Enhancing DNN-Based Binary Code Function Search With Low-Cost Equivalence Checking.

Nâng cao Tìm kiếm Hàm Mã nhị phân dựa trên DNN với Kiểm tra Tương đương Chi phí thấp.

Tóm tắt:

* Bài báo này đề xuất một giải pháp tổng quát để cải thiện các ứng cử viên đạt hạng cao nhất trong tìm kiếm hàm mã nhị phân dựa trên DNN.
* Chúng tôi thiết kế một kiểm tra tương đương thực tế và hiệu quả, được đặt tên là BinUSE.
* BinUSE được thiết kế đặc biệt để cung cấp một kiểm tra tương đương mức độ hàm lắp ráp, cải thiện tìm kiếm mã nhị phân dựa trên DNN bằng cách giảm thiểu các cảnh báo sai với chi phí thấp.

1. Giới thiệu:

* Để học các biểu diễn mã lệnh, mô hình DNN được huấn luyện với các đặc trưng cấp từ vựng, cấu trúc điều khiển hoặc luồng dữ liệu (nhẹ).
* Mặc dù việc trích xuất các biểu diễn như vậy dễ dàng, chúng có thể không bảo tồn ngữ nghĩa chương trình một cách đáng kể. Ngoài ra, các đặc trưng nhẹ thường không chịu được thử thách như tối ưu hóa trình biên dịch hay làm tròn mã, làm cho mã lắp ráp có cùng ý nghĩa dường như khác nhau đáng kể.
* Do đó, các mô hình DNN có thể có tính phân biệt thấp và tính ổn định thấp, dẫn đến một số lượng lớn cảnh báo sai trong các ứng viên hàng đầu lấy ra.
* Để thiết kế một kiểm tra tương đương giá rẻ và thực tế, chúng tôi xây dựng và kiểm tra các mối quan hệ đầu vào-ra của các hàm lắp ráp bằng cách sử dụng kỹ thuật giải ràng buộc và thực thi biểu tượng không bắt buộc (USE).
* So với thực thi biểu tượng tiêu chuẩn, USE được thực hiện để thực hiện luận lý biểu tượng linh hoạt và nhanh chóng trực tiếp từ các điểm vào của hàm, bỏ qua việc xử lý chi phí từ hàm main đến hàm mục tiêu.
* Chúng tôi tối ưu hóa lược đồ USE tiêu chuẩn như một công cụ thực tế, gọi là BinUSE.
* BinUSE khởi động duyệt USE từ điểm vào của hàm và duyệt qua mỗi đường dẫn cho đến khi đạt đến điểm gọi hàm bên ngoài đầu tiên, đánh dấu một nút thông tin và quan trọng trên CFG.
* Sau đó, BinUSE sử dụng các công thức biểu tượng của đầu vào điểm gọi hàm bên ngoài để hình thành các ràng buộc biểu tượng của mỗi đường dẫn, và để khớp hai hàm, BinUSE khám phá các ràng buộc biểu tượng phù hợp được thu thập từ mỗi đường dẫn trong mỗi hàm.

Tóm tắt, chúng tôi đóng góp như sau:

* Tại mức độ khái niệm, chúng tôi đề xuất một quan điểm mới để cải thiện tìm kiếm hàm mã nhị phân dựa trên DNN.
* Tại mức kỹ thuật, chúng tôi trình bày một kiểm tra tương đương bằng cách tối ưu hóa lược đồ USE tiêu chuẩn để giảm chi phí thêm nữa.
* Tại mức thực nghiệm, đánh giá của chúng tôi cho thấy rằng kiểm tra tương đương được thiết kế là tổng quát và hiệu quả để cải thiện các công cụ tìm kiếm hàm nhị phân dựa trên DNN với chi phí thấp.

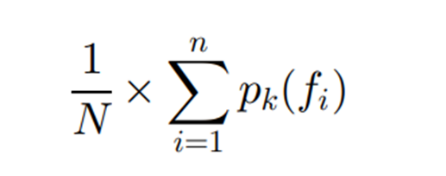
1. Kiến thức sơ bộ:
2. Công thức và đánh giá:

* Hình 1 minh họa một thiết lập đánh giá thường xuyên được sử dụng và khó khăn trong lĩnh vực này, trong đó chúng ta chuẩn bị hai chương trình thực thi Bin1 và Bin2 được biên dịch từ cùng một chương trình sử dụng các cài đặt biên dịch khác nhau.

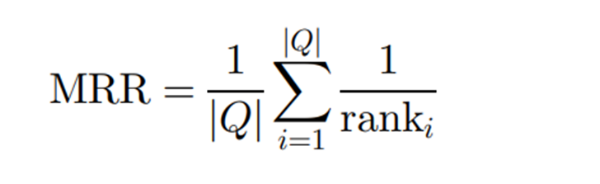
A diagram with numbers and words

Description automatically generated with medium confidence

* So sánh từng cặp hàm cho ra một điểm tương đồng, và độ chính xác hàng đầu k được định nghĩa như sau:



* Giải thích công thức: Trong đó N là tổng số hàm trong chương trình. Để hiểu công thức tiêu chuẩn về độ chính xác hàng đầu k: chúng ta lần lượt lấy mỗi hàm fi trong Bin1 làm hàm mục tiêu để truy vấn RP được hình thành bởi tất cả các hàm trong Bin2. Để tính toán độ chính xác hàng đầu k, cho phép khớp chính xác (tức là đúng) của fi là f0\_i. Trong công thức 1, pk(fi) cho ra giá trị 1 nếu f0\_i nằm trong danh sách hàng đầu k ứng cử viên được lấy ra, trong khi pk(fi) = 0 nếu f0\_i không nằm trong đó.
* Trong việc đánh giá trong Hình 1, thiết lập tiêu chuẩn và thông thường là coi một hàm f0\_i chia sẻ cùng tên hàm với fi là sự khớp chính xác.
* Một phương pháp đo thông thường khác được gọi là điểm số trung bình hạng đảo nghịch (MRR), có thể tính toán theo cách sau:



* Giải thích công thức: Trong đó |Q| biểu thị tổng số lượng truy vấn được thực hiện đối với RP, và rank\_i đại diện cho thứ hạng của sự khớp chính xác cho truy vấn thứ i được trả về bởi công cụ tìm kiếm hàm. Điểm số MRR càng lớn thì tìm kiếm hàm càng chính xác hơn. Ví dụ, cho ba truy vấn đối với RP, trong đó các sự khớp chính xác cho mỗi truy vấn được đặt ở vị trí thứ 4, thứ 2 và thứ 6 trong danh sách ứng cử viên được lấy ra. Sau đó, MRR được tính toán là 1/3 \* (1/4 + 1/2 + 1/6) = 0.31.

1. Kiểm tra tương tự mã nhị phân:

* Ngoài việc học biểu diễn dựa trên DNN phổ biến, một dòng nghiên cứu khác là thực hiện kiểm tra tương đương mã, sử dụng các mối quan hệ đầu vào-ra của chương trình được thu thập thông qua thực thi biểu tượng (SE).
* Ví dụ Thực hành. Bây giờ chúng tôi sẽ đưa ra một ví dụ thực hành đơn giản để minh họa cách kiểm tra tính tương đương của chương trình sử dụng SE. Xem đoạn mã dưới đây:

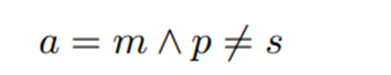
A white rectangular object with black letters

Description automatically generated

* Ở đây, a và m là hai giá trị đầu vào của hai đoạn mã lập trình tương ứng. SE đại diện cho các đầu vào dưới dạng biểu tượng tự do và diễn giải mỗi câu lệnh trong miền biểu tượng. Trong trường hợp của chúng tôi, các công thức đầu ra được hiển thị như sau:



* Và tính tương đương của các đoạn mã trên có thể được kiểm tra bằng cách xây dựng ràng buộc sau đây:



* Giải thích công thức: Trong đó bộ giải ràng buộc tìm kiếm không gian đầu vào của a và m để kiểm tra xem có hai đầu vào a và m làm cho các công thức đầu ra không tương đương. Trong trường hợp bộ giải ràng buộc không tìm thấy các giải pháp hợp lệ (tức là bộ giải ràng buộc trả về unsat), hai đoạn mã này sẽ được kiểm tra một cách nghiêm ngặt là tương đương.
* Mặc dù kiểm tra tính tương đương không thể được sử dụng để tính toán trực tiếp độ chính xác hàng đầu k, chúng tôi sử dụng kiểm tra tính tương đương kích hoạt bởi USE để loại bỏ các cảnh báo sai được tạo ra bởi các mô hình DNN, như sẽ được giới thiệu trong Phần 4.

1. Thực thi Biểu tượng Bất định hướng

* Để giải quyết vấn đề về tính mở rộng của SE, USE được đề xuất để kiểm tra trực tiếp các thành phần mã tùy ý (ví dụ: một hàm) thay vì toàn bộ chương trình, từ đó giảm thiểu độ phức tạp của SE.
* Để minh họa sự khác biệt kỹ thuật ở cấp độ cao (về mặt độ bao phủ đường đi) giữa SE và USE.
* Hình 2a trình bày một trường hợp phân tích một chương trình giải mã thông điệp và xác định một lỗi trong hàm decoding\_msg.
* Hàm chính nhận thông điệp đã được mã hóa với hàm receive\_msg và thực hiện quá trình giải mã trong một câu lệnh lặp. Thông điệp giải mã được chuyển đến hàm decoding\_msg, nơi một lỗi (đánh dấu là bug tại dòng 15 trong Hình 2a) được ẩn trong nhánh if.
* SE có thể bị trở ngại trong việc phân tích trường hợp đơn giản này do sử dụng tài nguyên tính toán cao và các ràng buộc dài dòng.
* Như được thể hiện trong Hình 2b, SE bắt đầu từ điểm nhập cửa chính để giải thích các câu lệnh của chương trình với các biến biểu tượng. Khi phân tích câu lệnh lặp, các biến biểu tượng trong bộ nhớ được tạo ra ngày càng nhiều, và mỗi giá trị biểu tượng được xử lý bởi decoding có thể tiếp tục tăng lên. Do đó, việc sử dụng bộ nhớ có thể trở nên không thực tế và các ràng buộc biểu diễn được tạo ra có thể quá phức tạp để giải quyết.
* USE giảm độ phức tạp theo một cách có nguyên tắc. Để đạt được bug, USE trực tiếp phân tích decoding\_msg. Kết quả đường đi thu được, như được hiển thị trong Hình 2c, không đặt ra ràng buộc phức tạp nào cho thông điệp giải mã msg22 và có thể tạo ra ràng buộc dễ dàng hơn nhiều để giải quyết. Phân tích toàn bộ chương trình đắt tiền hơn có thể được trì hoãn cho đến khi cần thiết.

A screenshot of a computer program

Description automatically generated

1. Phạm vi ứng dụng:
2. Động lực nghiên cứu:

* Các biểu diễn nhúng đã học chủ yếu chịu hai hạn chế sau đây:
* Độ phân biệt thấp. Hạn chế này cho biết rằng các mô hình DNN có thể xem xét các hàm logic khác nhau như là tương tự nhau. Do đó, độ phân biệt thấp dẫn đến việc báo cáo nhiều kết quả phù hợp sai dương (FP).
* Độ bền thấp. Tổng thể, độ bền đề cập đến khả năng chống chịu đối với nhiều điều kiện không hoàn hảo khi chạy phần mềm hoặc thuật toán. Do đó, độ bền thấp ngụ ý rằng các mô hình DNN có thể gặp khó khăn khi phù hợp các hàm chia sẻ cùng một logic nhưng có vẻ khác nhau về mặt cú pháp. Nói chung, độ bền thấp dẫn đến việc báo cáo nhiều kết quả phù hợp sai âm (FN).
* <Phần này nói khá nhiều nội dung về các hạn chế của các phương pháp trước đây nên em chỉ trình bày ngắn gọn và chuyển sang phần kiến trúc của BinUSE>.

1. Thiết kế của BinUSE:

* Phần này giới thiệu thiết kế của BinUSE, một công cụ kiểm tra tính tương đương dựa trên USE.
* Các thủ tục để điều chỉnh việc truy xuất hàng đầu-k của một mô hình DNN bằng BinUSE sẽ được trình bày trong Mục 6.
* Hình 7 mô tả quy trình làm việc tổng quan của BinUSE.

A diagram of a workflow

Description automatically generated

* Khi có một tệp thực thi đầu vào, BinUSE đầu tiên thực hiện kỹ thuật khôi phục thông tin hàm hợp ngữ.
* Sau đó, nó bắt đầu từ điểm nhập của mỗi hàm hợp ngữ để khởi chạy việc duyệt USE từng đoạn đường (Hình 7a), trong đó mỗi đoạn đường dừng lại khi đạt đến điểm gọi bên ngoài đầu tiên.
* Kết quả, một đồ thị con sẽ được tạo ra, trong đó mỗi nút lá tương ứng với một điểm gọi bên ngoài.
* Để so sánh hai hàm hợp ngữ, BinUSE so sánh các đồ thị con được tạo ra từ chúng (Hình 7b), bằng cách khởi chạy giải quyết ràng buộc để kiểm tra tính tương đương về mặt ngữ nghĩa của đầu vào điểm gọi bên ngoài và ràng buộc đoạn đường.
* Trước khi trình bày về thiết kế của BinUSE, chúng tôi xác định rõ giả định của mình về kỹ thuật khôi phục ngược và cơ sở.
* Giả định về Kỹ thuật khôi phục ngược. Phân tích của chúng tôi tập trung vào cấp độ hàm, và không giả định sự có mặt của ký hiệu chương trình hoặc thông tin gỡ lỗi. Nhị phân đã bị cắt giảm có thể được xử lý mà không gặp khó khăn bổ sung, miễn là các hàm đã được xác định để sử dụng.

1. Cơ sở (Baseline):

* Trong khi thiết kế này cho phép mô hình hóa đầy đủ các mối quan hệ đầu vào-ra, nghiên cứu ban đầu của chúng tôi đã chỉ ra khả năng thấp về tỷ lệ mở rộng.
* Cây CFG (Control Flow Graph) nội hàm được tạo ra thường rất phức tạp. Ngoài ra, chúng tôi đã triển khai thuật toán CoP tiêu chuẩn [4] để giảm độ phức tạp bằng cách trích xuất dãy con chung dài nhất (LCS) của các khối cơ bản tương đương trên một đoạn đường.
* Tuy nhiên, một số chi tiết thực thi (chẳng hạn như phân chia và hợp nhất khối) vẫn còn chưa biết. Trong khi phương pháp USE chủ yếu giảm độ phức tạp của toàn bộ phân tích SE, nghiên cứu thử nghiệm của chúng tôi trong việc phân tích các chương trình thực thi trong thế giới thực đã chỉ ra nhu cầu đòi hỏi phải hiệu chỉnh và tối ưu hóa thêm.

1. Tạo đồ thị con từ CFG:

* Xem xét khó khăn trong việc khám phá toàn bộ đồ thị điều khiển của một hàm, chúng tôi trích xuất một đồ thị con.
* Đồ thị con được trích xuất nên giữ lại các đặc trưng đại diện của CFG tương ứng và giảm đáng kể độ phức tạp phân tích.
* BinUSE được thiết kế để duyệt qua mọi đoạn đường thực thi từ điểm nhập của mỗi hàm mã lệnh.
* Khi phân tích một đoạn đường thực thi, BinUSE nạp chồng mỗi điểm gọi hàm trên đoạn đường đó mà có triển khai trong mã nhị phân. BinUSE dừng lại khi gặp điểm gọi hàm bên ngoài đầu tiên.

1. So sánh hai đồ thị con:

* Hình 8 mô tả các quy trình cần thiết để so sánh hai đồ thị con Gt và Gs tương ứng được xuất phát từ hàm mã lệnh ft và fs.

A close-up of a computer code

Description automatically generated

* Như được thể hiện trong Hình 8a, việc duyệt qua CFG và tiêu chuẩn dừng được đề xuất trong Phần 5.2 tạo thành một đồ thị con với mức độ phức tạp hợp lý từ mỗi CFG, với mỗi lá tượng trưng cho một điểm gọi hàm bên ngoài. Kết quả là việc so sánh hai CFG trở thành việc so sánh hai đồ thị con này: chúng ta so sánh từng điểm gọi hàm theo từng bước cho đến khi tìm ra một hoán vị làm cho các điểm gọi hàm bên ngoài trong Gt và Gs cùng tương đương với nhau.
* Trong khi so sánh từng cặp điểm gọi hàm trong Gt và Gs có thể đưa ra nhiều hoán vị, chúng tôi làm rõ rằng chúng tôi chỉ tiến hành kiểm tra tương đương nặng (xem Hình 8d) khi hai điểm gọi hàm tham chiếu đến cùng một hàm bên ngoài. Tên hàm bên ngoài có thể được lấy từ mã nhị phân ELF ngay cả khi đã loại bỏ. Ví dụ, chúng tôi sẽ so sánh hai điểm gọi hàm trong Gt và Gs trong trường hợp chúng đều tham chiếu đến fopen.
* Thay thế cuộc gọi thư viện C. Khi một số tham số đầu vào của cuộc gọi thư viện C là hằng số, trình biên dịch có thể thay thế cuộc gọi thư viện C này bằng các cuộc gọi khác.
* Việc tìm các điểm gọi thư viện phù hợp được coi là một bước quan trọng trong quá trình kiểm tra tính tương đương, và để thực hiện điều này, chúng tôi đã thu thập thủ công danh sách sau đây, trong đó mỗi mục nhập cũng là một danh sách L.
* BinUSE được xây dựng để xem xét các điểm gọi thư viện trong mỗi L như là như nhau.