Giải thuật chính :

Ban đầu chúng ta khởi tạo list\_proof và list\_goal, list\_proof có thể trống và có Gn = Go . Tiếp theo chúng ta kiểm tra xem G0 có mặt hay chưa? Giả sử G0 chưa có mặt, chúng ta tiến hành đi tìm các công thức mới bằng cách cố gắng áp dụng eliminate vào list\_proof. Nếu công việc này thành công, chúng ta sẽ quay lại kiểm G0 có mặt trong list\_proof hay chưa? Nếu list\_proof vẫn chưa có mặt thì chúng ta sẽ tiến hành phân rã G0 ra . Tùy thuộc vào cấu trúc của G0 , mà việc phân rã khác nhau. Lúc đó, thay vì đi chứng minh G0 chúng ta đi chứng minh Gn nào đó…

Trường hợp đặc biệt:

1. Eliminate OR:

Một cách tự nhiên, nếu có A->C, B-> C thì có được A|B -> C. Công thức này do chính chúng tôi đưa ra . Do đo để áp dụng eliminate OR vào A|B thì chúng ta A->C, B->C. Khả năng tìm được A->C, B->C trong list\_proof là rất thấp .Vì thế , thay vì chỉ tác động lên list\_proof , chúng tôi đi chứng minh công thức A->C, B->C với C là current goal.

Xét bài toán

{ P0, P1,P2 … A|B … Pn } |- { G0 G1 Gn}

Để áp dụng eliminate OR lên A|B với mục tiêu đi chứng minh Gn. Chúng ta thêm A->Gn, B->Gn vào list\_proof. Khi đó ,sau khi có A->Gn và B->Gn, chúng ta có áp dụng luật OR Eliminate vào A|B, Gn[A].Gn[B] để có được Gn.

{ P0, P1,P2 … A|B(\*) … Pn } |- { G0 G1 Gn, Gn[A].Gn[B] }.

A|B

If A

…

Nif C

If B

…

Nif C

C

Xét bài toán :

{A|B, C, C-> D} |- {D}

Dễ dàng tìm ra lời giải cho bài toán bằng cách áp dụng -> e vào C, C->D. Nhưng nếu chúng ta đã áp dụng |e vào A|B, thì rõ ràng D chịu sự ảnh hưởng của A|B

Tức là :

A|B

If A

Nif D

If B

Nif D

D

Chúng ta cần loại bỏ sự ảnh hưởng vô nghĩa của A|B. Để giải quyết vấn đề này chúng ta cần phải lưu lại nguồn gốc của dẫn xuất. Nếu trong trường hợp này D không hề có được dẫn xuất từ A|B thì sự ảnh hưởng sẽ bị loại bỏ.

Xét bài toán khác:

{F->H} |- { F|G -> H|G|R}

Bài giải được xem là hoàn hảo:

F->H

If F|H

If F

Nif H|G

If G

Nif H|G

H|G|R

Thế nhưng, với giải thuật trên bài toán sẽ được giải là

F->H

If F|H

If F

Nif H|G|R

If G

Nif H|G|R

H|G|R

Rõ ràng sự ảnh hưởng của F|H chỉ tác động trực tiếp lên H|G còn H|G|R chi chịu ảnh hưởng từ H|G. Do đó chúng ta cần xác định lại phạm vi ảnh hưởng của F|H lên các dẫn xuất

1. Eliminate exists:

Nếu f(x0) với x0 là giá trị bất kỳ thỏa mãn f(x0) là một công thức và với f(x0) , chúng ta dẫn xuất được công thức G. Như thế chúng ta sẽ có exists x f(x) -> G bằng cách áp dụng luật exists eliminate.

Mô hình

If x 0 f(x0)

Nif G

G

Xét bài toán {} |- {}

Để áp dụng –]e vào -]x f(x), chúng ta phải tìm một giá trị x0 chưa từng xuất hiện trong các dẫn xuất trước đó. Thêm vào công thức f(x0) vào list\_proof đồng thời xóa sự có mặt của -] x f(x) . Có thể -] x f(x) và f(x0) không hoàn toàn tương đương nhau nhưng trong trường hợp này là tương đương vì từ -] xf(x) chúng ta có được f(x) và ngược lại.

F(x0) sẽ tồn tại xuyên suốt quá trình chứng minh, vấn đề đặt ra là dẫn xuất nào chịu ảnh hưởng, dẫn xuất nào không bị ảnh hưởng.

Xét bài toán :

{Exists x f(x), all x f(x) -> G } |- {G|H}

Bài chứng minh

If xo f(xo)

F(x0) ->G

Nif G

G|H

Nif G chính là điểm dừng sự ảnh hưởng của f(x0) lên tập dẫn xuất.

Rõ ràng với giải thuật trên chưa , vấn đề này giải quyết được.

Để giải quyết vấn đề này chúng ta bắt đầu thì G0, đi dọc theo các goal con nó để chọn điểm cắt theo hình sau

Áp dụng của giải thuật tìm kiếm chứng minh:

Ở phần này chúng tôi sẽ trình chi tiết từng bước trong giải thuật trên .

1. Thí dụ F ->H |- F|G -> H|G

Ở bước khởi tạo chúng ta sẽ được 2 danh sách list\_proof và list\_goal như sau

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H | Go = {F|G -> H|G} |

Tiếp theo chúng ta thực hiện elimination cho tập proof. Việc eliminate thất bại vì có F->H nhưng không tìm thấy F.

Do đó trạng thái hiện tại của list\_proof và list\_goal vẫn không đổi

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H | Go = {F|G -> H|G} |

Theo giải thuật nếu việc eliminate thất bại, chúng ta tiến hành thực thi introduction lên current goal. Current goal chính là G0 và G0 có dạng A->B. Dựa theo module Introduction, giả thiếtF|G sẽ được thêm vào list\_proof và H|G sẽ được thêm vào list\_goal . G0 sẽ được thiết lập thuộc tính pending là 1.

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H   1. F|G | Go = {F|G -> H|G}  {F|G->H|G, H|G} |

Sau khi introduction thành công , current goal sẽ được thiết lập lại bằng G1 = {H|G}. Theo giải thuật, chúng ta sẽ quay lại bước 1 tức là sẽ đi kiểm tra H|G có mặt trong tập list\_proof hay chưa? Kết quả H|G chưa có mặt trong list\_proof.

Chúng ta tiến hành thực thi module elimination lên tập list\_proof. Lúc này list\_proof có 2 phần tử F->H và F|H . Module Elimination thực hiện theo breadth first search nên sẽ duyệt các phần tử từ trái sang phải . F->H không thể eliminate được vì không tìm được F. Công thức còn lại F|G có thể eliminate theo trường hợp đặc biệt OR. Theo (1) việc tiến hành eliminate OR sẽ thay đổi list\_goal và list\_proof như sau. List\_goal sẽ được thêm vào 2 công thức H|G [F] và H|G [G] còn Go = {F|G -> H|G} được thiết lập thuộc tính pending là 2.

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H   1. F|G ass | Go = {F|G -> H|G}  {F|G->H|G, H|G}  {F|G->H|G, H|G , H|G[F], H|G[G]} |

Lúc này chúng ta lại quay về bước 1. Current goal được thiết lập là H|G [G] . Tức là tìm H|G với điều kiện G. Thêm giả thiết G vào tập list\_proof, đánh dấu G là giả sử của H|G.

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H   1. F | G ass 2. F ass | Go = {F|G -> H|G}  {F|G->H|G, H|G}  {F|G->H|G, H|G , H|G[F], H|G[G]}  {F|G->H|G, H|G , H|G[F], H|G} |

Lúc này tiến hành tìm kiếm sự có mặt của H|G. Dễ dàng nhận ra H|G không có mặt trong tập list\_proof. Tiến hành áp dụng elimination lên list\_proof. F|H áp dụng thành công elimination nhờ có F. Do đó công thức H sẽ được thêm vào list\_proof

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H   1. F | G ass 2. F ass 3. H | Go = {F|G -> H|G}  {F|G->H|G, H|G}  {F|G->H|G, H|G , H|G[F], H|G[G]}  {F|G->H|G, H|G , H|G[F], H|G} |

Việc elimination thành công, chương trình quay về bước 1, tiến hành xác định H|G có mặt hay chưa? Kết quả là H|G vẫn chưa có mặt. Tiến hành elimination tiếp tục và thất bại. Do không thể áp dụng eliminate vào list\_proof nên phải “rã” current goal. Việc rã sẽ được thực hiện theo OR Introduction . Do H|G rã lần đầu tiên nên list\_goal sẽ thêm vào công thức thứ nhất là H

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H   1. F | G ass 2. F ass 3. H ->e,.. | Go = {F|G -> H|G}  {F|G->H|G, H|G}  {F|G->H|G, H|G , H|G[F], H|G[G]}  {F|G->H|G, H|G , H|G[F], H|G}  {F|G->H|G, H|G , H|G[F], H|G,H} |

Quay về bước 1, tiến hành tìm kiếm sự có mặt của H trong list\_proof. H có mặt. Như vậy goal H đã được chứng minh. Lấy goal H ra khỏi list\_goal. Thiết lập current goal là H||G. Quay về bước 1.

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H   1. F | G ass 2. F ass 3. H ->e,.. | Go = {F|G -> H|G}  {F|G->H|G, H|G}  {F|G->H|G, H|G , H|G[F], H|G[G]}  {F|G->H|G, H|G , H|G[F], H|G}  {F|G->H|G, H|G , H|G[F], H|G} |

Current goal là H|G và H|G có pending bằng H|G tức là 1 được sinh ra từ 1 goal kế trước nó. Lúc introduction , tên ND rule đã được thiết lập cho H|G là |I nên dễ dàng có được H|G trong list\_proof

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H   1. F | G ass 2. F ass 3. H ->e,..   H|G |I .. | Go = {F|G -> H|G}  {F|G->H|G, H|G}  {F|G->H|G, H|G , H|G[F], H|G[G]}  {F|G->H|G, H|G , H|G[F], H|G}  {F|G->H|G, H|G , H|G[F], H|G} |

Loại bỏ H|G ra khỏi list\_goal. Ở bước này , đồng thời xóa sự có mặt của giả sử H.

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H   1. F | G ass 2. F ass 3. H (-) ->e,..   H|G |I .. | Go = {F|G -> H|G}  {F|G->H|G, H|G}  {F|G->H|G, H|G , H|G[G], H|G[F]}  {F|G->H|G, H|G , H|G[G], H|G}  {F|G->H|G, H|G , H|G[F]} |

Thiết lập current goal là H|G[F], thêm công thức G vào list\_proof và đồng thời biến H|G[F] thành H|G.

|  |  |
| --- | --- |
| List\_proof | List\_goal |
| F->H   1. F | G ass 2. F ass 3. H (-) ->e,..   H|G |I ..  H | Go = {F|G -> H|G}  {F|G->H|G, H|G}  {F|G->H|G, H|G , H|G[F], H|G[G]}  {F|G->H|G, H|G , H|G[F], H|G}  {F|G->H|G, H|G , H|G[F]} |