch sử không bị giới hạn ở quá khứ; các giá trị tương lai được kỳ vọng cũng có thể xuất hiện.

Mục đích của kiểu dữ liệu *HIST* là để nắm bắt các giá trị lịch sử (và tương lai) đúng của một hạng mục, thay vì hình thức biên bản kiểm tra lịch sử của các giá trị mà một hệ thống cho trước bất kỳ đã giữ cho hạng mục đó.

```
template<ANY T>
type History<T> alias HIST<T> specializes SET<HXIT<T>> {
        HXIT<T>  earliest;
        HIST     exceptEarliest;
        HXIT<T>  latest;
        HIST     exceptLatest;
 demotion  HXIT<T>;
};
```

Ngữ nghĩa chủ yếu không ngăn cấm các khoảng thời gian chồng nhau. Tuy nhiên, nếu hai hạng mục lịch sử có cùng ranh giới IVL.low và IVL.high trong khoảng

thời gian có hiệu lực, thì không xác định được hạng mục nào được xem là sớm nhất (mới nhất).

```
invariant(HIST x)
    where x.nonNull {
  x.notEmpty;
  ((T)x).equal(x.latest);
};
```

4.2.1 Hạng mục sớm nhất : HXIT<T>

Định nghĩa: Hạng mục trong tập hợp mà có ranh giới IVL.low (thời gian bắt đầu hiệu lực) là nhỏ hơn hay bằng (nghĩa là, trước) ranh giới thấp của bất kỳ hạng mục lịch sử nào khác trong tập hợp.

```
invariant(HIST x; HXIT<T> e)
    where x.contains(e) {
  x.earliest.validTime.low.lessOrEqual(e.validTime.low);
};
```

4.2.2 Hạng mục mới nhất : HXIT<T>

Định nghĩa: Hạng mục trong tập hợp mà có ranh giới IVL.high (thời gian kết thúc hiệu lực) lớn hơn hay bằng (nghĩa là, sau) ranh giới cao của bất kỳ hạng mục lịch sử nào khác trong tập hợp.

```
invariant(HIST x; HXIT<T> e)
    where x.contains(e) {
  x.latest.validTime.high.greaterOrEqual(e.validTime.high);
};
```

4.2.3 Loại trừ hạng mục sớm nhất : HIST<T>

Định nghĩa: Lịch sử được dẫn xuất mà có hạng mục sớm nhất bị loại trừ.

```
invariant(HIST x)
    where x.nonNull {
  x.exceptEarliest.equal(x.except(x.earliest));
};
```

4.2.4 Loại trừ hạng mục mới nhất : HIST<T>

Định nghĩa: Lịch sử được dẫn xuất mà có hạng mục mới nhất bị loại trừ.

```
invariant(HIST x)
    where x.nonNull {
  x.exceptLatest.equal(x.except(x.latest));
```

```
};
```

4.2.5 Chuyển đổi giảm Lịch sử thành Hạng mục lịch sử đơn nhất : HXIT<T>

Một chuyển đổi kiểu giữa toàn bộ lịch sử HIST và một hạng mục lịch sử đơn nhất HIST. Chuyển đổi này lấy dữ liệu mới nhất từ lịch sử.

Mục đích của chuyển đổi này là để cho phép một người sản xuất thông tin tạo ra một lịch sử của giá trị bất kỳ thay vì chỉ gửi một giá trị. Một người tiêu thụ thông tin, là người không kỳ vọng lịch sử mà chỉ một giá trị đơn giản, sẽ chuyển đổi lịch sử thành giá trị mới nhất.

Từ định nghĩa của HXIT, lưu ý rằng kiểu dữ liệu HXIT chuyên biệt hóa T về mặt ngữ nghĩa. Điều này có nghĩa là người tiêu thụ thông tin, khi đang kỳ vọng một T nhưng lại nhận được một HXIT, sẽ không nhận ra bất kỳ sự khác biệt nào (khả năng thay thế của các chuyên biệt hóa).

4.3 Giá trị không chắc chắn - Có xác suất (UVP) chuyên biệt hóa T

Định nghĩa: Một phần mở rộng kiểu dữ liệu chung được dùng để chỉ một xác suất thể hiện niềm tin của người sản xuất thông tin về tính đúng đắn của giá trị thông tin được đưa ra.

Làm thế nào để có được số xác suất nằm ngoài phạm vi của đặc tả này.

Xác suất có tính chủ quan và (như bất kỳ giá trị dữ liệu nào) phải được giải thích trong ngữ cảnh riêng của nó, ví dụ, khi có thông tin mới được tìm thấy xác suất có thể thay đổi. Vì vậy, đối với bất kỳ bản tin nào (tài liệu, hay các hình thức biểu diễn thông tin khác) thông tin — và đặc biệt là xác suất — thể hiện rằng điều mà người sản xuất thông tin tin tưởng là đúng đắn cho mục đích và vào thời điểm mà bản tin (tài liệu) được tạo ra.

Ví dụ, vào thời điểm bắt đầu mùa bóng chày 2000 (tháng Năm), nhà cái Las Vegas cho rằng đội New York Yankees có xác suất chiến thắng Giải Thế Giới là 1 trên 10 (0.100). Vào thời điểm viết đặc tả này, cả đội Yankees và Mets đều đã giành chiến thắng trong các giải đấu của họ, nhưng Giải Thế Giới vẫn chưa bắt đầu. Xác suất để đội Yankees chiến thắng Giải Thế Giới hiển nhiên cao hơn rất nhiều vào thời điểm này, có lẽ là 6 trên 10 (0.600). Bối cảnh, và đặc biệt là thời điểm trong năm, tạo ra rất nhiều khác biệt trên thế giới.

Do xác suất là thước đo niềm tin có tính chủ quan, xác suất có thể được trình bày mà tự thân nó không là "đúng" hay "sai", chưa nói tới "chính xác" hay "không chính xác". Đáng chú ý là, không cần phải tiến hành thí nghiệm để đo tần suất của một kết quả nào đó để xác định xác suất. Trên thực tế, bất kỳ khi nào có tuyên bố về cá nhân hay sự kiện nào đó được đưa ra, đều không thể xác nhận xác suất bằng các thí nghiệm theo trường phái xác suất dựa trên tần số như vậy.

Quay lại ví dụ của chúng ta, nhà cái Las Vegas không thể khăng khăng bắt đội Yankees và Mets chơi đủ 1000 trận đấu thử trước Giải Thế Giới; ngay cả nếu họ có thể, họ cũng sẽ không có sự nhiệt tình của giải đấu thực và do đó không chính xác. Thay vào đó, nhà cái phải tính xác suất từ lịch sử quá khứ, thống kê về cầu thủ, chấn thương, vv.

```
template<ANY T>
type UncertainValueProbabilistic<T> alias UVP<T> specializes T {
  REAL  probability;
};
```

Kiểu dữ liệu T không chính thức bị ràng buộc. Về lý thuyết, chỉ có thể nêu xác suất rời rạc cho các giá trị dữ liệu rời rạc. Vì vậy, thông thường không nên sử dụng *UVP* với các giá trị REAL, PQ, hay MO.

4.3.1 Xác suất : REAL

Định nghĩa: Xác suất được gán cho giá trị, một số thập phân giữa 0 (rất không chắc chắn) và 1 (chắc chắn), bao gồm cả 0 và 1.

```
invariant(UVP<T> x)
    where x.nonNull.and(x.probability.nonNull) {
  ((IVL<REAL>)[0;1]).contains(x.probability);
};
```

Không có "xác suất mặc định" để có thể dùng làm giá trị giả định khi không có xác suất được nêu. Vì vậy, không thể tạo bất kỳ sự khác biệt ngữ nghĩa nào giữa một giá trị *UVP* không có xác suất và một giá trị T đơn giản. Giá trị *UVP* không có nghĩa là "không chắc chắn", và một giá trị T đơn giản không có nghĩa là "chắc chắn". Trên thực tế, xác suất của *UVP* có thể là 0,999 hay 1, tức là khá chắc chắn, trong khi một giá trị T đơn giản có thể là một phỏng đoán rất mơ hồ.

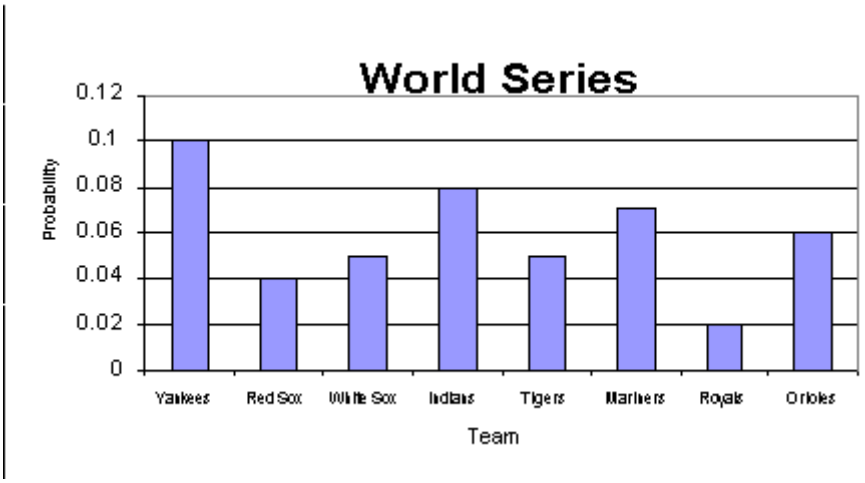
4.4 Phân bổ xác suất phi tham số (NPPD) chuyên biệt hóa SET<UVP>

Định nghĩa: Một tập hợp các giá trị UVP có xác suất (còn được gọi là biểu đồ tần suất). Tất cả các phần tử trong tập hợp đều được xem là lựa chọn thay thế và được đánh giá riêng biệt với xác suất của nó thể hiện niềm tin (hay tần suất) mà mỗi giá trị nhất định nắm giữ.

Mục đích của *NPPD* chủ yếu là để hỗ trợ báo cáo dữ liệu thống kê do nó xảy ra trong các phép đo được lấy từ nhiều đối tượng và được tổng hợp trong một biểu đồ tần suất. Điều này được thực hiện trong dịch tễ học, thú y, phòng xét nghiệm, và cả kiểm soát chi phí và thiết kế quy trình kinh doanh.

Về mặt ngữ nghĩa, thông tin của một giá trị được nêu tồn tại trái ngược với tập hợp bổ sung các giá trị có thể không được nêu. Vì vậy, về mặt ngữ nghĩa, một *NPPD* chứa *tất cả* các giá trị có thể và gán xác suất cho mỗi giá trị trong số đó.

Cách dễ dàng nhất để hình dung điều này là một biểu đồ hình cột như dưới đây



Ví dụ về biểu đồ tần suất

Ví dụ này minh họa xác suất chiến thắng tại Giải Thế Giới của các đội bóng chày chính được lựa chọn có tham gia giải đấu (trước khi mùa giải bắt đầu). Mỗi đội là loại trừ lẫn nhau, và nếu chúng ta gộp tất cả các đội, tổng xác suất sẽ bằng 1 (nghĩa là, chắc chắn rằng một trong số các đội này sẽ chiến thắng).

LƯU Ý: Mặc dù về mặt ngữ nghĩa *NPPD* gán xác suất cho tất cả các giá trị có thể, không phải tất cả các giá trị đều cần được biểu diễn rõ ràng. Các giá trị có thể mà không được đề cập tới sẽ có xác suất còn lại được phân bổ đồng đều cho tất cả các giá trị không được đề cập. Ví dụ, nếu tập hợp giá trị là {A; B; C; D} nhưng giá trị *NPPD* chỉ nêu {(B; 0.5); (C; 0.25)} thì xác suất còn lại là 1 - 0.75 = 0.25, xác suất này được phân bổ đồng đều cho toàn bộ tập hợp bổ sung: {(A; 0.125); (D; 0.125)}. Về mặt ngữ nghĩa, *NPPD* là hợp của phân bổ xác suất được nêu và phần bổ sung không được nêu với xác suất còn lại được phân bổ đồng đều.

```
template<ANY T>
type NonParametricProbabilityDistribution<T> alias NPPD<T>
    specializes SET<UVP<T>> {
  SET<UVP<T>> mostLikely(INT n);
};
```

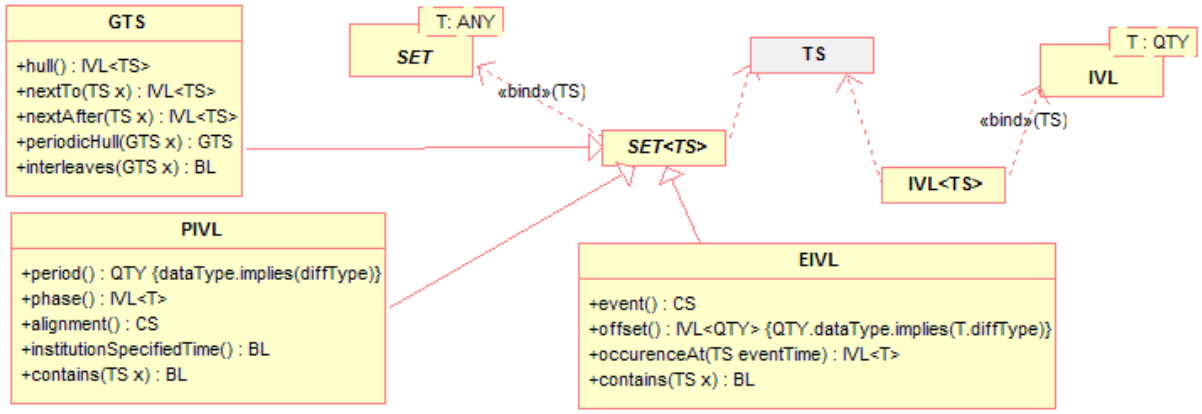
Cũng giống như kiểu dữ liệu UVP, kiểu dữ liệu T không bị ràng buộc chính thức, mặc dù có cả những trường hợp sử dụng hợp lý và thiếu hợp lý. Thông thường, kiểu dữ liệu *NPPD* sẽ được sử dụng cho các kiểu dữ liệu không có thứ tự, nếu chỉ một tập hợp "nhỏ" các giá trị có thể được gán xác suất rõ ràng, hoặc nếu phân bổ xác suất

không thể (hoặc không nên) được cho xấp xỉ với các phương pháp tham số. Trong các trường hợp khác, có thể tham khảo kiểu dữ liệu PPD.

4.4.1 Có khả năng nhất : UVP

```
invariant(NPPD<T> x)
    where x.nonNull {
  x.notEmpty;
  x.contains(x.mostLikely(n));
  x.mostLikely(n).
  forall(UVP<T> d, e; SET<UVP<T>> m; INT n)
      where x.contains(d).and(m.equal(x.mostLikely(n)))
          .and(m.contains(e)) {
    e.greaterOrEqual(d).or(m.contains(d));
    };
};
```

5. Đặc tả định thời gian



Bộ đặc tả định thời gian của các kiểu dữ liệu được sử dụng để chỉ rõ việc định thời gian phức tạp của các sự kiện và hành động xảy ra trong quản lý thứ tự và các hệ thống lập kế hoạch. Nó cũng hỗ trợ các mẫu hiệu lực theo chu kỳ có thể tồn tại cho một số loại thông tin nhất định, như số điện thoại (tối, ngày), giờ làm việc, và địa chỉ cho khái niệm được gọi bằng thuật ngữ "chim tuyết" (chỉ những người chọn sống gần đường xích đạo vào mùa đông và xa đường xích đạo vào mùa hè).

Các kiểu dữ liệu đặc tả định thời gian bao gồm thời điểm (TS), khoảng thời gian (IVL<TS>) và bổ sung các kiểu dữ liệu đặc biệt phù hợp cho những lịch trình lặp đi lặp lại. Các kiểu bổ sung này bao gồm PIVL, EIVL, và cuối cùng là bản thân kiểu dữ liệu GTS. Tất cả các kiểu định thời gian này mô tả phân bổ thời gian của các trạng thái hay sự kiện lặp đi lặp lại.

5.1 Khoảng thời gian định kỳ (PIVL) chuyên biệt hóa SET

Định nghĩa: Một khoảng thời gian định kỳ tái diễn. Kiểu dữ liệu *PIVL* có hai đặc tính, pha và chu kỳ. Pha (phrase) xác định "nguyên mẫu khoảng" được lặp lại sau mỗi chu kỳ (period).

Bảng 43. Tóm tắt đặc tính của Khoảng thời gian định kỳ

Tên	Kiểu	Mô tả
Phase (pha)	IVL<T>	Nguyên mẫu của khoảng lặp lại, xác định khoảng thời gian của mỗi lần xảy ra và thiết lập chuỗi giá trị *PIVL* tại một thời điểm nhất định.
Period (chu kỳ)	T.diff	Khoảng thời gian xác định số đo tần suất mà giá trị *PIVL* được lặp đi lặp lại.
Alignment (căn chỉnh theo lịch)	CS	Xác định xem các lần lặp lại có căn chỉnh theo các chu kỳ của lịch được không và theo cách nào (ví dụ, để phân biệt mỗi 30 ngày kể từ "ngày mùng 5 hàng tháng"). Một giá trị *PIVL* không căn chỉnh theo lịch xảy ra không phụ thuộc vào lịch. Còn một giá trị *PIVL* căn chỉnh theo lịch được đồng bộ hóa với lịch.
institutionSpecified (Do tổ chức quy định)	BL	Chỉ ra liệu việc định thời gian chính xác có phụ thuộc vào bên thực hiện lịch hay không (ví dụ, để phân biệt "mỗi 8 tiếng" với "3 lần một ngày").

Ví dụ, "mỗi 8 tiếng trong 2 phút" là một giá trị *PIVL* trong đó IVL.width của khoảng bằng 2 phút và chu kỳ tái diễn của khoảng là 8 tiếng.

pha cũng đánh dấu thời điểm thiết lập của toàn bộ chuỗi các khoảng lặp đi lặp lại theo định kỳ. Sự tái diễn của giá trị *PIVL* không có điểm bắt đầu hay điểm kết thúc, mà là vô hạn cả trong quá khứ và tương lai.

```
template<TS T>
protected type PeriodicInterval<T> alias PIVL<T>
        specializes  SET<T> {
        T.diff      period;
        IVL<T>      phase;
        CS          alignment;
        BL          institutionSpecifiedTime;
        BL          contains(TS);
  literal  ST;
```

```
};
```

Một giá trị *PIVL* được xác định đầy đủ khi cả chu kỳ và pha được xác định đầy đủ. Khoảng có thể chỉ được xác định một phần khi IVL.width hoặc khi một ranh giới được xác định.

Ví dụ: "mỗi 8 tiếng trong hai phút" chỉ xác định chu kỳ và IVL.width của pha nhưng không xác định ranh giới của pha. Ngược lại, "mỗi tám tiếng bắt đầu lúc 4 giờ" chỉ xác định chu kỳ và IVL.low của pha nhưng không xác định IVL.high của pha. "Mỗi tám tiếng trong hai phút bắt đầu lúc 4 giờ" xác định đầy đủ giá trị chu kỳ, và cả IVL.low và IVL.width của pha.

Kiểu dữ liệu *PIVL* là một phần mở rộng kiểu chung với tham số kiểu T được giới hạn trong kiểu dữ liệu TS và các phần mở rộng của nó. Kiểu dữ liệu PPD<TS>> là một phần mở rộng của TS và do đó có thể được sử dụng để hình thành các giá trị PIVL<PPD<TS>>.

Thông thường, các lịch trình lặp lại chỉ được xác định xấp xỉ. Ví dụ, "ba lần một ngày với mỗi lần mười phút" đôi khi không có nghĩa là một chu kỳ chính xác 8 tiếng và thường xuyên không có nghĩa là các khoảng chính xác 10 phút. Thực ra là khoảng cách giữa mỗi lần xảy ra có thể dao động tối đa trong khoảng 3 và 12 tiếng và IVL.width của khoảng có thể ít hơn 5 phút hoặc nhiều hơn 15 phút. Giá trị PIVL<PPD<TS>> có thể được sử dụng để chỉ ra mức chậm trễ cho phép hay cho biết đặc tả "đề cao vấn đề định thời gian" đến đâu.

5.1.1 Pha : IVL<T>

Định nghĩa: Nguyên mẫu của khoảng lặp lại, xác định khoảng thời gian của mỗi lần xảy ra và thiết lập chuỗi *PIVL* tại một thời điểm nhất định.

pha cũng đánh dấu thời điểm thiết lập của toàn bộ chuỗi các khoảng lặp đi lặp lại theo định kỳ. Sự tái diễn của chuỗi *PIVL* không có điểm bắt đầu hay điểm kết thúc, mà là vô hạn cả trong quá khứ và tương lai. Giá trị IVL.width của *pha* phải nhỏ hơn hoặc bằng chu kỳ.

```
invariant (PIVL<T> x)
    where x.nonNull {
  x.phase.nonNull.implies(x.phase.width.lessOrEqual(x.period));
};
```

5.1.2 Chu kỳ : T.diff

Định nghĩa: Khoảng thời gian xác định số đo tần suất mà giá trị *PIVL* được lặp đi lặp lại.

chu kỳ là một giá trị QTY theo đơn vị thời gian (T.diff). Đối với một giá trị *PIVL* không chắc chắn, *chu kỳ* là một phân bổ xác suất trên thời gian đã qua.

```
invariant(PIVL<T> x)
    where x.nonNull {
  x.period.nonNull;
};

invariant(PIVL x) {
  x.period.dataType.implies(PQ);
};
```

5.1.3 Căn chỉnh theo lịch : CS

Định nghĩa: Xác định xem các lần lặp lại có căn chỉnh theo các chu kỳ của lịch được không và theo cách nào (ví dụ, để phân biệt mỗi 30 ngày kể từ "ngày mùng 5 hàng tháng"). Một giá trị *PIVL* không căn chỉnh theo lịch xảy ra không phụ thuộc vào lịch. Còn một giá trị *PIVL* căn chỉnh theo lịch được đồng bộ hóa với lịch.

Ví dụ, "ngày mùng 5 hàng tháng" là một *PIVL* được căn chỉnh theo lịch. chu kỳ dao động từ 28 đến 31 ngày tùy thuộc vào tháng trong lịch. Ngược lại, "mỗi 30 ngày" là một chu kỳ độc lập mà mỗi tháng sẽ rơi vào một ngày khác nhau.

Căn chỉnh theo lịch xác định một chu kỳ lịch theo đó PIVL được căn chỉnh. Dòng chảy đều đặn của thời gian khi đó sẽ bị ngăn chia bởi các chu kỳ lịch. Sự ngăn chia này được gọi là "lưới" lịch được tạo ra bởi chu kỳ lịch cần căn chỉnh theo. Ranh giới của mỗi khoảng xảy ra khi đó sẽ có khoảng cách bằng nhau tính từ điểm đầu tiên trong mỗi ngăn. Nói cách khác, khoảng cách từ dây lưới thấp tiếp theo đến điểm bắt đầu của khoảng là một hằng số.

Ví dụ, với "ngày mùng 5 hàng tháng" chu kỳ lịch căn chỉnh sẽ là tháng trong năm (MY). Dòng chảy đều đặn của thời gian bị ngăn chia thành các tháng trong năm. Khoảng cách giữa điểm bắt đầu của mỗi tháng và điểm bắt đầu của khoảng xảy ra của nó là 4 ngày (4 ngày vì ngày trong tháng (DM) bắt đầu đếm từ 1.) Vì vậy, do các tháng khác nhau về số ngày trong tháng, khoảng cách giữa các khoảng tái diễn sẽ hơi khác nhau, sao cho khoảng luôn xảy ra vào ngày thứ 5.

5.1.4 Định thời gian do tổ chức quy định : BL

Định nghĩa: Chỉ ra liệu việc định thời gian chính xác có phụ thuộc vào bên thực hiện lịch hay không (ví dụ, để phân biệt "mỗi 8 tiếng" với "3 lần một ngày").

Ví dụ, với lịch trình "ba lần một ngày" thời gian trung bình giữa các lần lặp lại là 8 tiếng, tuy nhiên, nếu chỉ báo thời gian do tổ chức quy định là *đúng*, thì việc định thời gian có thể tuân theo một quy tắc nào đó do cá nhân hoặc tổ chức thực hiện ("tổ chức") đặt ra, ví dụ, lịch trình 3 lần một ngày được thực hiện vào lúc 7 giờ sáng, trưa, và 7 giờ tối.

5.1.5 Dạng chữ

Dạng chữ chung. Dạng chữ chung cho khoảng thời gian định kỳ là như sau:

(*pha* : IVL<T>(/ (*chu kỳ* : QTY ([**@** (*căn chỉnh*( ] [**IST**].

```
PIVL<T>.literal ST {
  PIVL<T>    : S2          { $.equal($1); }
           | S2 "IST"     { $.phase.equal($1.phase);
                            $.period.equal($1.period);
                            $.institutionSpecified.equal(true); };
  PIVL<T> S2 : S1                { $.equal($1); }
           | S1 "@" "(" ST ")"   { $.phase.equal($1.phase);
                                   $.period.equal($1.period);
                                   $.alignment.equal($4); };
  PIVL<T> S1 :
    IVL<T> "/" "(" QTY ")"       { $.phase.equal($1);
                                   $.period.equal($3); }
           | "/" "(" QTY ")"     { $.period.equal($2); };
};
```

Ví dụ, "[200004181100;200004181110]/(7 d)@DW" xác định mọi thứ ba từ 11:00 đến 11:10 sáng. Ngược lại, "[200004181100;200004181110]/(1 mo)@DM" xác định mọi ngày 18 trong tháng từ 11:00 đến 11:10 sáng.

Xem Bảng 36 để biết mã chu kỳ lịch được định nghĩa cho lịch Dương. Có các ký hiệu 1 ký tự và 2 ký tự. Ký hiệu 2 ký tự được ưa chuộng hơn cho việc căn chỉnh.

Dạng mẫu lịch. Dạng này được dùng để xác định định thời gian căn chỉnh theo lịch theo hướng trực quan hơn nhờ sử dụng các "mẫu lịch." Cú pháp mẫu lịch được định nghĩa (bán chính thức) như sau:

(*anchor*([(*calendar digits*([**..** (*calendar digits*( ]] / (*number* : INT([**IST**]

Một mẫu lịch là một ngày lịch trong đó chữ số quan trọng cao hơn (ví dụ, năm và tháng) bị bỏ qua. Để giải thích các chữ số, một định danh chu kỳ được thêm vào làm tiền tố để xác định chu kỳ lịch của các chữ số nằm ngoài cùng bên trái. Định danh chu kỳ lịch này *neo* các chữ số lịch tiếp theo về bên phải.

Xem Bảng 36 để biết mã chu kỳ lịch được định nghĩa cho lịch Dương. Có các ký hiệu 1 ký tự và 2 ký tự. Ký hiệu 1 ký tự được ưa chuộng hơn cho việc thiết lập mẫu lịch.

Ví dụ: "M0219" là toàn bộ ngày 19 tháng Hai hàng năm. Khoảng định kỳ này có ngày 19 tháng Hai của năm bất kỳ là pha của nó (ví dụ, "[19690219;19690220["),

chu kỳ một năm, và căn chỉnh tháng trong năm (M). Chu kỳ lịch căn chỉnh giống như thiết lập (ví dụ, tháng trong năm).

Các chữ số lịch cũng có thể bỏ qua các chữ số nằm về bên phải. Khi các chữ số bên phải được bỏ qua, điều này có nghĩa là khoảng từ thấp nhất tới cao nhất cho các chữ số này. Ví dụ, "M0219" là toàn bộ ngày 19 tháng Hai; "M021918" là ngày 19 tháng Hai, toàn bộ một tiếng từ 6 đến 7 giờ tối.

Khi không có định nghĩa chính thức cho điều này, các quy tắc để phân tích một mẫu lịch là như sau (ví dụ với "M021918..21"):

1. đọc định danh chu kỳ anchor (ví dụ, "M")
2. căn chỉnh bằng với chu kỳ lịch này (ví dụ, tháng trong năm)
3. sử dụng thời điểm hiện tại và định dạng một dạng chữ chính xác với chu kỳ lịch quan trọng cao hơn tiếp theo từ chu kỳ lịch anchor (ví dụ, năm, "2000", dựng "2000021918"); đây là "dạng chữ gốc"
4. đọc dạng chữ vừa được dựng này (ví dụ, "2000021918") thành một giá trị TS và chuyển đổi giá trị đó thành một khoảng dựa theo IVL_TS.promotionTS (ví dụ, "[2000021918;2000021919[") đây là "khoảng thấp."
5. nếu có thẻ toán tử bao (hull) ".." đi sau, đọc các chữ số lịch tiếp sau (ví dụ, "21")
6. căn chỉnh phải dạng chữ gốc và các chữ số lịch vừa đọc
7. "2000021918"
8. " 21"
9. rồi sao chép tất cả các chữ số từ dạng chữ gốc còn thiếu sang bên trái các chữ số lịch vừa đọc (ví dụ, có được "2000021921".)
10. đọc dạng chữ vừa được dựng này (ví dụ, "2000021918") thành một giá trị TS và chuyển đổi giá trị đó thành một giá trị IVL<TS> dựa theo IVL_TS.promotionTS (ví dụ, "[2000021921;2000021922[") đây là "khoảng cao."
11. pha là bao lồi của khoảng thấp và khoảng cao (ví dụ, "[2000021918;2000021922[").
12. nếu không có toán tử bao, pha đơn giản là khoảng thấp.

Xen kẽ. Một mẫu lịch theo sau bởi một dấu gạch chéo và một số nguyên *n* chỉ ra rằng mẫu lịch đã cho được áp dụng mỗi lần thứ n.

Ví dụ: "D19/2" là ngày 19 của mỗi tháng thứ hai.

Một biểu thức mẫu lịch được đánh giá tại thời điểm mẫu được gặp lần đầu tiên. Vào thời điểm này, các chữ số lịch bị thiếu ở bên trái được hoàn thiện bằng cách sử dụng ngày sớm nhất khớp với mẫu (và theo một mẫu trước đó với sự kết hợp các tập hợp thời gian).

Ví dụ: "D19/2" là ngày 19 của mỗi tháng thứ hai. Nếu biểu thức này được đánh giá vào ngày 14 tháng Ba năm 2000, pha sẽ được hoàn thiện thành: "[20000319;20000320[/(2 mo)@DM" và do đó chu kỳ hai tháng bắt đầu với ngày 19 tháng Ba, tiếp sau là ngày 19 tháng Năm, vv. Nếu biểu thức được đánh giá vào ngày 20 tháng Ba, chu kỳ sẽ bắt đầu vào ngày 19 tháng Tư, tiếp sau là ngày 19 tháng Sáu, v.v.

Nếu không có chữ số lịch nào tiếp sau định danh chu kỳ lịch, mẫu sẽ khớp với ngày bất kỳ. Số nguyên đứng sau dấu gạch chéo chỉ ra độ dài của chu kỳ. pha trong các trường hợp này chỉ có độ rộng được xác định là khoảng thời gian của chu kỳ lịch anchor (trong ví dụ này là 1 ngày).

Ví dụ: "CD/2" là cứ mỗi hai ngày, "H/8" là cứ mỗi tiếng thứ 8, trong khoảng thời gian một tiếng.

Thời gian do tổ chức quy định. Cả dạng chữ *PIVL* và mẫu lịch đều có thể được tiếp sau bởi ba chữ cái "IST" để chỉ rằng trong chu kỳ lịch lớn hơn (ví dụ, đối với "giờ trong ngày" chu kỳ lịch lớn hơn là "ngày") các sự kiện lặp lại được định vào thời gian do tổ chức quy định. Dạng này được sử dụng để xác định những lịch trình như "ba lần một ngày" trong đó chu kỳ giữa hai sự kiện tiếp nối có thể dao động giữa 4 tiếng (giữa bữa sáng và bữa trưa) và 10 tiếng (qua đêm).

Bảng 44. Ví dụ về biểu thức dạng chữ cho các PIVL

Dạng chung	Dạng mẫu lịch	Mô tả
[198709;198710[/(1 a)@MY	M09	tháng Chín, cả tháng, mỗi năm (lưu ý rằng trong năm 1987 dạng chung là không phù hợp do khoảng định kỳ tái diễn mỗi năm trong cả quá khứ và tương lai).
[19870915;19870916[/(1 a)@DM	M0915	ngày 15 tháng Chín, cả ngày, mỗi năm
[1987091516;1987091517[/(1 a)@DM	M091516	ngày 15 tháng Chín lúc 4 giờ chiều, cả tiếng, mỗi năm
[198709151630;198709151710]/(1 a)@DM	M09151630..1710	ngày 15 tháng Chín lúc 4:30 5:10 chiều, mỗi năm

Dạng chung	Dạng mẫu lịch	Mô tả
[1987091516;[/(1 a)@DM		ngày 15 tháng Chín lúc 4 giờ chiều, thời gian kết thúc không biết rõ, mỗi năm
[198709151630;198709151631[/(1 a)@DM	M09151630	ngày 15 tháng Chín lúc 4:30 chiều, cả phút, mỗi năm
[1987091516;1987091517[/(1 mo)@DM	D1516..17	ngày 15 mỗi tháng lúc 4 đến 5 giờ chiều
[1987091516;1987091517[/(1 mo)		ngày 15 tháng Chín năm 1987 từ 4 đến 5 giờ chiều và sau đó cứ mỗi 730,5 tiếng liên tục (ví dụ này ít có giá trị thực tiễn ngoài việc so sánh dạng không được căn chỉnh với dạng được căn chỉnh trong hàng phía trước).
[1987091516;1987091517[/(1 mo)@HD		ngày 15 tháng Chín năm 1987 từ 4 đến 5 giờ chiều và sau đó cứ mỗi 30,4375 ngày, nhưng được căn chỉnh với giờ trong ngày.
[1 mo]/(2 mo)@MY	M/2	cứ mỗi hai tháng một trong năm; (tháng Một, Ba, ...) hay (tháng Hai, Tư, ...) không được xác định
[198701;197502[/(2 mo)@MY	M01..12/2	cứ mỗi hai tháng một trong năm, tháng Một, Ba, ...
[198702;197503[/(2 mo)@MY	M02..12/2	cứ mỗi hai tháng một trong năm, tháng Hai, Tư, ...
[19870401;19870930[/(1 a)@DM	M04..09	ngày 1 tháng Tư cho tới (và

Dạng chung	Dạng mẫu lịch	Mô tả
		bao gồm) ngày 30 tháng Chín
19870401-0930/(1 a)@DM	M0401..0930	ngày 1 tháng Tư tới ngày 30 tháng Chín (dạng chung sử dụng dạng gạch ngang cho khoảng pha)
[20001202;20001203[/(1 wk)@DW	J6	mọi thứ bảy
[20001202;20001203[/(2 wk)@DW	J6/2	cứ hai tuần lại tính một thứ bảy
[20001202;20001203[/(3 wk)@DW	J6/3	mọi thứ bảy thứ ba trong tháng
[1 d]/(2 d)@DW	J/2	cứ hai ngày một trong tuần; (thứ Hai, Tư, Sáu, ...) hay (thứ Ba, Năm, Bảy, ...) không được xác định
[20001204;20001205[/(2 d)@DW	J2..6/2	cứ hai ngày một trong tuần (thứ Ba, Năm, Bảy, Ba, Năm, Bảy, ...)
[20001204;20001205[/(2 d)	D/2	cứ hai ngày một (thứ Ba, Năm, Bảy, Hai, Tư, Sáu, Chủ nhật, Ba, ...)
[19870601;19870606[/(1 wk)@DW	J1..5	thứ Hai đến thứ Sáu mỗi tuần
[19870601;19870608[/(2 wk)	W/2	cứ hai tuần một (liên tục)
[19870101;19870105[/(2 wk)@WY	WY/2	cứ hai tuần một trong năm (một ví dụ thiếu sắc bén về tác động của căn chỉnh lịch: khoảng pha chỉ kéo dài 4 ngày nhưng nó biểu diễn cả một tuần trong căn chỉnh lịch "tuần trong năm").

Dạng chung	Dạng mẫu lịch	Mô tả
[19870406;19870413[/(1 a)@WY	WY15	tuần lịch thứ 15 của mỗi năm
[19870105;19870112[/(1 mo)@WM	WM2	tuần thứ hai của tháng, mỗi tháng
[19870508;19870509[/(1 a)@DY	DY128	ngày thứ 128 của năm, mỗi năm
[10 min]/(2 d)		cứ hai ngày một trong 10 phút (chỉ biết độ rộng của khoảng lặp lại)
[1 h]/(8 h)	H/8	cứ mỗi tiếng thứ 8 (mỗi lẫn một khoảng 60 phút)
[1 h]/(8 h) IST	H/8 IST	ba lần một ngày theo thời gian do tổ chức quy định (mỗi lần một khoảng 60 phút)
/(8 h) IST		ba lần một ngày theo thời gian do tổ chức quy định. Không biết gì về khoảng lặp lại, nghĩa là chỉ bao gồm một chu kỳ (tần suất), trong khi pha không được định nghĩa

5.1.6 Khoảng định kỳ như tập hợp

Đặc tính cần thiết của một tập hợp là nó có chứa phần tử. Đối với các giá trị *PIVL* không được căn chỉnh theo lịch, đặc tính *chứa* được định nghĩa như sau. Một giá trị TS *t* được chứa trong kiểu dữ liệu *PIVL* khi và chỉ khi có một số nguyên *i* theo đó *t* cộng số lần chu kỳ *i* là một phần tử của pha.

```
invariant (PIVL<TS> x, TS t)
    where x.nonNull.and(x.alignment.isNull) {
  x.contains(t).equal(exists(INT i) {
    x.phase.contains(t.plus(x.period.times(i)));
    });
```

```
};
```

Đối với các giá trị *PIVL* được căn chỉnh theo lịch, đặc tính *chứa* được định nghĩa sử dụng đặc tính tổng (*t, n*) của chu kỳ lịch, cộng *n* chu kỳ lịch này với thời gian *t*.

```
invariant (PIVL<TS> x, TS t, CalendarCycle c)
    where x.nonNull.and(c.equal(x.alignment)) {
  x.contains(t).equal(exists(INT i) {
    x.phase.contains(c.sum(t, i));
    });
};
```

5.2 Khoảng thời gian định kỳ liên quan đến sự kiện (EIVL) chuyên biệt hóa SET

Định nghĩa: Xác định khoảng thời gian định kỳ trong đó mỗi lần tái diễn dựa trên các hoạt động của cuộc sống hàng ngày hoặc các sự kiện quan trọng khác có liên quan đến thời gian nhưng không được xác định hoàn toàn bởi thời gian.

Ví dụ, "một tiếng sau bữa sáng" xác định điểm bắt đầu của khoảng là một tiếng sau khi bữa sáng kết thúc. Bữa sáng được giả định diễn ra trước bữa trưa nhưng không được xác định sẽ xảy ra tại thời gian cụ thể bất kỳ.

```
template<TS T>
protected type EventRelatedPeriodicInterval<T> alias EIVL<T>
        specializes  SET<T>{
        CS          event;
        IVL<PQ>      offset;
        IVL<T>       occurrenceAt(TS eventTime);
        BL          contains(TS);
 literal  ST;
};
```

5.2.1 Sự kiện : CS

Định nghĩa: Một mã cho một hoạt động phổ biến (định kỳ) trong cuộc sống hàng ngày dựa trên đó khoảng định kỳ liên quan đến sự kiện được xác định.

Các sự kiện như vậy đủ điều kiện để được áp dụng trong miền của thuộc tính này mà đối với nó tất cả những điều sau đây là đúng:

- sự kiện xảy ra một cách phổ biến và thường xuyên
- sự kiện đang được sử dụng cho các hoạt động định thời gian, và
- sự kiện không được xác định hoàn toàn bởi thời gian

Bảng 45. Miền TimingEvent

mã	tên	định nghĩa
AC	AC	trước bữa ăn (từ tiếng Latinh. ante cibus)
ACD	ACT	trước bữa trưa (từ tiếng Latinh. ante cibus diurnus)
ACM	ACM	trước bữa sáng (từ tiếng Latinh. ante cibus matutinus)
ACV	ACV	trước bữa tối (từ tiếng Latinh. ante cibus vespertinus)
HS	HS	giờ đi ngủ
IC	IC	giữa các bữa ăn (từ tiếng Latinh. inter cibus)
ICD	ICD	giữa bữa trưa và bữa tối
ICM	ICM	giữa bữa sáng và bữa trưa
ICV	ICV	giữa bữa tối và giờ đi ngủ
PC	PC	sau bữa ăn (từ tiếng Latinh. post cibus)
PCD	PCD	sau bữa trưa (từ tiếng Latinh. post cibus diurnus)
PCM	PCM	sau bữa sáng (từ tiếng Latinh. post cibus matutinus)
PCV	PCV	sau bữa tối (từ tiếng Latinh. post cibus vespertinus)

5.2.2 Khoảng chênh thời gian : IVL<PQ>

Định nghĩa: Một khoảng thời gian trôi qua (khoảng thời gian, không phải thời điểm tuyệt đối) đánh dấu khoảng chênh thời gian cho khởi đầu, độ rộng và kết thúc của một giá trị *EIVL* được đo từ thời điểm mỗi sự kiện như vậy thực sự xảy ra.

Ví dụ: nếu đặc tả là "một tiếng trước bữa sáng trong 10 phút", IVL.low của *khoảng chênh thời gian* là 1 tiếng và IVL.width của *khoảng chênh thời gian* là 10 phút.

5.2.3 Dạng chữ

Dạng chữ cho giá trị *EIVL* bắt đầu với mã sự kiện tiếp đến là một khoảng tùy chọn mức chênh lệch thời gian.

```
EIVL<TS>.literal ST {
  EIVL<TS> : event         { $.event.equal($1); }
        | event offset   { $.event.equal($1);
                           $.offset.equal($2); };
  CS event : ST            { $.code.equal($1);
                           $.codeSystem.equal(2.16.840.1.113883.5.1019); }
  IVL<PQ> offset
        : "+" IVL<PQ>    { $.equal($2); }
        | "-" IVL<PQ>    { $.low.equal($2.high.negate);
                           $.high.equal($2.low.negate);
                           $.width.equal($2.width);
                           $.lowClosed($2.highClosed);
                           $.highClosed($2.lowClosed); };
};
```

Ví dụ, một tiếng sau bữa ăn sẽ là "PC+[1h;1h]". Một tiếng trước giờ đi ngủ trong 10 phút sẽ là: "HS-[50min;1h]".

5.2.4 Quyết định tính liên quan tới sự kiện

Một giá trị *EIVL* là một tập hợp thời gian, nghĩa là, có thể kiểm tra liệu một thời gian hay khoảng thời gian cụ thể có phải là một phần tử của tập hợp đó. Việc một giá trị *EIVL* có chứa một khoảng thời gian xác định hay không được quyết định bằng việc sử dụng một quan hệ sự kiện χ *thời gian* được gọi là EVENT(*event, time*) (SỰ KIỆN(*sự kiện, thời gian*)). Đặc tính *xảy ra tại* (*t*) là khoảng xảy ra sẽ tồn tại nếu sự kiện xảy ra tại thời gian *t*.

```
invariant(EIVL<T> x, T eventTime, IVL<T> v)
    where v.equal(x.occurrenceAt(eventTime)) {
  v.low.equal(eventTime.plus(x.offset.low));
  v.high.equal(eventTime.plus(x.offset.high));
  v.lowClosed.equal(x.offset.lowClosed);
  v.highClosed.equal(x.offset.highClosed);
};
```

Vì vậy, một giá trị *EIVL* chứa một giá trị TS *t* nếu có một thời gian sự kiện *e* với một khoảng xảy ra *v* sao cho *v* chứa *t*.

```
invariant(EIVL<T> x, T y) {
```

```
  x.contains(y).equal(exists(T e, IVL<T> v)
    where EVENT(x.event, y).and(v.resolvedAt(y)) {
      v.contains(y);
      });
};
```

5.3 Đặc tả định thời gian tổng quát (GTS) chuyên biệt hóa SET<TS>

Định nghĩa: Một giá trị <dt-TS>, xác định thời gian của các sự kiện và hành động và các mẫu hiệu lực theo chu kỳ có thể tồn tại đối với một số loại thông tin, ví dụ số điện thoại (liên lạc ban đêm, ban ngày), địa chỉ (còn gọi là "chim tuyết", sống gần đường xích đạo vào mùa đông và xa đường xích đạo vào mùa hè) và giờ làm việc.

GTS có các khía cạnh sau:

- *GTS* như một SET<TS> tổng quát. Từ khía cạnh này, *GTS* cho biết liệu có bất kỳ giá trị TS xác định nào rơi vào lịch trình do giá trị *GTS* mô tả hay không.
- *GTS* như một kết hợp của nhiều PIVL. Khía cạnh này mô tả các mẫu lặp cả đơn giản và phức tạp được xác định như thế nào với *GTS*.
- *GTS* như một bộ tạo sóng của một LIST<IVL<TS>>. Từ khía cạnh này, *GTS* có thể tạo ra tất cả các khoảng xảy ra của một sự kiện hay hành động, hoặc tất cả các giai đoạn hiệu lực cho một thực tế.
- *GTS* như một cú pháp biểu thức được định nghĩa cho một lịch. Khía cạnh này là dạng chữ của *GTS*.

Trong tất cả các trường hợp, *GTS* được định nghĩa là một SET<TS>. Sử dụng SET.union (TẬP HỢP.hợp), SET.intersection (TẬP HỢP.giao) và SET.difference (TẬP HỢP.hiệu), có thể xây dựng các SET<TS> phức tạp hơn từ những tập hợp đơn giản hơn. Sau cùng thì những khối xây dựng mà từ đó tất cả các giá trị *GTS* được tạo nên là IVL, PIVL, và EIVL. Việc xây dựng giá trị *GTS* có thể được chỉ rõ bằng dạng chữ. Không có cấu trúc kiểu dữ liệu đặc biệt nào được định nghĩa mà sẽ tạo ra một kết hợp các tập hợp thời gian đơn giản hơn từ một giá trị *GTS* cho trước. Trong khi bất kỳ triển khai nào cũng sẽ phải chứa một biểu diễn được cấu trúc như vậy, nó không cần thiết trong việc trao đổi các giá trị *GTS* với dạng chữ đã có.[56]

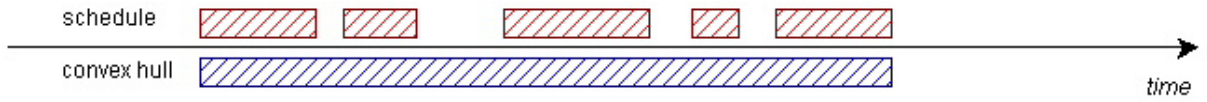
```
type GeneralTimingSpecification alias GTS specializes SET<TS> {
        IVL<TS>   hull;
        IVL<TS>   nextTo(TS x)
        IVL<TS>   nextAfter(TS x)
        GTS       periodicHull(GTS x);
        BL        interleaves(GTS x);
 demotion  LIST<IVL<TS>>;
```

```
  literal   ST;
};
```

5.3.1 Bao lồi

Bao lồi là khoảng nhỏ nhất mà là tập cha của tất cả các khoảng xảy ra. Như đã lưu ý trong SET.hull (TẬP HỢP.bao), tất cả các tập hợp được sắp xếp toàn phần đều có một bao lồi. Bởi vì *GTS* là một SET<TS>, vốn được sắp xếp toàn phần, tất cả các giá trị *GTS* đều có bao lồi.

Bao lồi của một *GTS* có thể được gọi ít chính thức hơn là "khoảng biên ngoài". Vì vậy, bao lồi của một *GTS* mô tả khởi đầu và kết thúc tuyệt đối của lịch trình lặp lại. Đối với các lặp lại vô hạn (ví dụ, một PIVL) bao lồi có biên vô hạn.



5.3.2 GTS như một Chuỗi Khoảng Lặp lại

Một giá trị *GTS* là một bộ tạo sóng của một chuỗi các khoảng thời gian mà trong đó có một sự kiện hay hoạt động xảy ra, hoặc trong đó có một trạng thái là có hiệu lực.

Đặc tính **tiếp theo tới** liên kết tới mọi thời điểm *t* tập cha liên tục lớn nhất (một "khoảng xảy ra") *v* của giá trị *GTS S*, trong đó *v* là khoảng gần nhất với *t* mà bắt đầu muộn hơn *t* hoặc chứa *t*.

```
invariant(GTS S, TS t, IVL<TS> v) {
  v.equal(S.nextTo(t)).equal(
    S.contains(o).and(
      forall(IVL<TS> u) where x.contains(u) {
        u.contains(v).implies(u.equal(v));
        })
      .and(v.contains(t).or(forall(TS i) where t.lessOrEqual(i)
          .and(i.lessThan(v.low)) {
        S.conatins(i).not; })));
};
```

Đặc tính **tiếp theo sau** liên kết tới mọi thời điểm *t* tập cha liên tục lớn nhất (một "khoảng xảy ra") *v* của giá trị *GTS S*, trong đó *v* là khoảng gần nhất với *t* mà bắt đầu muộn hơn *t*.

```
invariant(GTS S, TS t) {
  S.contains(t).not.implies(S.nextAfter(t).equal(S.nextTo(t)));
  S.contains(t).implies(S.nextAfter(t).equal(
```

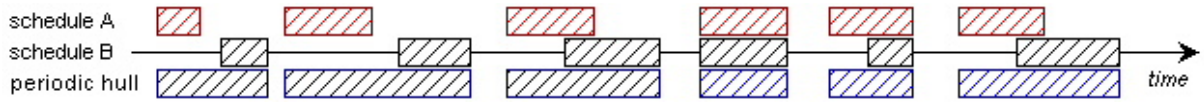
```
            S.except(nextTo(t)).nextTo(t)));
};
```

Có thể chuyển đổi một giá trị *GTS* thành một LIST<IVL<TS>>.

```
invariant(GTS x)
    where x.isEmpty {
  ((LIST<IVL<TS>>)x).isEmpty; };

invariant(GTS x, IVL<TS> first)
    where x.notEmpty.and(x.hull.low.nonNull)
                .and(first.equal(x.nextTo(x.hull.low)) {
  ((LIST<IVL<TS>>)x).head.equal(first);
  ((LIST<IVL<TS>>)x).tail.equal(
    (LIST<IVL<TS>>)x.except(first));
};
```

5.3.3 Xen kẽ Lịch trình và Bao định kỳ



Đối với hai giá trị *GTS A* và *B* chúng ta nói rằng *A* xen kẽ *B* nếu khoảng xảy ra của chúng xen kẽ trên dòng thời gian. Có thể hình dung khái niệm này qua Hình trên.

Để các giá trị *GTS A* và *B* xen kẽ, có thể sắp xếp các khoảng xảy ra của cả hai nhóm theo cặp các khoảng xảy ra tương ứng. Điều sau đây phải tiếp tục là đúng: đối với tất cả các khoảng xảy ra tương ứng $a \subseteq A$ và $b \subseteq B$, *a* bắt đầu trước khi *b* bắt đầu (hoặc cùng lúc) và *b* kết thúc sau khi *a* kết thúc (hoặc cùng lúc).

Quan hệ xen kẽ có hiệu lực khi hai lịch trình có cùng tần suất trung bình, và khi lịch trình thứ hai không bao giờ "đi nhanh hơn" lịch trình thứ nhất. Nghĩa là, không khoảng xảy ra nào trong lịch trình thứ hai có thể bắt đầu trước khoảng xảy ra tương ứng của nó trong lịch trình thứ nhất.

Với hai giá trị *GTS* xen kẽ, có thể lấy được một bao định kỳ sao cho các khoảng xảy ra của bao định kỳ là bao lồi của các khoảng xảy ra tương ứng.

Bao định kỳ có vai trò quan trọng trong việc xây dựng hai lịch trình bằng cách kết hợp các biểu thức GTS. Ví dụ, để xây dựng khoảng định kỳ từ Ngày lễ tưởng niệm chiến sỹ vong trận tới Ngày lễ lao động hàng năm, đầu tiên cần thiết lập các lịch trình *M* cho Ngày lễ tưởng niệm chiến sỹ trận vong (thứ Hai cuối cùng của tháng Năm) và *L* cho Ngày lễ lao động (thứ Hai đầu tiên của tháng Chín) rồi sau đó kết hợp hai lịch trình này sử dụng bao định kỳ của *M* và *L*.

```
invariant(GTS A, B)
```

```
      where x.nonNull.and(y.nonNull) {
    A.interleaves(B).equal(
      forall(IVL<TS> a, b, c; TS t)
        where a.equal(A.nextTo(t))
            .and(b.equal(B.nextTo(a.low)))
            .and(c.equal(A.nextTo(b.high))) {
          b.equal(B.nextTo(a.high));
          a.low.lessOrEqual(b.low);
          c.equal(A.nextTo(b.high));
          c.equal(a).or(c.equal(A.nextAfter(a.high)));
          });
};
```

Đối với hai giá trị *GTS A* và *B* trong đó *A* xen kẽ *B*, một bao định kỳ được định nghĩa là bao lồi theo cặp của các khoảng xảy ra tương ứng của *A* và *B*.

```
invariant(GTS A, B, C)
      where A.interleaves(B) {
    A.periodicHull(B).equal(C).equal(
      forall(IVL<TS> a, b; TS t)
        where a.equal(A.nextTo(t))
            .and(b.equal(B.nextTo(a.low))) {
          C.contains(c).equal(c.equal(a.hull(b)));
          });
};
```

Quan hệ xen kẽ có tính phản xạ, bất đối xứng, và không bắc cầu. Phép toán bao định kỳ có tính không giao hoán và không kết hợp.[57]

5.3.4 Dạng chữ

Dạng chữ *GTS* cho phép xác định các kết hợp IVL<TS>, PIVL, và sử dụng các phép toán tập hợp là hợp và giao.[58]

Phép toán Hợp được chỉ rõ bởi một danh sách được phân tách bằng dấu chấm phẩy. Phép toán Giao được chỉ rõ bởi một danh sách được phân tách bằng khoảng trắng. Phép toán Giao được ưu tiên hơn hợp. Hiệu tập hợp có thể được chỉ rõ bằng việc sử dụng dấu gạch chéo ngược; hiệu có mức ưu tiên trung gian, nghĩa là thấp hơn giao nhưng cao hơn hợp. Ngoài ra, có thể sử dụng ngoặc đơn để vượt quyền ưu tiên toán tử khi cần thiết.

```
GTS.literal ST {
  GTS symbol : union                { $.equal($1); }
```

```
            | exclusion                { $.equal($1); };
        SET<TS> union
            : symbol ";" intersection    { $.equal($1.union($3)); }
            | intersection               { $.equal($1); };
        SET<TS> exclusion
            : symbol "\" intersection   { $.equal($1.except($3)); }
            | intersection              { $.equal($1); };
        SET<TS> intersection
            : hull intersection   { $.equal($1.intersection($2)); }
            | hull                      { $.equal($1); };
        SET<TS> hull
            : hull ".." factor   { $.equal($1.periodicHull($3)); }
            | factor            { $.equal($1); };
        SET<TS> factor
            : IVL<TS>                  { $.equal($1); }
            | PIVL<TS>                 { $.equal($1); }
            | EIVL<TS>                 { $.equal($1); }
            | "(" GTS ")"              { $.equal($1); };
    };
```

Bảng dưới đây chứa các ví dụ kiểu mẫu cho các dạng chữ *GTS* phức tạp. Đối với các ví dụ đơn giản hơn, tham khảo dạng chữ của kiểu dữ liệu IVL, PIVL, và EIVL.

Bảng 46. Ví dụ về biểu thức dạng chữ cho Đặc tả định thời gian chung

Biểu thức dạng chữ	**Ý nghĩa**
M09 D15 H16 N30 S34.12	ngày 15 tháng Chín vào lúc 4:30:34.12 chiều là giao của nhiều khoảng thời gian định kỳ (mẫu lịch)
M0915163034.12	ngày 15 tháng Chín vào lúc 4:30:34.12 chiều là một khoảng thời gian định kỳ đơn giản (mẫu lịch)
M01; M03; M07	tháng Một, Ba, và Bảy (hợp của ba khoảng thời gian định kỳ)
M04..09 M/2	mỗi tháng thứ hai từ tháng Tư đến tháng Chín (tháng Tư, Sáu, Tám)

Biểu thức dạng chữ	Ý nghĩa
J1; J2; J4	Thứ Hai, Ba, Năm
W/2 J2	mỗi thứ Ba thứ hai của hai tuần (giao của cứ hai tuần một và mỗi thứ Ba)
1999 WY15	tuần lịch thứ 15 trong năm 1999 (mã chu kỳ là tùy chọn đối với đơn vị lịch cao nhất)
WM2 J6	thứ Bảy của tuần thứ hai trong tháng
M05 WM2 J6	thứ Bảy của tuần thứ hai trong tháng Năm
M05 DM08..14 J7	Ngày của Mẹ (Chủ nhật thứ hai trong tháng Năm)
J1..5 H0800..1600	thứ Hai tới thứ Sáu từ 8 giờ sáng tới 4 giờ chiều
J1..4 H0800..1600; J5 H0800..1200	thứ Hai tới thứ Năm từ 8 giờ sáng tới 4 giờ chiều và thứ Sáu từ 8 giờ sáng tới 12 giờ trưa.
[10 d] H/8	ba lần một ngày trong 10 ngày (mỗi lần một khoảng 60 phút).
H0800..1600 \J3	mỗi ngày từ 8 giờ sáng đến 4 giờ chiều, ngoại trừ thứ Tư.
(M0825..31 J1)..M0831	tuần lịch cuối cùng của tháng Tám.
JHNUSMEM..JHNUSLBR	mùa từ Ngày lễ tưởng niệm chiến sỹ vong trận tới Ngày lễ lao động của Mỹ

5.3.4.0 Các từ viết tắt cho biểu thức GTS.

Bảng 47 định nghĩa các từ viết tắt cho các giá trị *GTS* mà có thể được sử dụng ở dạng chữ thay vì thuật ngữ tương đương của chúng. Các từ viết tắt được định nghĩa cho các chu kỳ phổ biến trong ngày (sáng, chiều/tối), cho các chu kỳ trong tuần (ngày làm việc, cuối tuần), và cho các ngày lễ. Việc tính ngày của một số ngày lễ, ví dụ ngày lễ Phục sinh, khá phức tạp và vượt quá những gì sẽ được biểu diễn trong dạng chữ

GTS. Giả định rằng ngày của các ngày lễ này được rút ra từ một bảng nào đó hoặc một mô đun bộ tạo sóng nào đó nằm ngoài phạm vi của đặc tả này.

Các từ viết tắt này là các giá trị *GTS* được đặt tên và chúng có thể lần lượt là một yếu tố của dạng chữ *GTS*. Ví dụ, có thể nói "JHCHRXME H08..12" để chỉ giờ làm việc vào Đêm Giáng Sinh là từ 8 giờ sáng đến 1 giờ chiều. Và có thể nói "JHNUSMEM..JHNUSLBR" để chỉ mùa hồ bơi điển hình ở miền Trung Tây từ Ngày lễ tưởng niệm chiến sỹ vong trận tới Ngày lễ lao động.

Bảng 47. Miền Từ viết tắt GTS

mã	**định nghĩa**	**định nghĩa chính thức**
AM	Mỗi sáng vào thời gian do tổ chức quy định.	H00..11 IST
PM	Mỗi buổi chiều vào thời gian do tổ chức quy định.	H12..23 IST
BID	Hai lần một ngày vào thời gian do tổ chức quy định.	/(12 h) IST
TID	Ba lần một ngày vào thời gian do tổ chức quy định.	/(8 h) IST
QID	Bốn lần một ngày vào thời gian do tổ chức quy định.	/(6 h) IST
JB	Ngày làm việc thông thường (thứ Hai đến thứ Sáu ngoại trừ ngày lễ).	J1..5 \JH
JE	Cuối tuần thông thường (thứ Bảy và Chủ nhật ngoại trừ ngày lễ).	J6..7
JH	Ngày lễ	
GTSAbbreviationHolidays ChristianRoman		
JHCHRXME		M1224

mã	định nghĩa	định nghĩa chính thức
JHCHRXMS		M1225
JHCHRNEW		M0101
JHCHREAS		
JHCHRGFR		
JHCHRPEN		
JHNUS	Ngày lễ quốc gia của Hoa Kỳ (ngày lễ chung cho nhân viên liên bang được quy định bởi luật liên bang 5 U.S.C. 6103 của Hoa Kỳ).	
JHNUSMLK	Ngày tưởng niệm Tiến sỹ Martin Luther King, thứ Hai thứ ba của tháng Một.	M0115..21 J1
JHNUSPRE	Ngày kỷ niệm ngày sinh của Washington (ngày Tổng thống), thứ Hai thứ ba của tháng Hai.	M0215..21 J1
JHNUSMEM	Ngày tưởng niệm chiến sỹ vong trận, thứ Hai cuối cùng trong tháng Năm.	M0525..31 J1
JHNUSMEM5	Thứ Sáu trước cuối tuần ngày tưởng niệm chiến sỹ tử trận	M0522..28 J5
JHNUSMEM6	Thứ Bảy của cuối tuần ngày tưởng niệm chiến sỹ tử trận	M0523..29 J6
JHNUSIND	Ngày Quốc khánh (mùng 4 tháng Bảy)	M0704
JHNUSIND5	Thứ Sáu luân phiên trước cuối tuần ngày mùng 4 tháng Bảy [5 U.S.C. 6103(b)].	M0703 J5

mã	định nghĩa	định nghĩa chính thức
JHNUSIND1	Thứ Hai luân phiên sau cuối tuần ngày mùng 4 tháng 7 [5 U.S.C. 6103(b)].	M0705 J1
JHNUSLBR	Ngày lễ lao động, thứ Hai đầu tiên của tháng Chín	M0901..07 J1
JHNUSCLM	Ngày tưởng nhớ Columbus, thứ Hai tuần thứ hai trong tháng Mười.	M1008..14 J1
JHNUSVET	Ngày cựu chiến binh, ngày 11 tháng Mười một.	M1111
JHNUSTKS	Ngày lễ tạ ơn, thứ Năm tuần thứ tư trong tháng Mười một.	M1122..28 J4
JHNUSTKS5	Thứ Sáu sau ngày lễ tạ ơn.	M1123..29 J5

LƯU Ý: Bảng này chưa đầy đủ, và cũng không chứa các ngày lễ của các tôn giáo khác ngoài Thiên Chúa giáo (theo truyền thống Dương lịch [phương Tây]) hoặc các ngày lễ quốc gia của các nước khác ngoài Hoa Kỳ. Đây là hạn chế sẽ được khắc phục bằng các bổ sung sau này.

LƯU Ý: Các ngày lễ có tính đặc thù địa phương. Chính xác ngày lễ tôn giáo nào được xếp vào JH phụ thuộc vào mỗi địa phương và truyền thống văn hóa. Để đảm bảo mức độ liên kết vận hành toàn cầu, sử dụng các biểu thức *GTS* được xây dựng sẽ an toàn hơn các ngày lễ được đặt tên. Tuy nhiên, không thể dễ dàng biểu diễn những ngày lễ phụ thuộc vào chu kỳ mặt trăng (ví dụ, lễ Phục Sinh) hay quy định đặc biệt bằng dạng chữ *GTS*.

A. Các kiểu chứa thông tin

Các kiểu dữ liệu này hiện được đánh dấu là chứa thông tin trong khi các vấn đề được biết liên quan tới thiết kế của chúng đang được giải quyết.

A.1 Phân bổ xác suất theo tham số (PPD) chuyên biệt hóa T

Định nghĩa: Một phần mở rộng kiểu dữ liệu chung xác định tính không chắc chắn của dữ liệu định lượng nhờ dùng một hàm phân bổ và các tham số của nó. Ngoài các tham số cụ thể của phân bố, một giá trị trung bình (giá trị mong đợi) và một độ lệch chuẩn luôn được cung cấp để giúp duy trì một mức độ liên kết vận hành tối thiểu nếu các ứng dụng tiếp nhận không thể làm việc với một phân bổ xác suất nhất định.

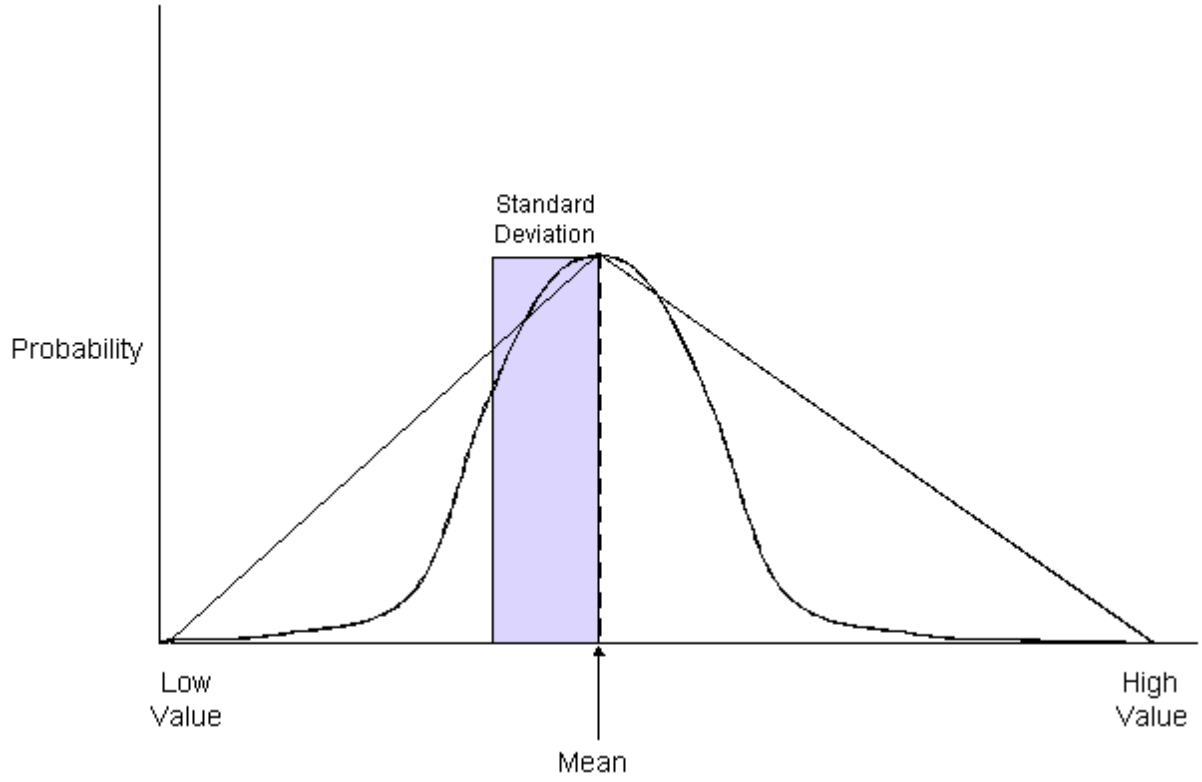
Bảng 48. Tóm tắt đặc tính của Phân bổ xác suất theo tham số

Tên	Kiểu	Mô tả
standardDeviation (Độ lệch chuẩn)	QTY	Thước đo chính cho sự biến thiên/tính không chắc chắn của giá trị (căn bậc hai của tổng các bình phương hiệu giữa tất cả các điểm dữ liệu và giá trị trung bình). *Độ lệch chuẩn* được dùng để chuẩn hóa các dữ liệu để tính hàm phân bố. Các ứng dụng không thể làm việc với các phân bố xác suất vẫn có thể nhận biết mức độ tin cậy khi nhìn vào *độ lệch chuẩn*.
distributionType (Kiểu phân bố)	CE	Mã xác định kiểu phân bố xác suất. Các giá trị có thể sẽ giống như được trình bày trong bảng kèm theo. Giá trị NULL (không biết) của mã kiểu chỉ báo rằng kiểu phân bố xác suất là không biết. Trong trường hợp đó, độ lệch chuẩn có ý nghĩa là một dự đoán không chính thức.

```
template<QTY T>
type ParametricProbabilityDistribution<T> alias PPD<T> specializes T {
  QTY     standardDeviation;
  CS      distributionType;
  IVL<T>  confidenceInterval(REAL p);
  REAL    probability(IVL<T> x);
  PPD<T>  times(REAL x);
};
```

Ví dụ, bài thi đầu vào đại học phổ biến nhất ở Mỹ là bài thi SAT với hai phần: đọc và toán. Mỗi phần có số điểm tối thiểu là 400 (không câu nào được trả lời đúng) và số điểm cao nhất là 800. Vào năm 1998, theo tổ chức College Board, 1.172.779 học sinh cấp 3 chuẩn bị vào đại học làm bài thi này. Điểm trung bình của phần thi toán của bài thi này là 512, và độ lệch chuẩn là 112. Các giá trị tham số này (512, 112), được đánh dấu là các tham số phân bố chuẩn, vẽ nên một bức tranh đầy đủ về phân bố điểm

thi. Trong hầu hết các trường hợp, không cần chỉ rõ tất cả hơn 1 triệu điểm dữ liệu khi chỉ cần 2 tham số là có thể thực hiện được điều này!



Ví dụ về phân bố xác suất theo tham số

Lưu ý rằng phân bố chuẩn chỉ là một trong số nhiều phân bố được định nghĩa cho tiêu chuẩn HL7.

Do kiểu dữ liệu *PPD* chuyên biệt hóa T, một giá trị T đơn giản là giá trị trung bình (giá trị được mong đợi hay mô men đầu tiên) của phân bố xác suất. Các ứng dụng không thể xử lý xác suất sẽ lấy giá trị T đơn giản và bỏ qua tính không chắc chắn. Giá trị đơn giản đó của kiểu T cũng được sử dụng để chuẩn hóa dữ liệu cho tính toán phân bố.

Phân bố xác suất được định nghĩa dựa trên số nguyên hoặc số thực và được chuẩn hóa đến một điểm tham chiếu nhất định (thường là 0) và một đơn vị tham chiếu nhất định (ví dụ, độ lệch chuẩn = 1). Khi các đại lượng khác mà được định nghĩa trong đặc tả này được sử dụng như các kiểu cơ bản, giá trị trung bình và độ lệch chuẩn được sử dụng để thể hiện theo tỷ lệ phân bố xác suất. Ví dụ, nếu một giá trị PPD<PQ> cho độ dài được cho với giá trị trung bình là 20ft và độ lệch chuẩn là 2in, hàm phân bố chuẩn hóa $f(x)$, liên hệ một số thực x với một mật độ xác suất, sẽ được chuyển ngữ thành hàm $f'(x')$, liên hệ một độ dài x' với một mật độ xác suất là $f'(x') = f((x' - \mu) / \sigma)$.

Khi có thể áp dụng, kiểu dữ liệu *PPD* phù hợp với tiêu chuẩn ISO *Hướng dẫn biểu diễn tính không chắc chắn trong đo lường* (GUM) như đã được thể hiện trong ấn phẩm NIST 1297 *Hướng dẫn đánh giá và biểu diễn tính không chắc chắn của các kết*

quả đo của NIST. Kiểu dữ liệu *PPD* không mô tả tính không chắc chắn sẽ được đánh giá như thế nào mà chỉ nó được thể hiện như thế nào. Khái niệm "tính không chắc chắn tiêu chuẩn" do ISO GUM đề ra tương ứng với độ lệch chuẩn.

A.1.1. Độ lệch chuẩn : QTY

Định nghĩa: Thước đo chính cho sự biến thiên/tính không chắc chắn của giá trị (căn bậc hai của tổng các bình phương hiệu giữa tất cả các điểm dữ liệu và giá trị trung bình). *Độ lệch chuẩn* được dùng để chuẩn hóa các dữ liệu để tính hàm phân bố. Các ứng dụng không thể làm việc với các phân bố xác suất vẫn có thể nhận biết mức độ tin cậy khi nhìn vào *độ lệch chuẩn*.

độ lệch chuẩn là một chuyên biệt hóa của kiểu dữ liệu QTY (từ T.diffType - T.Kiểu hiệu), thể hiện mức chênh lệch giữa các giá trị của kiểu dữ liệu T. Nếu kiểu dữ liệu T là REAL hay INT, thì T.diffType cũng lần lượt là REAL hay INT. Tuy nhiên, nếu kiểu dữ liệu T là TS, thì T.diffType là PQ theo đơn vị thời gian.

```
invariant(PPD x) {
  x.standardDeviation.dataType.implies(T.diffType);
};
```

A.1.2. Kiểu phân bố xác suất : CE

Định nghĩa: Mã xác định kiểu phân bố xác suất. Các giá trị có thể sẽ như được trình bày trong bảng kèm theo. Giá trị NULL (không biết) của mã kiểu chỉ báo rằng kiểu phân bố xác suất là không biết. Trong trường hợp đó, độ lệch chuẩn có ý nghĩa của một dự đoán không chính thức.

Bảng 49 liệt kê các phân bố xác suất được định nghĩa. Nhiều kiểu phân bố được định nghĩa theo các tham số đặc biệt (ví dụ, các tham số α và β cho phân bố γ, số bậc tự do cho phân bố *t*, vv.). Tuy nhiên, giá trị trung bình và độ lệch chuẩn được định nghĩa cho tất cả các kiểu phân bố.

Bảng 49. Miền ProbabilityDistributionType

mã	tên	định nghĩa
(NULL)	(NULL)	Được dùng để chỉ ra rằng giá trị trung bình được ước tính mà không xem xét kỹ hơn về phân bố xác suất của nó. Trong trường hợp này, ý nghĩa của độ lệch chuẩn không được định nghĩa rõ ràng. Tuy nhiên, việc diễn dịch nên dựa theo phân bố chuẩn, ví dụ, khoảng được bao phủ bởi giá trị trung bình ±1 độ lệch chuẩn nên có mức độ tin cậy khoảng 2/3.
U	uniform	Phân bố đều gán một xác suất không đổi cho toàn bộ khoảng các

mã	tên	định nghĩa
	(phân bố đều)	kết quả có thể xảy ra, trong khi mọi kết quả bên ngoài khoảng này được giả định là có xác suất bằng 0. Độ rộng của khoảng này là $2\sigma\sqrt{3}$. Do đó, phân bố đều gán mật độ xác suất $f(x) = (2\sigma\sqrt{3})^{-1}$ cho các giá trị $\mu - \sigma\sqrt{3} \geq x \leq \mu + \sigma\sqrt{3}$ và $f(x) = 0$ cho các trường hợp còn lại.
N	normal (Gaussian) phân bố chuẩn (Gaussian)	Đây là phân bố chuẩn hình chuông nổi tiếng. Do có định lý giới hạn trung tâm, phân bố chuẩn là phân bố được lựa chọn cho một biến ngẫu nhiên không giới hạn vốn là kết quả của sự kết hợp rất nhiều quá trình ngẫu nhiên. Ngay cả với các giá trị bị giới hạn ở một phía (tức là lớn hơn 0), phân bố chuẩn cũng có thể đủ chính xác nếu giá trị trung bình "cách xa" giới hạn của thang đo được đo theo độ lệch chuẩn.
LN	log-normal phân bố logarit chuẩn	Phân bố logarit chuẩn được dùng để chuyển một biến ngẫu nhiên lệch X thành một biến ngẫu nhiên phân bố chuẩn U = log X. Phân bố logarit chuẩn có thể được đặc tả với các đặc tính là giá trị trung bình μ và độ lệch chuẩn σ. Tuy nhiên, cần lưu ý rằng giá trị trung bình μ và độ lệch chuẩn σ là các tham số của phân bố giá trị thô, chứ không phải là các tham số đã được chuyển đổi của phân bố logarit chuẩn mà thường vẫn được gọi bằng các chữ cái đó. Các tham số của phân bố logarit chuẩn μ_{log} và σ_{log} quan hệ với giá trị trung bình μ và độ lệch chuẩn σ của giá trị dữ liệu qua các biểu thức $\sigma_{log}^2 = \log(\sigma^2/\mu^2 + 1)$ và $\mu\log = \log\mu - \sigma_{log}^2/2$.
G	γ (gamma) phân bố γ (gamma)	Phân bố gamma được dùng cho các dữ liệu bị lệch và giới hạn phía bên phải, tức là khi max của đường phân bố nằm cạnh gốc. Phân bố γ có hai tham số là α và β. Quan hệ với giá trị trung bình μ và phương sai σ^2 là $\mu = \alpha\beta$ và $\sigma^2 = \alpha\beta^2$
E	exponential phân bố mũ	Được dùng cho các dữ liệu mô tả sự suy giảm dần. Phân bố mũ là dạng đặc biệt của phân bố γ trong đó $\alpha = 1$, do đó, quan hệ với giá trị trung bình μ và phương sai σ^2 là $\mu = \beta$ và $\sigma^2 = \beta^2$.
X2	χ phân bố χ	Được dùng để mô tả tổng bình phương của các biến ngẫu nhiên xảy ra khi một phương sai được ước lượng (thay vì được chấp nhận là đúng) từ mẫu. Tham số duy nhất của phân bố χ^2 là υ, còn được gọi là *số bậc tự do* (là số các thành phần tự do trong tổng). Phân bố χ^2 là một kiểu đặc biệt của phân bố γ với tham số $\alpha = \upsilon/2$ và $\beta = 2$. Do

mã	tên	định nghĩa
		đó, $\mu = \upsilon$ và $\sigma^2 = 2\,\upsilon$.
T	t (Student) phân bố t (Sinh viên)	Được dùng để mô tả thương số giữa một biến ngẫu nhiên phân bố chuẩn và căn bậc hai của một biến ngẫu nhiên phân bố χ^2. Phân bố *t* có một tham số là υ, số bậc tự do. Quan hệ với giá trị trung bình μ và phương sai σ^2 là: $\mu = 0$ và $\sigma^2 = \upsilon / (\upsilon - 2)$.
F	F phân bố F	Được dùng để mô tả thương số giữa hai biến ngẫu nhiên phân bố χ^2. Phân bố *F* có hai tham số là υ_1 và υ_2, lần lượt là số bậc tự do của biến tử số và biến mẫu số. Quan hệ với giá trị trung bình μ và phương sai σ^2 là: $\mu = \upsilon_2 / (\upsilon_2 - 2)$ và $\sigma^2 = (2\,\upsilon_2\,(\upsilon_2 + \upsilon_1 - 2)) / (\upsilon_1\,(\upsilon_2 - 2)^2\,(\upsilon_2 - 4))$.
B	β (beta) phân bố β (beta)	Phân bố beta được dùng cho các dữ liệu bị giới hạn ở cả hai phía và có thể bị lệch hoặc không (ví dụ, xảy ra khi ước lượng xác suất). Hai tham số α và β được cung cấp để điều chỉnh đường phân bố. Giá trị trung bình μ và phương sai σ^2 quan hệ với tham số như sau: $\mu = \alpha / (\alpha + \beta)$ và $(\sigma^2 = \alpha\,\beta/((\alpha + \beta)^2\,(\alpha + \beta + 1))$.

Ba kiểu phân bố *chưa biết* (NULL)*, đều và chuẩn* phải được hỗ trợ bởi tất cả các hệ thống có hỗ trợ kiểu dữ liệu *PPD*. Toàn bộ các kiểu phân bố khác là tùy chọn. Khi một hệ thống đang diễn dịch một biểu diễn *PPD* gặp một kiểu phân bố mà nó không nhận ra, nó sẽ khớp kiểu này với kiểu phân bố chưa biết (NULL).

A.1.3. Dạng chữ

Cú pháp tổng quát của dạng chữ cho kiểu dữ liệu *PPD* là như sau:

```
PPD<T>.literal ST {
  PPD<T>  : T "(" type QTY ")"  { ((T)$).equal($1);
                                  $.distributionType.equal($3);
                                  $.standardDeviation.equal($4); };
  CV type : ST                  { $.value.equal($1);
                                  $.codeSystem.equal(2.16.840.1.113883.5.1020); };
};
```

Ví dụ: một ví dụ cho PPD<REAL> là "1.23(N0.005)" cho một Kiểu phân bố chuẩn quanh 1.23 với độ lệch chuẩn là 0.005. Một ví dụ cho PPD<PQ> là "1.23 m (5 mm)" cho một Kiểu phân bố chưa biết quanh độ dài 1.23 mét với độ lệch chuẩn là 5 milimét. Một ví dụ cho giá trị PPD<TS> là "2000041113(U4 h)" cho một Kiểu phân

bố đều quanh ngày 11 tháng Tư năm 2000 vào lúc 1 giờ chiều với độ lệch chuẩn là 4 tiếng.

A.2. Phân bố xác suất trên số thực (PPD<REAL>) chuyên biệt hóa PPD

```
type ParametricProbabilityDistribution<REAL> alias PPD<REAL>;
```

Phân bố xác suất theo tham số của số thực được định nghĩa đầy đủ bởi kiểu dữ liệu chung. Tuy nhiên, có một số cân nhắc đặc biệt về biểu diễn dạng chữ và chuyển đổi của các phân bố xác suất trên các kiểu dữ liệu REAL, sẽ được chỉ rõ trong mục này.

A.2.1. Chuyển đổi một số thực (REAL) thành một số thực không chắc chắn (PPD<REAL>)

Khi chuyển đổi một giá trị REAL thành một giá trị dạng PPD<REAL>, PPD.standardDeviation được tính từ thứ tự về độ lớn và REAL.precision (số chữ số quan trọng) của giá trị REAL. Nếu lấy x là một giá trị REAL với REAL.precision là n, chúng ta có thể xác định thứ tự độ lớn e của x là $e = \log10\ |x|$ trong đó e được làm tròn lên số nguyên tiếp sau gần với 0 hơn (trường hợp đặc biệt: nếu x là 0, e là 0). Giá trị của chữ số ít quan trọng nhất l khi đó là $l = 10^{e-n}$ và PPD.standardDeviation $\sigma = l / 2$.

Bảng 50. Ví dụ về độ lệch chuẩn được tính từ độ chính xác p và thứ tự độ lớn e

Biểu diễn	x	e	p	e - p + 1	l	σ
0	0	(0)	1	0	1	0.5
1	1	0	1	0	1	0.5
2	2	0	1	0	1	0.5
9	9	0	1	0	1	0.5
10	10	1	2	0	1	0.5
100	100	2	3	0	1	0.5
1e+1	10	1	1	1	10	5
1e+2	100	2	1	2	100	50

Biểu diễn	x	e	p	e - p + 1	l	σ
10e+1	100	2	2	1	10	5
1.1	1.1	0	2	-1	0.1	0.05
10.1	10.1	1	3	-1	0.1	0.05
1.1e+2	110	2	2	1	10	5
1.1e-2	0.011	-2	2	-3	0.001	0.0005
1.1e-4	0.00011	-4	2	-5	0.00001	0.000005
10.1e-4	0.00101	-3	3	-5	0.00001	0.000005
0.1e-1	0.01	-2	1	-2	0.01	0.005
0.01e-1	0.001	-3	1	-3	0.001	0.0005
0.01e-2	0.0001	-4	1	-4	0.0001	0.00005
0.00	0	(0)	3	-2	0.01	0.005

A.2.2. Dạng chữ ngắn gọn

Ngoài dạng chữ chung của kiểu dữ liệu *PPD<REAL>*, có một dạng chữ ngắn gọn được định nghĩa cho kiểu dữ liệu *PPD<REAL>* trên số thực. Dạng chữ ngắn gọn này được định nghĩa sao cho có thể biểu diễn PPD.standardDeviation theo chữ số ít quan trọng nhất trong phần định trị. Dạng chữ này được định nghĩa là phần mở rộng của dạng chữ REAL:

```
PPD<REAL>.literal ST {
  PPD<REAL> mantissa
   : REAL.mantissa "(" type QTY ")" { ((T)$).equal($1);
                                      $.distributionType.equal($3);
                                      $.standardDeviation.equal($4); }
   | REAL.mantissa      { $.equal($1);
```

```
                    $.distributionType.equal($3);
                    $.standardDeviation.equal(
                    $1.leastSignificantDigit.times(0.5)); };
  CS type : ST      { $.value.equal($1);
                  $.system.equal(2.16.840.1.113883.5.1019); };
};
```

Ví dụ: "1.23e-3 (U5e-6)" là PPD.distributionType đều quanh 1.23×10^{-3} với PPD.standardDeviation 5×10^{-6} ở dạng chữ chung. "1.230(U5)e-3" cùng là giá trị này ở dạng chữ ngắn gọn.

A.3. Phân bố xác suất theo tham số trên đại lượng vật lý (PPD<PQ>) chuyên biệt hóa PPD

Kiểu dữ liệu *PPD<PQ>* được xây dựng từ kiểu dữ liệu PPD. Tuy nhiên, ý thức được rằng PQ.unit có thể được tính từ các ranh giới, chúng tôi thêm vào ngữ nghĩa bổ sung và một dạng chữ riêng biệt. Quan điểm bổ sung về một *PPD<PQ>* là một phân bố xác suất trên các số thực với một đơn vị.

```
type ParametricProbabilityDistribution<PQ> alias PPD<PQ> {
  PPD<REAL> value;
  CS  unit;
};
```

Đơn vị áp dụng cho cả giá trị trung bình và PPD.standardDeviation

```
invariant(PPD<PQ> x)
    where x.nonNull {
  x.value.nonNull;
  ((REAL)x.value).equal(((PQ)x).value);
  x.unit.equal(((PQ)x).unit);
  x.value.standardDeviation.equal(x.standardDeviation.value);
  x.standardDeviation.unit.equal(x.unit);
};
```

A.3.1. Dạng chữ ngắn gọn

Một dạng chữ ngắn gọn cho kiểu dữ liệu *PPD<PQ>* được định nghĩa dựa trên dạng chữ ngắn gọn của kiểu dữ liệu PPD<REAL> trong đó REAL là giá trị. Dạng chữ này được định nghĩa là phần mở rộng của dạng chữ PQ.

```
PPD<PQ>.literal ST {
  PPD<PQ> : PPD<REAL> " " unit  { $.value.equal($1);
                    $.unit.equal($3); }
```

```
};
```

Ví dụ: "1.23e-3 m (N5e-6 m)" là độ dài được phân bố chuẩn 1.23×10^{-3} m với PPD.standardDeviation 5×10^{-6} m ở dạng chữ chung. "1.230(N5)e-3 m" cùng là giá trị này ở dạng chữ ngắn gọn. "1.23e-3(N0.005e-3) m " cũng hợp lệ; nó là dạng chữ ngắn gọn cho kiểu dữ liệu *PPD<PQ>* được kết hợp với dạng chữ chung cho PPD<REAL>

A.4. Phân bố xác suất trên thời điểm (PPD<TS>) chuyên biệt hóa PPD

```
type ParametricProbabilityDistribution<TS> alias PPD<TS>;
```

Kiểu dữ liệu *PPD<TS>* được định nghĩa đầy đủ bởi kiểu dữ liệu chung. PPD.stanrdDeviation là thuộc kiểu dữ liệu TS.diffType, là một khoảng thời gian (một giá trị PQ theo đơn vị thời gian).

A.4.1. Chuyển đổi TS thành *PPD<TS>*

Khi chuyển đổi một giá trị TS thành một giá trị *PPD<TS>*, PPD.standardDeviation được tính từ thứ tự độ lớn và độ chính xác (số chữ số quan trọng) của giá trị TS sao cho hai giá trị PPD.standardDeviation bao trùm dải thời gian lớn nhất của các chữ số không được xác định. Ví dụ, trong 20000609 các chữ số không được xác định là giờ trong ngày và thấp hơn. Tất cả các chữ số này cùng bao trùm một khoảng thời gian là 24 tiếng, và do đó, PPD.standardDeviation là (= 12 tiếng từ 20000609000000.0000... cho tới 20000609999999.9999... (= 20000610)

Quy tắc này khác với quy tắc được xác định cho REAL ở chỗ phạm vi của tính không chắc chắn nằm trên giá trị thời gian đã được chỉ rõ. Điều này đi kèm với nhận định phổ biến rằng ngày 9 tháng Sáu bao trùm toàn bộ ngày 9 tháng Sáu với giữa trưa là trung tâm, không phải nửa đêm.

Chú giải cuối cùng

1. [nguồn] Khung Phát Triển Bản Tin HL7 định nghĩa các "chế độ cập nhật" cho các trường dữ liệu trong một bản tin. Lưu ý rằng bởi giá trị dữ liệu không có định danh hay trạng thái hay thay đổi trạng thái, các chế độ cập nhật này không áp dụng cho các đặc tính của giá trị dữ liệu. Các giá trị dữ liệu và các đặc tính của chúng không bao giờ được cập nhật. Một trường dữ liệu của một đối tượng (ví dụ, một bản tin) có thể được cập nhật trong trường hợp giá trị của trường đó được thay thế bởi một giá trị khác. Nhưng bản thân giá trị không bao giờ được cập nhật.
2. [nguồn] Đây là lí do vì sao tiêu chuẩn ISO Ký hiệu Cú pháp Trừu tượng 1 (ASN.1) không phải là hình thức phù hợp cho các đặc tả kiểu dữ liệu ngữ nghĩa.
3. [nguồn] Ngôn ngữ định nghĩa kiểu dữ liệu được sử dụng ở đây là một kết luận của nhiều thí nghiệm và kinh nghiệm với nhiều lựa chọn thay thế khác nhau. Các lựa chọn thay thế này bao gồm các bảng định nghĩa kiểu dữ liệu và việc sử

dụng Ngôn ngữ Định nghĩa Giao diện (IDL) của Nhóm Quản lý Đối tượng (OMG). Điểm bất lợi của các bảng định nghĩa kiểu dữ liệu là chúng tạo một ấn tượng sai lầm về đặc tả này là đặc tả của cú pháp trừu tượng chứ không phải của ngữ nghĩa. Ngược lại, điểm bất lợi của IDL là IDL tạo một ấn tượng sai lầm về đặc tả này là một định nghĩa giao diện lập trình ứng dụng (API).

Ngôn ngữ định nghĩa kiểu dữ liệu cuối cùng vay mượn đáng kể từ IDL, Ngôn ngữ Ràng buộc Đối tượng (OCL), JAVA, C++, và các công cụ tạo bộ phân tích cú pháp LEX và YACC. Nó lấy cảm hứng từ các đặc điểm và phong cách của các ngôn ngữ này nhưng pha trộn và bổ sung các ngôn ngữ này thành chính xác thứ cần cho đặc tả kiểu dữ liệu này. Mục tiêu là một ngôn ngữ tối giản và nhất quán. Ngoài ra, do mục tiêu chính của ngôn ngữ này là để định nghĩa các kiểu dữ liệu, nó cố gắng làm được điều này mà không cần bất kỳ kiểu dữ liệu tạo sẵn nào.

4. [nguồn] Như có thể thấy, từ khóa kiểu thay thế cho từ khóa giao diện của IDL và Java và từ khóa lớp của C++ và Java. Mệnh đề bí danh là duy nhất của đặc tả này do chúng ta rất cần các kiểu dữ liệu cực ngắn dễ nhớ bên cạnh những tên gọi có tính mô tả nhiều hơn. Mệnh đề chuyên biệt hóa được ưa chuộng hơn so với mệnh đề hai chấm của C++ hay IDL do ý nghĩa của nó rõ ràng hơn.
5. [nguồn] Lưu ý rằng khái niệm đối số đầu vào và đầu ra của IDL và khái niệm trả giá trị và ngoại lệ của IDL, JAVA và C++ đều là những khái niệm không phù hợp cho đặc tả này. Ngữ nghĩa của các kiểu dữ liệu không phải là về gọi thủ tục và truyền tham số hay trả điều khiển bình thường hay bất thường từ một nhóm thủ tục. Thay vào đó, mỗi đặc tính ngữ nghĩa được khái niệm hóa thành một chức năng liên kết một giá trị và các đối số tùy chọn với một giá trị khác. Việc liên kết này không được "tính toán" hay "tạo ra", nó tồn tại một cách logic và chúng ta không cần "gọi" một chức năng như vậy để hiện thực hóa việc liên kết.
6. [nguồn] Khía cạnh thu hẹp của chuyên biệt hóa xứng đáng được giải thích. Người ta thường cho rằng thừa kế không nên rút lại các đặc tính đã được định nghĩa cho chi. Điều này vẫn đúng đối với việc thu hẹp do các đặc tính không thực sự bị rút lại mà được ràng buộc với một tập hợp giá trị nhỏ hơn. Điều này có thể đồng nghĩa với việc ràng buộc các đặc tính với NULL, nếu NULL là một giá trị được chấp nhận cho đặc tính đó trong kiểu cha. Dù trong bất kỳ trường hợp nào, về mặt logic, thu hẹp là một chuyên biệt hóa, với tính thừa kế và khả năng thay thế.
7. [nguồn] Lưu ý rằng ý nghĩa của được bảo vệ hơi khác so với từ hạn định khả năng tiếp cận (công cộng, gói, được bảo vệ, cá nhân) như đã biết trong JAVA và C++. Bảo vệ được sử dụng ở đây không phải với nghĩa che giấu thông tin kiểu hay ngăn chặn các đặc tính được định nghĩa bởi một kiểu được bảo vệ khỏi tiếp cận bên ngoài "gói" đặc tả này. Nó chủ yếu là một khuyến cáo mạnh mẽ

không khai báo các thuộc tính hay đặc điểm khác của những kiểu được bảo vệ này. Các kiểu được bảo vệ nên được sử dụng như "được bọc" trong các kiểu khác. Kiểu được bảo vệ vẫn có thể được tiếp cận trực tiếp bên trong "bọc," không tồn tại khái niệm "đặc tính được ủy thác".

8. [nguồn] Cú pháp và ngữ nghĩa câu lệnh bất biến tương tự với mệnh đề "inv" của OCL. Tuy nhiên, chúng tôi không sử dụng OCL trong đặc tả này vì một số lí do. (1) Cú pháp OCL có phong cách Chuyện phiếm không phù hợp với phong cách của C++/Java của ngôn ngữ định nghĩa kiểu dữ liệu. (2) OCL có rất nhiều kiểu dữ liệu và cấu trúc nguyên thủy, trong khi đặc tả này tránh tối đa nguyên thủy. (3) Do có nhiều cấu trúc nguyên thủy, OCL khá phức tạp, hơn mức cần thiết cho đặc tả này.
9. [nguồn] Cấu trúc này phần nào có tính chu kỳ; có một khái niệm đã có từ trước của giá trị Boolean mặc dù Boolean là một kiểu được định nghĩa giống như bất kỳ kiểu nào khác. Thêm vào đó, do ngôn ngữ định nghĩa kiểu dữ liệu này được viết bằng các chuỗi ký tự, khái niệm chuỗi ký tự có trước định nghĩa về kiểu chuỗi ký tự. Hai kiểu này, chuỗi ký tự và Boolean do đó là ngoại lệ, nhưng trên bề mặt, chúng được định nghĩa giống như bất kỳ kiểu dữ liệu nào khác. Do ngôn ngữ đặc tả kiểu dữ liệu này không phải là để triển khai, tính chu kỳ không thực sự là vấn đề. Ngay cả nếu ngôn ngữ này được triển khai, có thể sử dụng kỹ thuật "bootstrapping" (khởi động) như khá phổ biến, ví dụ, đối với các trình biên dịch mà biên dịch chính chúng.
10. [nguồn] Hầu hết các đặc điểm cú pháp này là theo tinh thần của ngôn ngữ JAVA, sử dụng các danh sách đối số, ngoặc cong để bao quanh các khối, chấm phẩy để kết thúc câu lệnh, và chấm để tham chiếu đặc tính giá trị. Dấu hai chấm kép :: như được sử dụng trong C++ hay IDL để phân biệt tham chiếu thành viên và tham chiếu giá trị không được sử dụng (như trong Java). Khác với Java nhưng giống như C++ và IDL, mọi câu lệnh được kết thúc bởi một dấu chấm phẩy, bao gồm cả các khai báo kiểu. Các chuyển đổi kiểu ẩn cũng được giữ lại từ C++.
11. [nguồn] Điều này có nghĩa là nếu một người mong đợi một giá trị ED nhưng thực tế lại có một giá trị ST, người đó có thể chuyển giá trị ST thành ED.
12. [nguồn] Ngữ pháp khác nhau của các dạng chữ không phải là để kết hợp thành một ngữ pháp biểu thức giá trị HL7 tổng quát. Mặc dù đã có những nỗ lực được thực hiện để loại trừ những mập mờ có thể xảy ra giữa các dạng chữ của các kiểu khác nhau tại nơi chúng có thể gây hại, vẫn còn những mập mờ chưa thể giải quyết. Ví dụ "1.2" có thể là dạng chữ hợp lệ cho cả Định danh Đối tượng (OID) và một Số thực.
13. [nguồn] Biến thể BNF được sử dụng ở đây tương tự với các ngôn ngữ bộ tạo sóng bộ phân tích từ vựng LEX và bộ phân tích cú pháp YACC nhưng được đơn giản hóa và làm cho nhất quán với cú pháp và phong cách khai báo của

ngôn ngữ định nghĩa kiểu dữ liệu này. Sự khác biệt nằm ở chỗ tất cả các ký hiệu có chính xác một thuộc tính, giá trị của chúng được định kiểu như một trong số các kiểu dữ liệu được định nghĩa. Kiểu của mỗi ký hiệu được khai báo trước định nghĩa của ký hiệu (ví dụ: chữ số INT : "0" | "1" | ... | "9";). Ký hiệu bắt đầu không có tên mà chỉ có một kiểu (ví dụ, INT : chữ số | chữ số INT;). Một tên kiểu dữ liệu có thể xuất hiện như một tên ký hiệu với ý nghĩa một dạng chữ của kiểu dữ liệu đó.

14. [nguồn] Lưu ý rằng đặc tính bình đẳng (được định nghĩa cho tất cả các kiểu dữ liệu, xem bình đẳng) là một quan hệ, một kiểm tra tính bình đẳng, chứ không phải một câu lệnh gán. Không thể gán một giá trị cho một giá trị khác. Không giống các bộ phân tích YACC và LEX, ngôn ngữ định nghĩa kiểu dữ liệu này thuần túy có tính khai báo, nó không có khái niệm gán. Vì vậy, các quy tắc ngữ pháp định nghĩa cả các biểu thức dạng chữ phân tích và xây dựng.
15. [nguồn] Phần mở rộng kiểu chung đôi khi được gọi là các "mixin" (trộn), do tác dụng của chúng là để trộn một số đặc tính nhất định vào các kiểu dữ liệu đã có.
16. [nguồn] Thuật toán của tổng kiểm tra đủ mạnh về mặt mã hóa Thuật toán Băm An toàn-1 (SHA-1) hiện là tiêu chuẩn trong ngành. Nó mới chỉ thay thế thuật toán MD5 vài năm gần đây, khi người ta phát hiện một số lỗi về bảo mật của MD5. Hiện nay, thuật toán băm SHA-1 là lựa chọn mặc định cho thuật toán kiểm tra tính toàn vẹn. Lưu ý rằng SHA-256 cũng đang bắt đầu được sử dụng rộng rãi.
17. [nguồn] RFC 3066 [http://www.ietf.org/rfc/rfc3066.txt] là hệ thống mã hóa được HL7 thông qua cho tất cả các tham chiếu tới các ngôn ngữ của con người, trong các kiểu dữ liệu và những nơi khác.
18. [nguồn] Vì lí do này, một hệ thống hay trang không xử lý các tên hay văn bản đa ngôn ngữ trong thế giới thực có thể an toàn bỏ qua đặc tính ngôn ngữ.
19. [nguồn] Thuật toán của tổng kiểm tra đủ mạnh về mặt mã hóa Thuật toán Băm An toàn-1 (SHA-1) hiện là tiêu chuẩn trong ngành. Nó mới chỉ thay thế thuật toán MD5 vài năm gần đây, khi người ta phát hiện một số lỗi về bảo mật của MD5. Hiện nay, thuật toán băm SHA-1 là lựa chọn mặc định cho thuật toán kiểm tra tính toàn vẹn. Lưu ý rằng SHA-256 cũng đang bắt đầu được sử dụng rộng rãi.
20. [nguồn] ISO/IEC 10646-1: 1993 định nghĩa một ký tự là "một thành viên của một tập hợp các phần tử được dùng cho việc tổ chức, điều khiển, hoặc biểu diễn dữ liệu." ISO/IEC TR 15285 - Mô hình hoạt động cho ký tự và nét chạm, thảo luận các vấn đề liên quan đến việc định nghĩa ký tự. Đáng lưu ý, ký tự là các thực thể thông tin trừu tượng, độc lập với phông chữ hay ngôn ngữ. Tiêu chuẩn ISO 10646 (UNICODE [http://www.unicode.org]) - hay ở Nhật Bản, JIS X0221 - là một tập hợp ký tự có thể ứng dụng trên toàn cầu xác định duy nhất tất cả các ký tự của bất kỳ ngôn ngữ nào trên thế giới.

Trong đặc tả này, tiêu chuẩn ISO 10646 là mô hình ngữ nghĩa cho các chuỗi ký tự. Điểm quan trọng là để phục vụ mục đích ngữ nghĩa, không có khái niệm tập hợp ký tự riêng biệt và chuyển đổi giữa các tập hợp ký tự. Tập hợp ký tự và mã hóa ký tự là các cân nhắc lớp của ITS. Định nghĩa chính thức biểu thị tác động này bởi vì mỗi ký tự tự thân nó là một giá trị *ST* có một đặc tính tập hợp ký tự. Vì vậy, mã hóa nhị phân mỗi ký tự luôn được hiểu trong bối cảnh của một tập hợp ký tự nhất định. Điều này không có nghĩa rằng ITS nên biểu diễn một chuỗi ký tự như một chuỗi các giá trị ED hoàn thiện. Ý nghĩa ở đây là trên lớp ứng dụng, khái niệm mã hóa ký tự là không phù hợp khi chúng ta làm việc với chuỗi ký tự.

21. [nguồn] RFC 3066 [http://www.ietf.org/rfc/rfc3066.txt] là hệ thống mã hóa được HL7 thông qua cho tất cả các tham chiếu tới các ngôn ngữ của con người, trong các kiểu dữ liệu và những nơi khác.
22. [nguồn] Vì lí do này, một hệ thống hay trang không xử lý các tên hay văn bản đa ngôn ngữ trong thế giới thực có thể an toàn bỏ qua đặc tính ngôn ngữ.
23. [nguồn] Một dạng chữ *ST* là một chuyển đổi từ một chuỗi ký tự sang một kiểu dữ liệu khác. Hiển nhiên, dạng chữ *ST* cho các chuỗi ký tự là một đặc điểm có tính chu kỳ nếu không phải là thừa thãi. Vì vậy, dạng chữ này chủ yếu chỉ ra các chuỗi ký tự được phân tích như thế nào trong ngôn ngữ đặc tả kiểu dữ liệu.
24. [nguồn] Ưu điểm của kiểu dữ liệu *CD* là tính biểu diễn của nó, tuy nhiên, nếu tất cả các đặc điểm của nó, như ngoại lệ mã hóa, văn bản, chuyển ngữ và từ hạn định, luôn được sử dụng, triển khai và sử dụng sẽ trở nên rất khó khăn và không an toàn. Vì vậy, kiểu dữ liệu *CD* thường được sử dụng nhất ở dạng thu hẹp với các đặc điểm được rút gọn.
25. [nguồn] Phiên bản hệ thống mã không quan trọng trong kiểm tra tính bình đẳng do xét về định nghĩa một ký hiệu mã phải có cùng một ý nghĩa trong suốt tất cả các phiên bản của một hệ thống mã. Giữa các phiên bản, mã có thể được cho hết hiệu lực nhưng không được rút lại hay tái sử dụng.
26. [nguồn] Các chuyển ngữ không được đưa vào kiểm tra tính bình đẳng của bộ mô tả khái niệm vì lí do an toàn. Một lựa chọn thay thế có thể là coi hai giá trị *CD* là bình đẳng nếu bất kỳ chuyển ngữ nào của chúng là bình đẳng. Tuy nhiên, một vài chuyển ngữ có thể bình đẳng do hệ thống mã hóa của chuyển ngữ đó chưa tinh tế. Những so sánh phức tạp hơn giữa các bộ mô tả khái niệm là những cân nhắc về ứng dụng không được đề cập tới trong đặc tả này.
27. [nguồn] Các giá trị NULL là các giá trị ngoại lệ, không phải khái niệm hợp lệ. Sẽ là không an toàn khi cho hai giá trị là bình đẳng chỉ đơn thuần dựa trên cơ sở cả hai là ngoại lệ (ví dụ, không mã hóa được hay không biết). Tương tự, không có đảm bảo nào cho việc văn bản gốc biểu diễn một mô tả có nghĩa hay duy nhất cho khái niệm sao cho tính bình đẳng của văn bản gốc đó không tạo thành bình đẳng khái niệm. Điều ngược lại cũng đúng: do có nhiều hơn một văn bản

gốc có thể cho một khái niệm, thực tế rằng văn bản gốc khác nhau không tạo thành sự khác biệt về khái niệm.

28. [nguồn] Quyết định này tại thời gian thiết kế là cần thiết để ngăn không cho các giao diện HL7 phải chịu gánh nặng từ các chuyển đổi phong cách dạng chữ mã trong thời gian chạy. Điều này bất chấp thực tế là một số ứng dụng có thể đòi hỏi liên kết từ một dạng sang một dạng khác nếu ứng dụng đó đã chốt một tùy chọn biểu diễn không được HL7 lựa chọn.
29. [nguồn] Đây là một lí do giải thích vì sao từ hạn định nên được sử dụng một cách tiết kiệm và thận trọng cho hậu kết hợp. Một vấn đề nữa của mã hóa hậu kết hợp là có thể hoàn toàn không tồn tại một quy tắc tổng quát cho tính bình đẳng.
30. [nguồn] Điều này bất chấp thực tế là một miền được tham chiếu bên ngoài, như kiểu phương tiện IETF MIME, có thể chứa một cơ chế mở rộng. Các mã kiểu MIME được mở rộng này sẽ không được coi là "phần mở rộng" theo nghĩa đã vi phạm quy định của kiểu dữ liệu CNE. Quy định của kiểu dữ liệu CNE chỉ bị vi phạm khi có một nỗ lực trong việc sử dụng một hệ thống mã khác (bằng phương tiện của CD.codeSystem), vốn là không thể với kiểu dữ liệu *CS*.
31. [nguồn] Quan điểm giá trị/không gian tên trên các định danh đối tượng ISO có sự liên quan ngữ nghĩa quan trọng. Nó biểu diễn khái niệm giá trị định danh so với hệ thống cấp định danh (= không gian tên), vốn phổ biến trong các hệ thống thông tin y tế nói chung, và HL7 v2.x nói riêng.
32. [nguồn] Các đối tượng DICOM được xác định chỉ bởi UID. Để phục vụ mục đích tích hợp DICOM/HL7, sẽ là bất tiện nếu HL7 yêu cầu phần mở rộng phải là bắt buộc và coi UID chỉ như một hệ thống cấp định danh. Do các giá trị UID đơn giản hơn và không có nguy cơ chứa những trang trí vô nghĩa, chúng tôi khuyến khích các hệ thống sử dụng các định danh UID đơn giản như các tham chiếu ngoài cho đối tượng của chúng.
33. [nguồn] Quyết định này tại thời gian thiết kế là cần thiết để ngăn không cho các giao diện HL7 phải chịu gánh nặng từ các chuyển đổi phong cách dạng chữ định danh trong thời gian chạy. Điều này bất chấp thực tế là một số ứng dụng có thể đòi hỏi liên kết từ một dạng sang một dạng khác nếu ứng dụng đó đã chốt một tùy chọn biểu diễn không được HL7 lựa chọn.

    Từ kinh nghiệm thực tiễn chúng tôi khuyến cáo rằng II.extensions, như một định danh chữ-số, không chứa các chữ số 0 ở phía trước (nếu có số 0 nào), do các số 0 này thường bị bỏ đi một cách không chính xác. "000123" và "123" là các giá trị phần mở rộng khác nhau, nhưng thường dễ bị hiểu nhầm, dẫn tới so lệch giả và lặp nhập bản ghi. Tuy nhiên, các ứng dụng nên duy trì bất kỳ chữ số 0 phía trước nào gặp phải trong II.extensions. Chữ số 0 phía trước bị cấm trong phần mở rộng của OID, nhưng có thể xuất hiện trong phần mở rộng của UUID, tại đây chúng phải được duy trì.

Không có đặc tính chữ số kiểm tra riêng biệt. Chữ số kiểm tra được dùng cho mục đích của con người và làm việc tốt nhất nếu được giữ hoàn toàn minh bạch. II.extensions CÓ THỂ chứa các chữ số kiểm tra ở bất cứ đâu, và sơ đồ chữ số kiểm tra cụ thể (nếu có) sẽ được ngụ ý bởi II.root. Tuy nhiên, đặc tả này cố ý không công nhận một đặc tính chữ số kiểm tra riêng biệt.

34. [nguồn] Kiểu dữ liệu của *sơ đồ* vẫn là CS và với mục đích của HL7, *sơ đồ* là một miền CNE. Điều này có vẻ mâu thuẫn với thực tế rằng không có một danh sách chính thức các sơ đồ URL, và rất nhiều sơ đồ URL đang sử dụng có thể được định nghĩa ở địa phương. Tuy nhiên, chúng ta không thể chấp nhận phần mở rộng của sơ đồ URL sử dụng cơ chế HL7 cho các hệ thống mã thay thế địa phương, đó là lí do vì sao về mặt kỹ thuật *sơ đồ* là một kiểu dữ liệu CS.
35. [nguồn] Cần nhớ rằng các đặc tính ngữ nghĩa không có tất cả các ngữ nghĩa luồng điều khiển. Phần được định dạng có thể được triển khai như một "thủ tục" sẽ "trả về" địa chỉ được định dạng, nhưng nó thường không phải là một biến để có thể gán cho nó một địa chỉ được định dạng. Tuy nhiên, HL7 không định nghĩa các ứng dụng mà chỉ định nghĩa ngữ nghĩa của các giá trị dữ liệu được trao đổi. Vì vậy, mô hình ngữ nghĩa rút ra từ các khái niệm như "thủ tục", "trả", và "gán" nhưng chỉ nói về đặc tính và giá trị.
36. [nguồn] Cần nhớ rằng các đặc tính ngữ nghĩa không có tất cả các ngữ nghĩa luồng điều khiển. Phần được định dạng có thể được triển khai như một "thủ tục" sẽ "trả về" địa chỉ được định dạng, nhưng nó thường không phải là một biến để có thể gán cho nó một địa chỉ được định dạng. Tuy nhiên, HL7 không định nghĩa các ứng dụng mà chỉ định nghĩa ngữ nghĩa của các giá trị dữ liệu được trao đổi. Vì vậy, mô hình ngữ nghĩa rút ra từ các khái niệm như "thủ tục", "trả", và "gán" nhưng chỉ nói về đặc tính và giá trị.
37. [nguồn] Các quy tắc định dạng địa chỉ này là một phần của ngữ nghĩa của địa chỉ do địa chỉ chủ yếu được định nghĩa là văn bản được hiển thị hay in ra và được con người sử dụng. Các trường hợp sử dụng khác (ví dụ, dịch tễ học) là thứ yếu — mặc dù không bị cấm, kiểu dữ liệu AD có thể phục vụ các trường hợp sử dụng khác này không tốt lắm, và HL7 định nghĩa các cách tốt hơn để xử lý các trường hợp sử dụng này. Lưu ý rằng các quy tắc định dạng này không phải là vấn đề của ITS, vì việc định dạng này áp dụng đối với các biểu diễn cho con người trong khi các đặc tả ITS là biểu diễn cho máy tính trao đổi.
38. [nguồn] Mã hóa XML được thể hiện ở đây là chỉ dựa theo XML ITS nhằm tránh đưa ra thêm một ký hiệu thực thể khác. Điều này không ngụ ý rằng chức năng này sẽ chỉ hoạt động trong XML, thậm chí cũng không ngụ ý rằng XML là biểu diễn được ưa chuộng.
39. [nguồn] Ví dụ này thể hiện sức mạnh của phương pháp đánh dấu đối với địa chỉ. Một hệ thống điển hình ở Đức lưu trữ số nhà và tên phố trong các trường riêng biệt sẽ in địa chỉ với tên phố trước tiên rồi đến số nhà. Đối với các địa chỉ

ở Mỹ, điều này là không đúng vì số nhà ở Mỹ được viết trước tên phố. Địa chỉ được đánh dấu cho phép giữ lại trật tự tự nhiên các phần của địa chỉ mà vẫn hiểu được vai trò của chúng.

40. [nguồn] Cần nhớ rằng các đặc tính ngữ nghĩa không có tất cả các ngữ nghĩa luồng điều khiển. Phần được định dạng có thể được triển khai như một "thủ tục" sẽ "trả về" tên được định dạng, nhưng nó thường không phải là một biến để có thể gán cho nó một tên được định dạng. Tuy nhiên, HL7 không định nghĩa các ứng dụng mà chỉ định nghĩa ngữ nghĩa của các giá trị dữ liệu được trao đổi. Vì vậy, mô hình ngữ nghĩa rút ra từ các khái niệm như "thủ tục", "trả", và "gán" nhưng chỉ nói về đặc tính và giá trị.
41. [nguồn] Cần nhớ rằng các đặc tính ngữ nghĩa không có tất cả các ngữ nghĩa luồng điều khiển. Phần được định dạng có thể được triển khai như một "thủ tục" sẽ "trả về" tên được định dạng, nhưng nó thường không phải là một biến để có thể gán cho nó một tên được định dạng. Tuy nhiên, HL7 không định nghĩa các ứng dụng mà chỉ định nghĩa ngữ nghĩa của các giá trị dữ liệu được trao đổi. Vì vậy, mô hình ngữ nghĩa rút ra từ các khái niệm như "thủ tục", "trả", và "gán" nhưng chỉ nói về đặc tính và giá trị.
42. [nguồn] Các quy tắc định dạng tên này là một phần của ngữ nghĩa của tên do các phần của tên được thiết kế với trường hợp sử dụng quan trọng là hiển thị và trình diễn trên nhãn. Lưu ý rằng các quy tắc định dạng này không phải là vấn đề của ITS, vì việc định dạng này áp dụng cho các biểu diễn cho con người trong khi các đặc tả ITS là biểu diễn cho máy tính trao đổi.
43. [nguồn] Trừu tượng hóa kiểu dữ liệu đại lượng tương ứng với khái niệm thang mức chênh lệch, trái ngược với thang thứ tự và thang tỷ lệ (Guttman và Stevens). Một kiểu dữ liệu chỉ với yêu cầu về thứ tự mà không có yêu cầu về mức chênh lệch sẽ là một thứ tự. Các thứ tự hiện chưa được định nghĩa với một kiểu dữ liệu đặc biệt. Thay vào đó, thứ tự thường là giá trị được mã hóa, trong đó hệ thống mã cơ bản xác định ngữ nghĩa thứ tự. Tuy nhiên, ngữ nghĩa thứ tự này không được phản ánh trong ngữ nghĩa kiểu dữ liệu HL7 vào thời điểm này.
44. [nguồn] H. Grassman. *Lehrbuch der Arithmetik.* 1861. Chúng tôi ưa dùng các tiên đề ban đầu của Grassman hơn các tiên đề của Peano, bởi vì các tiên đề của Grassman phù hợp cho tất cả số nguyên, chứ không chỉ cho các số tự nhiên. Ngoài ra, "nó khá nổi tiếng, qua thừa nhận của chính Peano, Peano đã mượn các tiên đề của mình từ Dedekind và đã sử dụng rộng rãi công trình của Grassmann trong việc phát triển các tiên đề của mình." (Hao Wang. Tiên đề hóa số học. J. Symb. Logic; 1957:22(2); tr. 145.)
45. [nguồn] Thuật ngữ "Thực" cho một kiểu dữ liệu số phân số bắt nguồn và được hình thành trong truyền thống của các ngôn ngữ lập trình Algol, Pascal.

46. [nguồn] Hãy tưởng tượng có một cái đồng hồ đặc biệt đo các chu kỳ, trong đó các kim không được gắn vào cùng một trục mà mỗi kim lại được gắn vào đầu của kim đo chu kỳ lớn hơn tiếp theo.
47. [nguồn] Vào thời điểm hiện tại, các đặc tính của Chu kỳ lịch là tổng và giá trị chưa được định nghĩa chính thức. Việc tính toán các chữ số lịch bao gồm một số tính toán phức tạp mà để có thể chỉ ra ở đây sẽ rất khó để hiểu và đánh giá được độ chính xác. Điều không may là chưa có tiêu chuẩn nào định nghĩa chính thức mối quan hệ giữa các biểu thức lịch và thời gian đã trôi qua kể từ một thời đại. ASN.1, đặc tả Kiểu dữ liệu sơ đồ XML và SQL92, tất cả đều tham khảo ISO 8601, tuy nhiên, ISO 8601 chỉ xác định cú pháp của các biểu thức lịch Dương, chứ không phải là ngữ nghĩa của chúng. Trong tiêu chuẩn này, chúng tôi định nghĩa cú pháp và ngữ nghĩa một cách chính thức, tuy nhiên, chúng tôi giả định ngữ nghĩa của đặc tính tổng và đặc tính giá trị đã được định nghĩa ở chỗ khác.
48. [nguồn] Vào thời điểm này, chưa có lịch nào khác ngoài lịch Dương được định nghĩa. Tuy nhiên, khái niệm lịch như một quy ước tùy ý để xác định thời gian tuyệt đối là quan trọng để định nghĩa một cách đúng đắn ngữ nghĩa của thời gian và các kiểu dữ liệu liên quan đến thời gian. Hơn nữa, các lịch khác có thể được hỗ trợ khi được cần đến để tạo điều kiện cho việc sử dụng HL7 trong các nền văn hóa khác.
49. [nguồn] Trong một số ngôn ngữ lập trình, "kiểu bộ sưu tập" được hiểu là cái chứa các hạng mục dữ liệu được liệt kê riêng lẻ, và do đó, một khoảng (thấp - cao) sẽ không được coi là một bộ sưu tập. Tuy nhiên, cách hiểu hẹp như vậy về "bộ sưu tập" lại phụ thuộc rất nhiều vào biểu diễn/triển khai. Trên quan điểm ngữ nghĩa kiểu dữ liệu, việc liệu một phần tử của một bộ sưu tập "có thực sự được chứa trong bộ sưu tập đó" hay không không quan trọng. Không nhất thiết là tất cả các phần tử trong một bộ sưu tập phải được liệt kê riêng lẻ.
50. [nguồn] Lưu ý sự khác biệt với GTS. GTS là bộ tạo sóng cho SET<TS> chứ không phải cho LIST<TS>. Một chuỗi các giá trị rời rạc từ một miền liên tục không có nhiều ý nghĩa ngoại trừ trong các ứng dụng lấy mẫu. Tuy nhiên, có thể hiểu SET<TS> như một chuỗi của IVL<TS>, vốn vẫn khác so với LIST<TS>.
51. [nguồn] Sự có mặt của quá nhiều lựa chọn xứng đáng được giải thích. Về nguyên tắc, chỉ cần dạng khoảng và dạng chỉ có độ rộng. Tuy nhiên, dạng khoảng tạo cảm giác xa lạ đối với nhiều người trong lĩnh vực tin học trong y tế. Một mục đích quan trọng của dạng chữ là để loại trừ những trường hợp không tuân thủ bằng cách khiến cho việc tuân thủ trở nên dễ dàng, mà không gây tổn hại đến tính đúng đắn của các khái niệm.

    Hơn thế nữa, tất cả các dạng chữ khác nhau đều có ưu và nhược điểm. Điểm mạnh của dạng khoảng và dạng trung tâm-độ rộng là chúng có tính chính xác

cao nhất, thể hiện các ranh giới đóng và mở. Tuy nhiên, điểm yếu của dạng khoảng là các ranh giới vô hạn đòi hỏi những ký hiệu đặc biệt cho các vô hạn, không nhất thiết ở dạng "bộ so sánh". Dạng trung tâm-độ rộng hoàn toàn không thể chỉ ra các khoảng có một ranh giới vô hạn. Tuy nhiên, dạng "bộ so sánh" chỉ có thể biểu diễn các khoảng có ranh giới đơn (nghĩa là, ranh giới còn lại là vô hạn hoặc không biết). Dạng gạch ngang, mặc dù là dạng yếu nhất so với tất cả các dạng khác, lại là dạng trực quan nhất cho các khoảng có ranh giới kép.

52. [nguồn] Phát biểu này có vẻ mâu thuẫn trực tiếp với quyết định về chuyển đổi tăng TS thành IVL<TS>. Tuy nhiên, không hề có mâu thuẫn nào. Độ chính xác của một ranh giới không có bất kỳ liên quan nào, nhưng độ chính xác của một dấu thời gian đơn giản (không phải như một ranh giới khoảng) lại liên quan, khi dấu thời gian đó được chuyển đổi tăng thành một khoảng.
53. [nguồn] Dạng bao có thể có vẻ dư thừa đối với chỉ riêng khoảng đơn giản. Tuy nhiên, dạng bao sẽ trở nên quan trọng đối với biểu diễn khoảng định kỳ do nó rút ngắn phép biểu diễn và (có thể gây tranh cãi) khiến biểu diễn của các cấu trúc định thời gian phức tạp hơn trở nên trực quan hơn.
54. [nguồn] Đặc tả này tự áp đặt một giới hạn lên chính nó để cho phép các hệ thống hiện có được chuyển tiếp một cách uyển chuyển. Tuy nhiên, đặc tả chính thức vẫn để các phần mở rộng kiểu chung là có thể thay thế cho các kiểu cơ bản của chúng. Sự tự giới hạn này có thể được loại bỏ trong tương lai. Các triển khai mới được khuyến nghị cung cấp một số hỗ trợ có thể tổng quát hóa cho các phần mở rộng kiểu dữ liệu chung này.
55. [nguồn] Lưu ý rằng các kiểu dữ liệu là đặc tả của các đặc tính trừu tượng của giá trị. Đặc tả này không quy định bắt buộc cách thức biểu diễn các giá trị này trong một ITS hay triển khai chúng trong một ứng dụng. Cụ thể, nó không quy định bắt buộc cách thức đặt tên hay định vị các thành phần được biểu diễn. Ngoài ra, hệ thống phân cấp khái quát ngữ nghĩa có thể khác so với hệ thống phân cấp lớp được lựa chọn cho triển khai (nếu công nghệ triển khai có thừa kế). Cần nhớ sự khác biệt giữa một kiểu (giao diện) và một triển khai (cấu trúc dữ liệu cụ thể, lớp). ITS phải chứa liên hệ giữa các đặc điểm được ITS định nghĩa của bất kỳ kiểu dữ liệu nào với các đặc điểm ngữ nghĩa được định nghĩa ở đây.
56. [nguồn] *GTS* là một ví dụ về kiểu dữ liệu chỉ được định nghĩa về mặt đại số mà không đưa ra bất kỳ định nghĩa nào về cấu trúc dữ liệu có thể triển khai hành vi của kiểu dữ liệu đó. Định nghĩa đại số trông có vẻ cực kỳ đơn giản, tới mức có người có thể cho rằng nó chưa hoàn thiện. Do vào thời điểm này chúng tôi dựa hoàn toàn vào dạng chữ để biểu diễn các giá trị *GTS*, tất cả định nghĩa của cấu trúc dữ liệu.
57. [nguồn] Đặc tính xen kẽ có thể có vẻ quá hạn chế. Tuy nhiên, các hạn chế này là hợp lý đối với trường hợp sử dụng mà trong đó các đặc tính xen kẽ và bao

định kỳ được định nghĩa. Để có thể kết hợp một cách an toàn và dễ dự đoán hai lịch trình, người ta sẽ muốn biết toán hạng nào thiết lập điểm khởi đầu và toán hạng nào thiết lập điểm kết thúc của các khoảng xảy ra của bao định kỳ.

58. [nguồn] Đặc tả dạng chữ này lại trông có vẻ đơn giản đến ngạc nhiên, vì thế mà có thể sẽ có người cho là nó chưa hoàn thiện. Tuy nhiên, dạng chữ *GTS* được dựa trên dạng chữ TS, IVL, PIVL, và EIVL và cũng ngụ ý dạng chữ cho các phần mở rộng của TS, đáng chú ý là PPD<TS>. Bản thân đặc tả dạng chữ *GTS* chỉ cần liên kết các dạng chữ khác lại với nhau, mà đây thực ra lại là một nhiệm vụ khá đơn giản.