tả đầy đủ bằng ngôn ngữ JML; JML có nhiều công cụ hỗ trợ cho việc kiểm tra chương trình như ESC/Java [25], KeY [26],... Do vậy, trong phần tiếp theo của bài báo này, JML là ngôn ngữ đặc tả hình thức được chúng tôi lựa chọn để tiếp tục phát triển cho khung thức tìm kiếm, tổng hợp và tái sử dụng hàm API tự động được đề cập trong các phần sau.

Bảng 1: So sánh các ngôn ngữ đặc tả hình thức theo hướng thoả thuận

	JML	Spec#	ACSL	Jahob	ADA 2012
Ngôn ngữ chủ	Java	Mở rộng của C#	C	Tập con của Java	ADA 2012
Cách thức đặc tả	Chèn vào dưới dạng chú giải trong các chú thích	Đưa thêm từ khóa vào ngay trong ngôn ngữ	Chèn vào dưới dạng chú giải trong các chú thích	Chèn vào dưới dạng chú giải trong các chú thích	Đưa thêm từ khóa vào ngay trong ngôn ngữ
Các thành phần đặc tả	Tiền điều kiện Hậu điều kiện Bất biến Ngoại lệ	Tiền điều kiện Hậu điều kiện Bất biến Ngoại lệ	Tiền điều kiện Hậu điều kiện Bất biến Hỗ trợ con trỏ	Tiền điều kiện Hậu điều kiện Bất biến Cấu trúc dữ liệu	Tiền điều kiện Hậu điều kiện Bất biến
Mục đích chính	Kiểm tra tính đúng của phương thức được hiện thực so với đặc tả	Kiểm tra tính đúng của phương thức được hiện thực so với đặc tả	Kiểm tra tính đúng của hàm được hiện thực so với đặc tả	Kiểm tra tính đúng của cấu trúc dữ liệu được hiện thực so với đặc tả	Kiểm tra tính đúng của hàm/ thủ tục được hiện thực so với đặc tả
Hỗ trợ đặc tả phi hình thức	Có	Không có	Không có	Không có	Không có
Số lượng công cụ hỗ trợ/ sử dụng	Nhiều (ECS/Java, KeY, JMLUnint,…)	Boogie	Frama-C	Hệ thống Jahob	Bản thân trình biên dịch ADA 2012

3 VÀ TÁI SỬ DỤNG HÀM API TỰ ĐỘNG

Hình 7 mô tả khung thức cho việc tìm kiếm và tái sử dụng hàm API tự động. Khung thức này có ba khối chức năng chính sau đây:

3.1 Bộ đánh giá sự tương tự giữa các đặc tả

Khối chức năng này giải quyết bài toán cơ bản để thực hiện việc tìm kiếm các đặc tả phù hợp với một yêu cầu đầu vào. Đó là đánh giá sự tương tự giữa các công thức logic đặc tả các hàm. Nhờ vậy, chúng ta có thể thu hẹp không gian tìm kiếm hàm. Việc thu hẹp này dựa trên sự tính toán mức độ tương tự của đặc tả yêu cầu với các đặc tả của hàm API có trong thư viện.

Nguyên lý cho việc phân loại hàm dựa trên học máy là với một tập các hàm đã cho (P) và một lời yêu cầu (c), giải thuật sẽ *tiên đoán* những hàm từ P có khả năng thỏa mãn được yêu cầu của c. Hướng tiếp cận này dựa trên một số định nghĩa sau:

Định nghĩa 1: Ma trận phụ thuộc

Gọi Γ là một tập đặc tả được biểu diễn dưới dạng logic của các hàm thư viện. Khi đó chúng ta định nghĩa:

$$\mu = \Gamma \times \Gamma \rightarrow \{0,1\}$$

$$\mu(c,p) = \begin{cases} 1 & \text{nếu } p \text{ có khả năng thỏa mãn được yêu cầu của } c \\ 0 & \text{ngược lại} \end{cases}$$

Định nghĩa 2: Ma trận đặc trưng

Gọi $T=\{t_1, ..., t_m\}$ là tập các mệnh đề (term) đặc tả trong Γ. Chúng ta định nghĩa hàm Φ như sau:

$$\Phi = \Gamma \times \{1, ..., m\} \rightarrow \{0,1\}$$

$$\Phi(c,i) = \begin{cases} 1 & \text{nếu } t_i \text{ xuất hiện trong } c \\ 0 & \text{ngược lại} \end{cases}$$

Định nghĩa 3: Hàm đặc trưng