

**ĐA-I HQC QUOC GIA HÀ NQI TRƯONG ĐA-I HQC CÔNG NGH:¢**

## Nguy@n Thj Vân Anh

**KET HQP SINH BAT BIEN VÒNG LA-P VÀ KY THU�T TH{JC THI TƯQNG TRƯNG TRONG KIEM CHUNG MQT SO TÍNH CHAT**

**CUA CHƯƠNG TRÌNH C/C++**

### KHÓA LU�N TOT NGHI:¢P ĐA-I HQC H:¢ CHÍNH QUY

**Ngành: Công ngh� thông tin**

**HÀ NQI - 2019**

**ĐA-I HQC QUOC GIA HÀ NQI TRƯONG ĐA-I HQC CÔNG NGH:¢**

## Nguy@n Thj Vân Anh

**KET HQP SINH BAT BIEN VÒNG LA-P VÀ KY THU�T TH{JC THI TƯQNG TRƯNG TRONG KIEM CHUNG MQT SO TÍNH CHAT**

**CUA CHƯƠNG TRÌNH C/C++**

### KHÓA LU�N TOT NGHI:¢P ĐA-I HQC H:¢ CHÍNH QUY

**Ngành: Công ngh� thông tin**

**Cán b9 hưong d§.n: TS. Tô Văn Khánh**

**HÀ NQI - 2019**

**VIETNAM NATIONAL UNIVERSITY, HA NOI UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

## Nguyen Thi Van Anh

**APPLYING LOOP INVARIANT GENERATION AND SYMBOLIC EXECUTION IN C/C++ SOFTWARE VERIFICATION**

### BACHELOR’S THESIS

**Major: Information Technology**

**Supervisor: Dr. To Van Khanh**

### HANOI - 2019

**LOI CAM ƠN**

Lai đ&u tiên, em xin to lòng bi�t ơn chân thành và sâu sic t6i th&y giáo TS.Tô Văn Khánh - ngưai đã hư6ng dan ti;n tình, đ<ng viên và giúp đa em trong su6t quá trình thijc hi�n khóa lui;n t6t nghi�p này.

Em cũng xin gui t6i quý th&y cô Khoa Công ngh� thông tin nói riêng và trưang Đ1i h9c Công ngh� - Đ1i h9c Qu6c gia Hà N<i nói chung, đã ti;n tình truyen đ1t nhfing ki�n th11c quý báu làm nen tang vfing chic ho trQ em trên con đưang h9c ti;p và làm vi�c trong tương lai.

Con xin to lòng bi�t ơn vô h1n t6i b6 m� và nhfing ngưai thân trong gia đình. Nhfing ngưai đã nuôi con khôn l6n, trưdng thành và là ngu6n đ<ng viên, điJm tija vfing chic cua con trên con đưang h9c ti;p.

Cu6i cùng, em xin cam ơn nhfing t6i các anh chi và các b1n, đi;c bi�t là ti;p thJ l6p K60CLC đã đ6ng hành cùng em trong su6t 4 năm h9c qua. M9i ngưai đã chia se, giúp đa và đưa ra nhfing góp ý chân thành cho em trong h9c ti;p và cu<c s6ng.

### TÓM TAT

Sij phát triJn bùng n6 cua ngành công nghi�p ph&n mem trong nhfing năm g&n đây khi�n cho van đe đam bao chat lưQng ph&n mem ngày càng đưQc chú ý. Thijc thi tưQng trưng là m<t kĩ thui;t phân tích chương trình cho phép su dvng đ&u vào là các giá tri tưQng trưng, thay cho các giá tri cv thJ. Bat bi�n là công th11c đúng trư6c và trong moi l&n thijc thi cua vòng li;p, vì vi;y công th11c này có thJ đưQc áp dvng đJ ch11ng minh tính đúng đin m<t ph&n cua vòng li;p.

Trong nghiên c11u trư6c đó, công cv VTSE su dvng thijc thi tưQng trưng đJ kiJm ch11ng m<t s6 tính chat cua chương trình C/C++ đã đưQc cài đi;t. Tuy nhiên, v6i phương pháp su dvng là ga vòng li;p v6i s6 l&n nhat đinh, VTSE chi có thJ kiJm tra chương trình là đúng v6i s6 vòng li;p đã ga. Đieu này cũng khi�n VTSE có k�t qua chưa t6t trên các bài toán có s6 ga l&n li;p l6n hoi;c không có gi6i h1n vòng li;p. Khóa lui;n này xin trình bày hư6ng ti�p ci;n xu lý vòng li;p trong kiJm ch11ng b�ng cách k�t hQp bat bi�n vòng li;p vào kĩ thui;t thijc thi tưQng trưng.

Bat bi�n vòng li;p đưQc sinh ra su dvng b6 đe Farkas và m<t công cv con - InvaGen áp dvng phương pháp này cũng đưQc cài đi;t. Sau đó, InvaGen đưQc tích hQp vào công cv kiJm ch11ng cũ đJ xây dijng phiên ban VTSEinv. K�t qua thijc nghi�m cua VTSEinv so sánh v6i phiên ban VTSE trư6c đây cùng hai công cv CBMC và Veriabs là kha quan khi có s6 lưQng bài giai t6t và trong thai gian nhanh hơn rat nhieu, đi;c bi�t là các bài toán có gi6i h1n vòng li;p l6n.

***Tit khóa:*** *kiem chitng phdn mem, th'(tc thi tư(Jng trưng, bat bien vòng lrJp, phân tích mã ngubn*

### ABSTRACT

Recent years, software quality assurance has evolved and become increasingly impor- tant, especially with classified safety- or business-critical software. Symbolic Execution is a program analysis technique that allows the execution of a program using symbolic values instead of concrete ones. A loop invariant is a formula that is true immediately before and after each iteration of a loop, therefore can be used to prove the partial correctness of the loop.

In earlier work, a tool called VTSE using Symbolic Execution technique to verify some properties of C/C++ program was implemented. However, because of using the unfolding method when handling loops, VTSE can only ensure the program is correct with a fixed number of loops and also have disadvantages if a high number of loop unfolding is required. This thesis introduces an approach to handle loops in verification by combining Loop Invariant and Symbolic Execution technique.

Loop invariants are generated using the method which is Farkas’ Lemma and a tool InvaGen is implemented. The verification tool VTSEinv is improved by incorpo- rating with the use of the loop invariant by integrating with InvaGen. Experiments are performed on a set of benchmarks by using loop invariant. Results are compared with VTSE using the unfolding method and two other tools from SV-COMP are CBMC and VeriAbs. VTSEinv when using invariant can solve more tasks and produces re- sults in a much shorter time than other tools.

***Keywords:*** *software verification, symbolic execution, loop invariant, source code an- alyzing*

### LOI CAM ĐOAN

Tôi xin cam đoan r�ng nhfing nghiên c11u ve phương pháp k�t hQp bat bi�n vòng li;p và ky thui;t thijc thi tưQng trưng đJ ki�m ch11ng chương trình ph&n mem đưQc trình bày trong khóa lui;n này là cua tôi và chưa tlfng đưQc n<p như m<t báo cáo khóa lui;n t1i trưang Đ1i h9c Công Ngh� - Đ1i h9c qu6c gia Hà N<i hoi;c bat kỳ trưang đ1i h9c khác. Nhfing gì tôi vi�t ra không sao chép tlf các tài li�u, không su dvng các k�t qua cua ngưai khác mà không trích dan cv thJ. Tôi xin cam đoan công cv tôi trình bày trong khoá lui;n là do tôi tij phát triJn, không sao chép mã ngu6n cua ngưai khác. N�u sai tôi hoàn toàn chiu trách nhi�m theo quy đinh cua trưang Đ1i H9c Công Ngh�

- Đ1i H9c Qu6c Gia Hà N<i.

Hà N<i, ngày 22 tháng 05 năm 2019 Sinh viên

Nguyen Thi Vân Anh

# Möc löc

|  |  |
| --- | --- |
| **Danh sách chii' vi�t tiit**  **Danh sách bang Danh sách hình ve Chương 1 Md đfiu** | [**vii**](#_bookmark0)  [**vii**](#_bookmark0)[**vii**](#_bookmark1)[**1**](#_bookmark2) |
| 1.1 Đi;t van đe . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | [1](#_bookmark3) |
| 1.2 Ky thui;t thijc thi tưQng trưng . . . . . . . . . . . . . . . . . . . . . . . . | [3](#_bookmark6) |
| 1.3 Bat bi�n vòng li;p . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | [4](#_bookmark8) |
| 1.3.1 Logic Hoare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | [4](#_bookmark9) |
| 1.3.2 Bat bi�n vòng li;p . . . . . . . . . . . . . . . . . . . . . . . . . . . | [5](#_bookmark10) |
| 1.3.3 B6 đe Farkas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | [6](#_bookmark11) |
| 1.4 Đánh giá phương pháp ga vòng li;p . . . . . . . . . . . . . . . . . . . . . | [8](#_bookmark14) |
| 1.5 Đánh giá phương pháp xu lý vòng li;p b�ng bat bi�n . . . . . . . . . . . | [9](#_bookmark17) |
| **Chương 2 Phương pháp sinh b.it bi�n vòng l�p b�ng b6 đ� Farkas** | [**11**](#_bookmark20) |
| 2.1 Phương pháp sinh bat bi�n vòng li;p . . . . . . . . . . . . . . . . . . . . | [11](#_bookmark21) |
| 2.2 Xây dijng công cv sinh bat bi�n InvaGen . . . . . . . . . . . . . . . . . | [13](#_bookmark23) |
| 2.2.1 Ki�n trúc InvaGen . . . . . . . . . . . . . . . . . . . . . . . . . . | [13](#_bookmark24) |
| 2.2.2 H� th6ng chuyJn đ6i tr1ng thái . . . . . . . . . . . . . . . . . . . | [14](#_bookmark26) |

* + 1. Ma tri;n tr1ng thái Initiation và Consecution [14](#_bookmark27)
    2. Công th11c đưang thijc thi [16](#_bookmark29)
    3. Giai công th11c ràng bu<c và sinh bat bi�n [17](#_bookmark31)

Chương 3 K�t h<p b.it bi�n trong kiJm chung chương trình [19](#_bookmark34)

* 1. Công cv kiJm ch11ng chương trình VTSEinv [19](#_bookmark35)
     1. Cây cú pháp trlfu tưQng [20](#_bookmark37)
     2. Đ6 thi lu6ng đieu khiJn [21](#_bookmark40)
     3. K�t hQp bat bi�n vòng li;p vào VTSEinv [24](#_bookmark44)
     4. Đánh chi s6 và sinh biJu th11c ràng bu<c [26](#_bookmark47)
  2. Minh ho1 ho1t đ<ng cua VTSEinv [27](#_bookmark50)

Chương 4 Nghiên cuu liên quan [33](#_bookmark56)

* 1. M<t s6 công cv kiJm ch11ng ph&n mem [33](#_bookmark57)
     1. CBMC [33](#_bookmark58)
     2. VeriAbs [35](#_bookmark60)
  2. Phương pháp sinh bat bi�n vòng li;p [37](#_bookmark64)
     1. Sinh bat bi�n su dvng N<i suy Craig [37](#_bookmark65)
     2. Sinh và làm min bat bi�n b�ng phương pháp lay mau ch9n l9c . [38](#_bookmark68)

Chương 5 K�t qua th\fc nghi�m [40](#_bookmark70)

* 1. Đieu ki�n thijc nghi�m [40](#_bookmark71)
  2. K�t qua thijc nghi�m [41](#_bookmark73)

K�t lu�n [44](#_bookmark75)

**Danh sách chii' vi�t tiit**

**AST** Abstract Syntax Tree

**CFG** Control Flow Graph

**SMT** Satisfiability Modulo Theories

**SV-COMP** Software Verification Competition

**VTSE** Verification Tool based on Symbolic Execution

# Danh sách bang

* 1. K�t qua b< thijc nghi�m Loop-acceleration [43](#_bookmark74)

# Danh sách hình ve

* 1. Chương trình đ6i cho hai s6 và cây thijc thi tưQng trưng tương 11ng . . [4](#_bookmark7)
  2. Chương trình và h� th6ng chuyJn tr1ng thái tương 11ng [[14]](#_bookmark89) . . . . . . . [7](#_bookmark13)
  3. BiJu đ6 Venn cho bài toán Unfold v6i k�t qua **true** . . . . . . . . . . . [8](#_bookmark15)
  4. BiJu đ6 Venn cho bài toán Unfold v6i k�t qua **false** . . . . . . . . . . . [8](#_bookmark16)
  5. BiJu đ6 Venn cho bài toán Invariant v6i k�t qua **false** . . . . . . . . . . [9](#_bookmark18)
  6. BiJu đ6 Venn cho bài toán Invariant v6i k�t qua **true** [10](#_bookmark19)
  7. Ki�n trúc cua InvaGen [13](#_bookmark25)
  8. Ràng bu<c thu đưQc sau khi dùng REDLOG [18](#_bookmark33)
  9. Ki�n trúc cua VTSEinv [19](#_bookmark36)
  10. Cây cú pháp trlfu tưQng trong CDT [20](#_bookmark39)
  11. Cây cú pháp trlfu tưQng tlf mã ngu6n [27](#_bookmark51)
  12. CFG chèn bat bi�n vòng li;p [29](#_bookmark52)
  13. CFG phương pháp ga vòng li;p [30](#_bookmark53)
  14. K�t qua tlf Z3 (trên) và báo cáo Excel (dư6i) cua VTSEinv dùng Invariant [32](#_bookmark54)
  15. K�t qua tlf Z3 (trên) và báo cáo Excel (dư6i) cua VTSEinv dùng Unfold [32](#_bookmark55)
  16. Ki�n trúc cua CBMC [34](#_bookmark59)
  17. Ki�n trúc VeriAbs [36](#_bookmark62)

**Chương 1**

# Md đfiu

## Đ�t v.in đ�

Sij phát triJn bùng n6 cua ngành công nghi�p ph&n mem trong nhfing năm g&n đây khi�n cho van đe đam bao chat lưQng ph&n mem ngày càng đưQc chú ý. Trong đó, kiJm thu là m<t phương pháp hfiu hi�u đJ kiJm tra sij đúng đin cua chương trình, tuy nhiên, kiJm thu chi thi hành m<t ti;p nho đưang thijc thi v6i các đ&u vào là giá tri cv thJ. Đieu đó có thJ dan đ�n vi�c bo sót nhieu loi cua chương trình khi thijc hi�n kiJm thu. Vì vi;y, đJ ch11ng minh chương trình là không có loi, các phương pháp kiJm ch11ng ph&n mem đưQc 11ng dvng.

Trong kiJm ch11ng chương trình, vòng li;p là m<t trong nhfing van đe thách th11c nhat. Vi�c xác đinh s6 l&n li;p là không đơn gian và con s6 này còn phv thu<c vào dfi li�u đ&u vào. M<t giai pháp cho van đe này là su dvng bat bi�n vòng li;p. Bat bi�n là m<t công th11c logic thJ hi�n các tính chat đúng v6i m9i l&n thijc thi cua vòng li;p. Công th11c này cung cap thông tin cơ ban, thJ hi�n đưQc mvc đích và n<i dung chính cua vòng li;p là gì. M<t bat bi�n đu m1nh có thJ giúp tránh đưQc vi�c ga vòng li;p trong phương pháp kiJm ch11ng mô hình *(Model Checking)* [[4],](#_bookmark79) mà cv thJ là trong thijc thi tưQng trưng *(Symbolic Execution)* [[10].](#_bookmark85)

Chính vì sij quan tr9ng và khó khăn cua bài toán, đã có rat nhieu công trình ve tij đ<ng sinh bat bi�n vòng li;p tlf mã ngu6n chương trình đưQc thijc hi�n. Nhfing công

trình đ&u tiên bit đ&u tlf nhfing năm 70. Các phương pháp n6i bi;t hi�n nay trong sinh bat bi�n vòng li;p bao g6m dien giai trlfu tưQng *(Abstract Refinement)* [[3],](#_bookmark78) các kĩ thui;t giai ràng bu<c [[8][14],](#_bookmark89) phương pháp CEGAR [[9]](#_bookmark84) và phương pháp áp dvng công th11c h6i quy [[13].](#_bookmark88) Trong đó, giai ràng bu<c là hư6ng ti�p ci;n giúp sinh ra công th11c bat bi�n m<t cách nhanh chóng và hi�u qua. M<t trong nhfing đe xuat cua phương pháp này là áp dvng b6 đe Farkas đJ sinh bat bi�n vòng li;p.

Trong nghiên c11u trư6c đây, công cv VTSE su dvng thijc thi tưQng trưng đJ kiJm ch11ng m<t s6 tính chat cua chương trình C/C++ đã đưQc gi6i thi�u. Công cv cho k�t qua t6t trên b< thijc nghi�m cua cu<c thi SV-COMP[1](#_bookmark4) và đ6ng thai có kha năng thijc hi�n các bài toán có s6 dòng l�nh l6n đ�n cijc l6n. Tuy nhiên, v6i các bài toán có s6 l&n li;p l6n hoi;c không gi6i h1n l&n li;p, phương pháp ga v6i s6 l&n nhat đinh cua VTSE cho hi�u qua không cao. Vì vi;y, đJ giai quy�t van đe này, khóa lui;n xin đe xuat phương pháp k�t hQp bat bi�n vòng li;p và kĩ thui;t thijc thi tưQng trưng đJ kiJm ch11ng chương trình. Bat bi�n vòng li;p đưQc sinh tij đ<ng tlf mã ngu6n áp dvng b6 đe Farkas, sau đó k�t n6i bat bi�n v6i công cv VTSE trư6c đây đJ t1o nên phiên ban cai ti�n VTSEinv[2](#_bookmark5).

Các ph&n cua khóa lui;n đưQc cau trúc như sau. **Chương** [**1**](#_bookmark2)trình bày mvc tiêu cua khóa lui;n và các ki�n th11c lý thuy�t nen tang. **Chương** [**2**](#_bookmark20)trình bày phương pháp sinh bat bi�n và ki�n trúc công cv đưQc cài đi;t áp dvng phương pháp này là InvaGen. Ti�p theo, cách th11c k�t n6i công cv sinh bat bi�n vào kiJm ch11ng chương trình đưQc thJ hi�n d **Chương** [**3**](#_bookmark34). **Chương** [**4**](#_bookmark56)là các nghiên c11u liên quan đ�n m<t s6 phương pháp xu lý vòng li;p và sinh bat bi�n khác. **Chương** [**5**](#_bookmark70)đưa ra k�t qua thijc nghi�m cua VTSEinv so v6i phiên ban cũ và m<t s6 công cv khác. Cu6i cùng là **K�t lu�n** cua toàn b< khóa lui;n và công cv, cũng như trình bày các hư6ng phát triJn ti�p theo.

1https://sv-comp.sosy-lab.org/

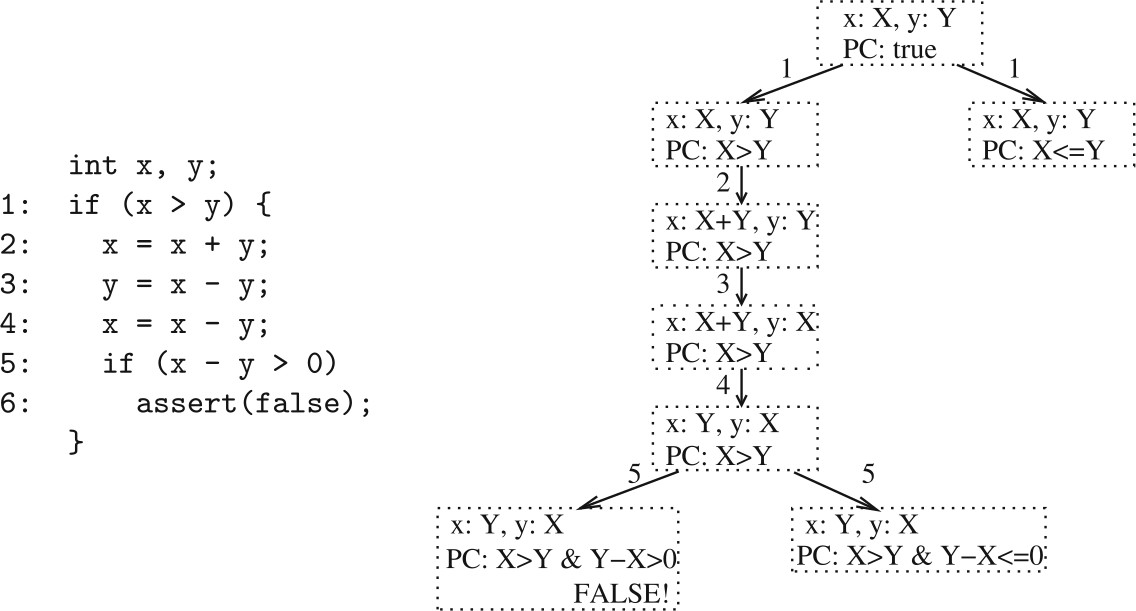
2https://github.com/vananhnt/Vtse-VerificationTool/tree/invariant-new

## Ky thu�t th\fc thi tư<ng trưng

Thijc thi tưQng trưng (*Symbolic Execution*) [[10]](#_bookmark85) là m<t kĩ thui;t phân tích chương trình đưQc đe xuat tlf nhfing năm 70, tuy nhiên đang ngày càng đưQc quan tâm do sij ph6 bi�n cua máy tính và sij phát triJn cua các thui;t toán m6i. Trong kĩ thui;t này, thay vì đ&u vào là các giá tri cv thJ, chương trình đưQc thijc thi v6i các giá tri là tưQng trưng. Vì vi;y, đ&u ra sinh bdi chương trình se đưQc thJ hi�n dư6i d1ng m<t hàm s6 cua các đ&u vào tưQng trưng. Ong dvng cua thijc thi tưQng trưng có thJ thay tlf sinh kich ban kiJm thu tij đ<ng đ�n ch11ng minh tính đúng đin cua chương trình.

Tr1ng thái cua chương trình khi đưQc thijc thi tưQng trưng bao g6m các giá tri tưQng trưng cua bi�n, m<t đưang đieu ki�n *(path condition - PC)*, và m<t b< đ�m. Đưang đieu ki�n là m<t biJu th11c đ1i s6 không ch11a lưQng tlf dijng trên các giá tri đ&u vào tưQng trưng. Tlf moi đưang đieu ki�n ta thu đưQc các ràng bu<c mà đ&u vào c&n thoa mãn đJ chương trình có thJ thijc thi theo đưang đó. M<t cây thijc thi tưQng trưng *(symbolic execution tree)* thJ hi�n các tính chat cua đưang thijc thi. Các nút cua cây đ1i di�n các tr1ng thái cua chương trình và đưQc k�t n6i bdi các chuyJn tr1ng thái *(transition)* cua h� th6ng.

Xem xét đo1n mã ngu6n trong Hình [1.1](#_bookmark7) (trái), chương trình đ6i cho giá tri cua hai bi�n s6 nguyên x và y, khi x l6n hơn y [[16].](#_bookmark91) Hình [1.1](#_bookmark7) (phai) biJu dien cây thijc thi tương 11ng. Khdi t1o, PC là *true* và x, y có giá tri tưQng trưng l&n lưQt là X, Y. T1i moi điJm chia nhánh, PC đưQc ci;p nhi;t theo các nhánh. Sau khi thijc thi *if* t1i 1, cây thijc thi chia thành hai nhánh *if* và *else*, và PC đưQc ci;p nhi;t tương 11ng. V6i moi câu l�nh, tr1ng thái m6i cua bi�n x, y cũng đưQc ci;p nhi;t. N�u m<t *path condition* là *false*, nghĩa là không có ti;p đ&u vào nào thoa mãn, tr1ng thái đay không bao gia xay ra *(unreachable)* thì thijc thi trên *path condition* đay se không đưQc ti�p tvc. Theo như ví dv, t1i câu l�nh *if* s6 5 thì PC cua nhánh *then* là không thoa mãn, vì vi;y câu l�nh 6 se không bao gia đưQc thijc thi vào.



Hình 1.1: Chương trình đ6i cho hai s6 và cây thijc thi tưQng trưng tương 11ng

## B.it bi�n vòng l�p

#### Logic Hoare

**Logic Hoare** (Logic Floyd-Hoare) là m<t h� chính quy cung cap m<t ti;p các quy tic suy lui;n tính đúng đin cua chương trình. Cài đi;t cua m<t chương trình là đúng m<t ph&n *(partially correct)* so v6i đi;c ta n�u như gia đinh tien đieu ki�n là đúng, thì n�u chương trình k�t thúc, hi;u đieu ki�n đưQc thoa mãn. M<t chương trình là đúng hoàn toàn *(totally correct)* khi gia đinh tien đieu ki�n là đúng, chương trình chic chin se cham d11t *(terminate)* và khi nó cham d11t, thì hi;u đieu ki�n thoa mãn. Đieu đó tương đương, chương trình đúng hoàn toàn khi thoa mãn lui;t đúng m<t ph&n và chic chin cham d11t.

M<t b< ba Hoare đưQc biJu dien d d1ng *{P}S{Q}*, là đúng n�u ta bit đ&u t1i tr1ng thái *P* là đúng và thijc thi *S*, thì *S* se k�t thúc t1i tr1ng thái mà *Q* đúng. Theo

lui;n lý Hoare, tính đúng đin m<t ph&n cua lui;t **While** đưQc đinh nghĩa như dư6i đây:

*{Inv ∧ C}* Body *{Inv}*

*{Inv}* **while** C **do** Body **done** *{¬C ∧ Inv}*

trong đó *Inv* là bat bi�n vòng li;p, *C* là biJu th11c đieu ki�n và *Body* là các câu l�nh trong thân vòng li;p.

Vì vi;y, c&n tìm ra bat bi�n đJ có thJ ch11ng minh tính đúng m<t ph&n cua vòng li;p. Tuy nhiên, vi�c tìm ra bat bi�n đu m1nh và phù hQp là m<t thách th11c không de dàng. Đ6ng thai, khóa lui;n này se chi đe ci;p đ�n ch11ng minh vòng li;p là đúng m<t ph&n và mi;c đinh chương trình se cham d11t t1i m<t thai điJm nhat đinh.

#### B.it bi�n vòng l�p

Vòng li;p trong m<t chương trình có thJ đưQc thJ hi�n v6i nhieu cau trúc khác nhau. D1ng *while* su dvng m<t đieu ki�n liên tvc và k�t thúc khi đieu ki�n này sai, d1ng *do-until* luôn luôn thijc thi vòng li;p ít nhat m<t l&n, kiJm tra đieu ki�n d cu6i thay vì ban đ&u, *for* duy�t qua m<t dãy s6 hoi;c m<t cau trúc dfi li�u. Tat ca các d1ng đeu là bi�n thJ cua vòng li;p và có thJ de dàng chuyJn đ6i mà không thay đ6i ngfi nghĩa. Moi vòng li;p trong chương trình đeu có ch11a các thành ph&n như dư6i đây.

1 P; //pre-condition

2 Init;

3 While (Cond) {

4 //invariant Inv

5 Body;

6 }

*Inv* là bat bi�n cua vòng li;p n�u thoa mãn các đieu ki�n [[7]:](#_bookmark82)

* + - 1. V6i moi l&n *Init* thijc hi�n, nghĩa là t1i tr1ng thái trư6c vòng li;p, se dan đ�n tr1ng thái mà *Inv* thoa mãn.
      2. V6i moi l&n *Body* đưQc thijc thi, bit đ&u t1i tr1ng thái *Inv* và *Cond* thoa

mãn, se dan đ�n m<t tr1ng thái *Inv* ti�p tvc thoa mãn.

N�u có các tính chat trên, moi khi vòng li;p cham d11t se dan đ�n tr1ng thái *Inv* và *¬Cond.* Moi vòng li;p đưQc đinh nghĩa là m<t thijc thi cua *Init* và sau đó là không hoi;c nhieu hơn thijc thi cua *Body*, đưQc li;p khi *Cond* van thoa mãn. Vì vi;y, n�u *Init* thoa mãn bat bi�n và m<t l&n nào đó *Body* thoa mãn, thì *Init* theo sau bdi bao nhiêu l&n thijc hi�n cua *Body* cũng vi;y. BiJu dien b�ng Logic Hoare, ta có bat bi�n *Inv* là công th11c thoa mãn:

* *{P}Init{Inv}*, đưQc g9i là bư6c khdi đ&u *(initiation)*
* *{Iniv ∧ Cond}Body{Inv}*, đưQc g9i là bư6c quy h6i *(consecution; inductiveness)*.

Bat bi�n *Inv* không có vai trò ngfi nghĩa trijc ti�p nhưng có thJ giúp ta hiJu rõ hơn ve vòng li;p và tính đúng đin cua vòng li;p.

#### B6 đ� Farkas

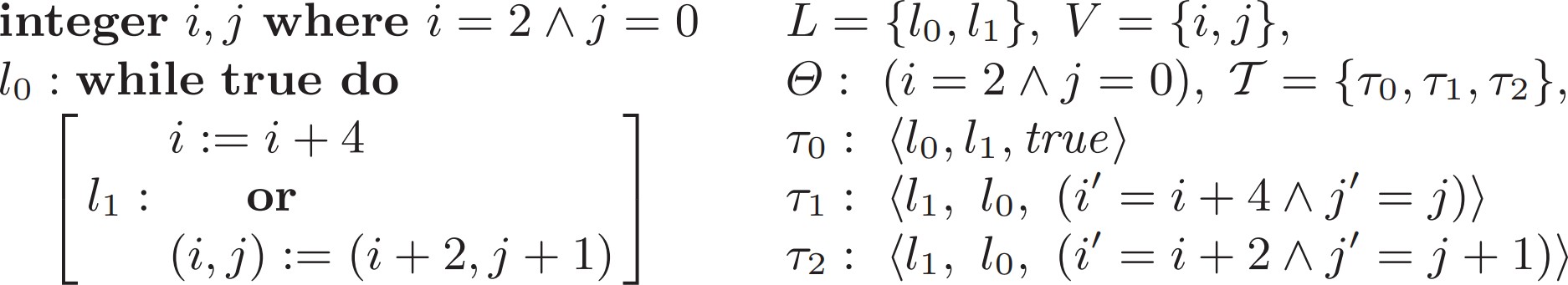
Sau nghiên c11u cua Colón và các c<ng sij [[14](#_bookmark89)] hư6ng ti�p ci;n sinh bat bi�n vòng li;p trên giai ràng bu<c đã đưQc phát triJn m1nh me. N6i bi;t trong s6 đó là phương pháp áp dvng b6 đe Farkas. V6i đ&u vào là m<t h� th6ng chuyJn đ6i tr1ng thái, b6 đe Farkas su dvng kĩ thui;t tính toán trên ràng bu<c đJ sinh ra các bat bi�n vòng li;p d d1ng biJu th11c tuy�n tính.

##### H� th6ng chuyJn tr9-ng thái

M<t h� th6ng chuyJn tr1ng thái P <V, L, *l*0, *θ*, T> bao g6m m<t ti;p hQp các bi�n V, vi trí đ&u *l*0, ti;p hQp vi trí L, đieu ki�n kiJm tra *θ* và m<t ti;p các chuyJn đ6i T.

##### B6 đ�

Xem xét h� th6ng S g6m các bat đ�ng th11c tuy�n tính trên ti;p bi�n *x*1, *x*2, ..., *xn*:



Hình 1.2: Chương trình và h� th6ng chuyJn tr1ng thái tương 11ng [[14]](#_bookmark89)

 *a*11*x*1 + *...*+ *a*1*nxn*+ *b*1 *≤* 0 

S:  .

. . 

*am*1*x*1 + *...*+ *amnxn*+ *bm ≤* 0

Khi S là thoa mãn, tlf S có thJ suy ra đưQc bat đ�ng th11c

*ψ* : *c*1*x*1 + *...* + *amnxn* + *bm ≤* 0 (1.1) khi và chi khi t6n t1i các s6 thijc không âm *λ*0,*λ*1,...*λm* thoa mãn

*m m m*

*c*1 = *λiai*1*, ... cn* = *λiain, d* = ( *bi*) *− λ*0 (1.2)

*i*=1

*i*=1

*i*=1

S là không thoa mãn khi và chi khi tlf S suy ra bat đ�ng th11c 1*≤*0. ĐJ 11ng dvng, b6 đe Farkas có thJ đưQc thJ hi�n dư6i d1ng bang như sau:

*λ*0

*λ*1

.

*λm*

*−*1 *≤* 0

.

*am*1*x*1 + *...* + *amnxn* + *bm ≤* 0

*a*11*x*1 + *...* + *a*1*nxn* + *b*1 *≤* 0 



S

*c*1*x*1 + *...* + *cnxb* + *d ≤* 0 *← ψ*

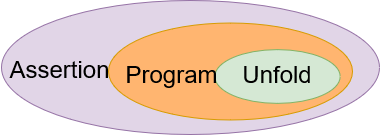
1 *≤* 0 *←* disable

Tlf đó, ta có thJ tìm ra bat bi�n vòng li;p dư6i d1ng m<t biJu th11c có d1ng:

*c*1*x*1 + *c*2*x*2 + *...* + *d ≤* 0 (1.3)

## Đánh giá phương pháp ga vòng l�p

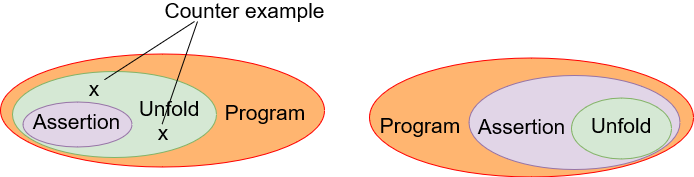
Gia su mien giá tri thoa mãn chương trình, thoa mãn biJu th11c đieu ki�n ngưai dùng và biJu th11c khi ga vòng li;p v6i s6 l&n nhat đinh đưQc thJ hi�n qua biJu đ6 Venn dư6i đây.



Hình 1.3: BiJu đ6 Venn cho bài toán Unfold v6i k�t qua **true**

*Program - chương trình, Assertion - đieu ki(!n ngưoi dùng, Unfold - grJ vòng lrJp.*

Trong trưang hQp bài toán cho k�t qua đúng **true** (Hình [1.6)](#_bookmark19), có nghĩa là đieu ki�n ngưai dùng luôn đưQc chương trình thoa mãn, thì mien giá tri cua Program se luôn n�m trong mien Assertion. Ta có Unfold là m<t xap xi dư6i cua Program, vi;y nên trong trưang hQp này ta luôn có thJ ch11ng minh chương trình là **true** khi dùng phương pháp Unfold.



Hình 1.4: BiJu đ6 Venn cho bài toán Unfold v6i k�t qua **false**

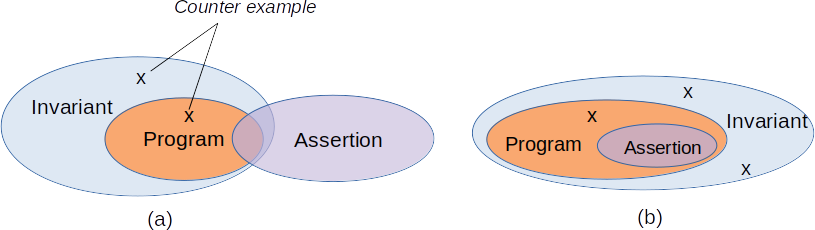
*Program - chương trình, Assertion - đieu ki(!n ngưoi dùng, Unfold - grJ vòng lrJp, Counter Example - phiin ví dV.*

Trong trưang hQp bài toán cho k�t qua đúng **false** (Hình [1.4),](#_bookmark16) nghĩa là chương trình có ch11a giá tri không thoa mãn đieu ki�n ngưai dùng, hay còn đưQc g9i là phan ví dv. Công th11c Unfold là xap xi dư6i cua Program, vì vi;y, v6i trưang hQp (trái) k�t qua **false** và phan ví dv se đưQc xác đinh đúng. Tuy nhiên v6i trưang hQp (phai) thì

se bi báo k�t qua sai *(unsound)*. Chương trình là **false** nhưng khi su dvng công th11c Unfold se đưa ra k�t qua là **true**. Đây là loi bo sót *bug* và rat nghiêm tr9ng trong kiJm ch11ng ph&n mem.

## Đánh giá phương pháp xu' lý vòng l�p b�ng b.it bi�n

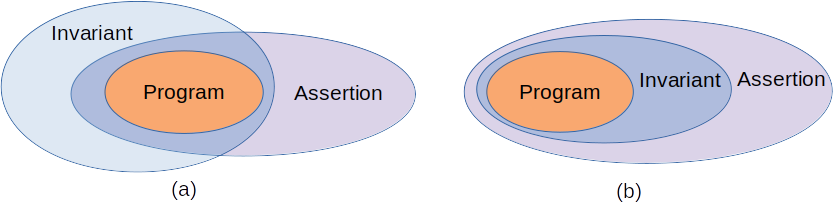
Gia su mien giá tri thoa mãn chương trình, thoa mãn biJu th11c đieu ki�n ngưai dùng và thoa mãn bat bi�n vòng li;p l&n lưQt đưQc thJ hi�n qua biJu đ6 Venn.



Hình 1.5: BiJu đ6 Venn cho bài toán Invariant v6i k�t qua **false**

*Program - chương trình, Assertion - đieu ki(!n ngưoi dùng, Invariant - b**at bien, Counter Example - phiin ví dV.*

Trong trưang hQp bài toán có k�t qua đúng là **false** (Hình [1.5).](#_bookmark18) Khi này se có nhfing phan ví dv là giá tri cua Program n�m ngoài Assertion. Do bat bi�n vòng li;p là m<t xap xi trên cua công th11c chương trình, vì vi;y ta luôn có thJ su dvng bat bi�n vòng li;p đJ ch11ng minh chương trình là **false**. Tuy nhiên, khi này phan ví dv đưQc đưa ra se không thJ chic chin có n�m trong mien giá tri cua Program hay không. Vì vi;y, khi mien giá tri cua Invariant càng sát v6i Program thì kha năng phan ví dv đưa ra là giá tri cua chương trình càng tăng.



Hình 1.6: BiJu đ6 Venn cho bài toán Invariant v6i k�t qua **true**

*Program - chương trình, Assertion - đieu ki(!n ngưoi dùng, Invariant - bat* *bien.*

Trong trưang hQp bài toán cho k�t qua đúng **true** (Hình [1.6),](#_bookmark19) ta có Invariant là m<t xap xi trên cua Program, vi;y nên v6i chương trình **true** thì chưa chic su dvng bat bi�n se cho k�t qua đúng. Khi mien giá tri cua Invariant là quá r<ng (Hình 4.2 (a)) thì k�t qua tra ve se là sai. Vì vi;y, vi�c thu h�p bat bi�n sát v6i chương trình se có thJ cho k�t qua chính xác hơn. Xét mã ngu6n C dư6i đây, v6i m<t công th11c bat bi�n y�u thì biJu th11c đieu ki�n ngưai dùng se bi vi ph1m, chương trình chi có thJ đưQc kiJm ch11ng **true** n�u m<t bat bi�n đu m1nh đưQc sinh ra.

1 int sumN(int n) {//pre-condition n >= 0;

2 int i = 0; s = 0;

3 while (i <= n) {

4 //strong invariant ((s == i\*(i-1)/2) && (i <= n+1))

5 //weak invariant ((s >= 0) && (i <= n + 1))

6 s = s + i;

7 i = i + 1;}

8 return s;

9 VERIFIER\_assert(return == n\*(n+1)/2);

10 //assertion user want to verify}

Có thJ nói, vi�c tìm đưQc bat bi�n phù hQp và đu m1nh đJ kiJm ch11ng chương trình là m<t bài toán không xác đinh *(undecidable)*. Cho đ�n nay, các phương pháp đã và đang đưQc nghiên c11u đeu nhim t6i vi�c sinh ra đưQc bat bi�n đu m1nh cho nhieu bài toán nhat có thJ.

**Chương 2**

# Phương pháp sinh b.it bi�n vòng l�p b�ng b6 đ� Farkas

Trong chương này, khóa lui;n xin trình bày các bư6c sinh bat bi�n vòng li;p b�ng b6 đe Farkas d [2.1](#_bookmark21) . Đ6ng thai, công cv InvaGen áp dvng phương pháp này đã đưQc xây dijng. Ph&n [2.2](#_bookmark23) se thJ hi�n ki�n trúc và cách th11c cài đi;t công cv này.

## Phương pháp sinh b.it bi�n vòng l�p

Phương pháp sinh bat bi�n vòng li;p tuy�n tính đã đưQc gi6i thi�u t1i [[14].](#_bookmark89) Đ&u vào cua phương pháp là m<t h� th6ng chuyJn đ6i tr1ng thái như đã đưQc trình bày d Hình

[1.2.](#_bookmark13) Ban đ&u, coi các bat bi�n c&n tìm là công th11c có d1ng

*c*1*x*1 + *c*2*x*2 + *..* + *d ≤* 0

mà trong đó, *x*1*, x*2*, ...* là các bi�n xuat hi�n trong vòng li;p, *c*1*, c*2*, ..., d* là các h� s6 c&n tìm. Hai đieu ki�n đJ tìm ra bat bi�n quy n1p đưQc thJ hi�n dư6i d1ng các bang tính **Initiation** và **Consecution**. Cv thJ các bang tính đưQc xây dijng như sau.

**Initiation:** Ti;p câu l�nh khdi đ&u *θ*, nghĩa là các câu l�nh đ11ng trư6c vòng li;p se đưQc đưa vào bang tính.

*λ*0

*λ*1

.

*λm*

*−*1 *≤* 0

.

*am*1*x*1 + *...* + *amnxn* + *bm ≤* 0

*a*11*x*1 + *...* + *a*1*nxn* + *b*1 *≤* 0 



*θ*

*c*1*x*1 + *...* + *cnxb* + *d ≤* 0 *← ϕ*

1 *≤* 0 *←* disable

Trong đó *λ*0*, λ*1*, ...λm* là các s6 thijc không âm, *ϕ* là công th11c bat bi�n thoa mãn trư6c khi bư6c vào vòng li;p.

**Consecution:** V6i moi bang Consecution thì ti;p câu l�nh đưQc thijc hi�n trong tlfng bi�n đ6i tr1ng thái se đưQc đưa vào bang tính[1](#_bookmark22).

*µ*

*λ*0 *λ*1

.

*λm*

*c*1*x*1 + *...* + *cnxn*

*a*11*x*1 + *...* + *a*1*nxn*

.

*am*1*x*1 + *...* + *amnxn*

+ *al xl* + *...* + *al*

11 1

1*n*

*xl*

*n*

.

+ *al xl* + *...* + *al*

*m*1 1

*mn n*

*xl*

+ *di ≤* 0

*−*1 *≤* 0

+*b*1 *≤* 0

.

+*bm ≤* 0

 T

*c*1*xl* + *...* + *cnxl*



1

*b*

+*d ≤* 0 *← ϕl*

1 *≤* 0 *←* disable

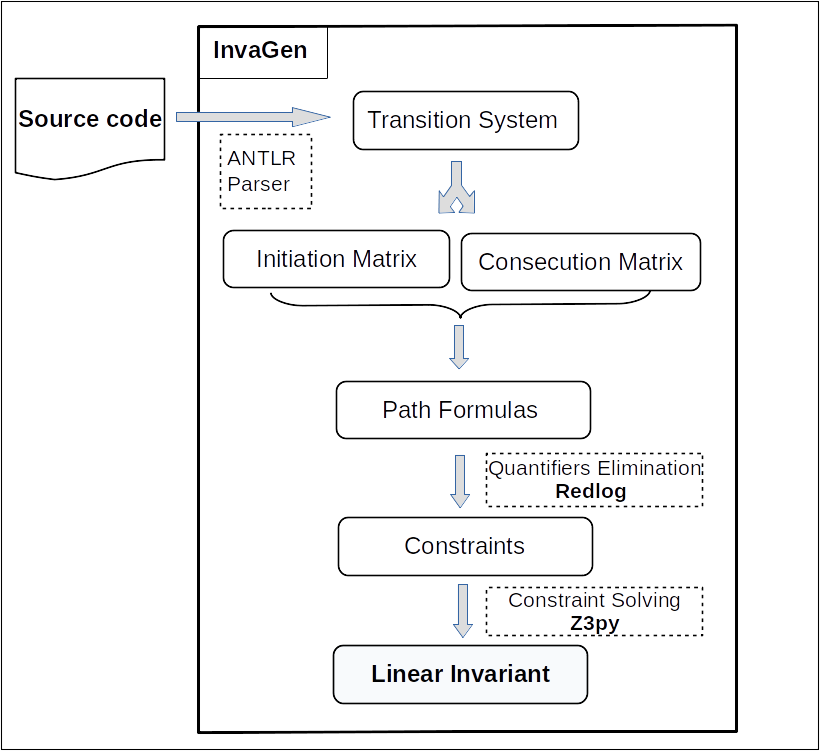
Trong đó *µ, λ*0*, λ*1*, ...λm* là các s6 thijc không âm. *ϕl* là công th11c bat bi�n thoa mãn đieu ki�n trong vòng li;p.

Tlf hai bang tính trên, ta thu đưQc công th11c ràng bu<c gifia các h� s6 *c*1*, c*2*, ..., d*. V6i moi b< nghi�m h� s6 thoa mãn công th11c ràng bu<c tìm đưQc, tlfng công th11c bat bi�n tương 11ng đưQc t1o nên. Khi này, bài toán chuyJn ve thu&n tính toán s6 h9c. Vi�c quan lý các bang tính và h� s6 tương 11ng có thJ chuyJn ve bài toán xu lý ma tri;n và giai bat đ�ng th11c. Chi ti�t phương pháp và thui;t toán đJ giai ràng bu<c đưQc trình bày d mvc sau.

1a, b ký hi�u cho h� s6 đã biet, c, d là các giá tri chưa biet

## Xây d\fng công cv sinh b.it bi�n InvaGen

#### Ki�n trúc InvaGen



Hình 2.1: Ki�n trúc cua InvaGen

Ph&n này xin trình bày ve ki�n trúc cua InvaGen - công cv sinh bat bi�n vòng li;p su dvng b6 đe Farkas. Ki�n trúc này dija trên phương pháp sinh bat bi�n tuy�n tính b�ng vi�c giai công th11c ràng bu<c tuy�n tính đã đưQc phát triJn bdi Colon và các c<ng sij [[14].](#_bookmark89) V6i đ&u vào là mã ngu6n C/C++ ch11a vòng li;p tuân thu các *template* đã đưQc đinh nghĩa, thì đ&u ra cua InvaGen là các biJu th11c bat bi�n d d1ng tuy�n tính *c*1*x*1 + *c*2*x*2 + *...* + *d ≤* 0. Ban đ&u, mã ngu6n đưQc chuyJn ve d1ng H� th6ng chuyJn tr1ng thái, ti�p theo ma tri;n Initiation và Consecution se đưQc dijng lên tlf các chuyJn đ6i tr1ng thái trên. B�ng vi�c tính toán trên các ma tri;n, công th11c đ1i di�n cho tlfng đưang thijc thi cua chương trình đưQc hình thành. Sau khi lo1i bo các

lưQng tlf, công th11c ràng bu<c ve *c*1*, c*2*, ..., d* đưQc tìm ra và đưa vào b< giai Z3py đJ sinh ra b< nghi�m thoa mãn, tlf đó hình thành các bat bi�n cua vòng li;p.

#### H� th6ng chuyJn đ6i tr9-ng thái

Mã ngu6n cua chương trình se đưQc thJ hi�n dư6i d1ng H� th6ng chuyJn đ6i tr1ng thái (*Transition System*) đã đưQc trình bày t1i [1.3.3.](#_bookmark12) H� th6ng chuyJn đ6i tr1ng thái là đ<c li;p v6i ngôn ngfi li;p trình và đưQc phân tích su dvng b< phân tích cú pháp ANTLR4 [2](#_bookmark28). Sau khi phân tích, ta thu đưQc ti;p các biJu th11c thJ hi�n tr1ng thái ban đ&u *θ* và ti;p các chuyJn tr1ng thái ti�p theo T.

Mã ngu6n 2.1: Ví dv file xml biJu dien Transition System

1 <TransitionSystem>

2 <Variables>x, y</Variables>

3 <Initiation> x = 2 and y = 0 </Initiation>

4 <Consecutions>

5 <Transition> x’ = x + 4; y’ = y </Transition>

6 <Transition> x’ = x + 2; y’ = y + 1 </Transition>

7 </Consecutions>

8 </TransitionSystem>

#### Ma tr�n tr9-ng thái Initiation và Consecution

Ta dijng lên ma tri;n tr1ng thái Initiation cho các tr1ng thái ban đ&u *θ* và ma tri;n Consecution cho các chuyJn tr1ng thái ti�p theo thu<c ti;p T. Trong bài toán này, m<t bang tính Initiation và hai bang tính Consecution đưQc xây dijng. Vì vi;y, s6 ma tri;n tương 11ng se đưQc dùng đJ quan lý các h� s6 trong bang tính trên.

**Ví dv**, v6i đ&u vào là h� th6ng đưQc thJ hi�n d Mã ngu6n 3.1, bang tính se có giá tri như sau, trong đó *λ*0*, λ*1*, λ*2 *≥* 0.

2h[ttps://www.antlr.org/](http://www.antlr.org/)

*λ*0

*λ*1 *x λ*2

*−*1 *≤* 0

*−* 2 = 0

*y* = 0

*c*1*x* + *c*2*y* + *d ≤* 0 *← ϕ*

Khi đó, ma tri;n h� s6 dư6i đây cua bang tính đưQc lưu vào h� th6ng.

 0 0 *−λ*0 

*λ*1 0 *−*2*λ*1

 

0 *λ*2 0

Do h� th6ng có hai chuyJn tr1ng thái, vì vi;y hai bang tính Consecution l&n lưQt đưQc dijng lên, v6i *µ, λ*1*, λ*2 *≥* 0 . Bang tính và ma tri;n cho *τ*1 : (*xl* = *x* + 4; *yl* = *y*) là

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *µ* | *c*1*x* + | *c*2*y* |  | + *d ≤* 0 |
| *λ*0  *λ*1 *λ*2 | *x* | *y* | *−xl*  *−yl* | *−*1 *≤* 0  +4 = 0  = 0 |
|  | *c*1*xl* + *c*2*yl* +*d ≤* 0 *← ϕl*  1 *≤* 0 *←* disable | | | |

*µc*1 *µc*2 0 0 *µd* 

 0 0 0 0 *−λ*0

*λ*1 0 *−λ*1 0 4*λ*1

 

0 *λ*2 0 *−λ*2 0

Bang tính và ma tri;n cho *τ*2 : (*xl* = *x* + 2; *yl* = *y* + 1) đưQc biJu dien tương tij.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *µ* | *c*1*x* + | *c*2*y* |  | + *d ≤* 0 |
| *λ*0  *λ*1 *λ*2 | *x* | *y* | *−xl*  *−yl* | *−*1 *≤* 0  +2 = 0  +1 = 0 |
|  | *c*1*xl* + *c*2*yl* +*d ≤* 0 *← ϕl*  1 *≤* 0 *←* disable | | | |

V6i bang tính trên, ta có ma tri;n h� s6 dư6i đây.

*µc*1 *µc*2 0 0 *µd* 

 0 0 0 0 *−λ*0

*λ*1 0 *−λ*1 0 2*λ*1

 

0 *λ*2 0 *−λ*2 *λ*2

#### Công thuc đưong th\fc thi

Như đã thJ hi�n d trên, moi bang tính se có hai công th11c đưang thijc thi *(path formula)* đưQc t1o ra, m<t đưang đ1i di�n trưang hQp thoa mãn - *ϕ, ϕl*, m<t đ1i di�n trưang hQp không thoa mãn - *disable*. Áp dvng b6 đe Farkas, trong tlfng *path formula*, ta xác đinh đưQc m6i quan h� gifia *c*1*, c*2*, ..., d* dija trên *µ, λ*0*, λ*1*, ...λm*. Công th11c cu6i cùng là hQp cua tat ca các *path formula* có thJ cua chương trình, nghĩa là đưQc sinh ra tlf ca ma tri;n Initiation và các ma tri;n Consecution.

V6i bài toán d trên, ta có công th11c t1i Initiation là

(*∃λ*1*, λ*2)[*c*1 = *λ*1 *∧ c*2 = *λ*2 *∧ d* = *−*2*λ*1] (2.1) V6i Consecution cho *τ*1, công th11c là

 *µ.c*1 + *λ*1 = 0*∧*   *µ.c*1 + *λ*1 = 0*∧* 

*∃*(*µ, λ* ) 

*i*

*µ.c*2 + *λ*2 = 0*∧*

 *∨ ∃*(*µ, λ* ) 

*µ.c*2 + *λ*2 = 0*∧* 

(2.2)

*c*1

= *−λ*1

*∧ c*2

= *λ*2*∧*

*λ*1

= 0 *∧ λ*2

= 0*∧*

 (*µ −* 1)*d* = *λ*0 *−* 4*λ*1   *µ.d − λ*0 = 1 

*i*

Tương tij, ma tri;n cho *τ*2 : (*xl* = *x* + 2; *yl* = *y* + 1) đưQc biJu dien và tlf đó thu đưQc công th11c

 *µ.c*1 + *λ*1 = 0*∧*   *µ.c*1 + *λ*1 = 0*∧* 

*∃*(*µ, λ* ) 

*i*

*µ.c*2 + *λ*2 = 0*∧*

 *∨ ∃*(*µ, λ* ) 

*µ.c*2 + *λ*2 = 0*∧* 

(2.3)

 *c*1

= *−λ*1

*∧ c*2

= *−λ*2*∧* 

 *λ*1

= 0 *∧ λ*2

= 0*∧* 

*d* = *µ.d* + 2*λ*1 + *λ*2 *− λ*0 *µ.d* + 2*λ*1 + *λ*2 *− λ*0 = 1

*i*

Vì vi;y, v6i h� th6ng chuyJn tr1ng thái đưQc ví dv, ta thu đưQc ba *path formula* là các công th11c (2.1), (2.2), (2.3). Thui;t toán [1](#_bookmark30) sinh ra đưang thijc thi cu6i cùng đưQc

trình bày dư6i đây. Trong đó, công th11c đưQc dijng lên tlf tlfng bang tính *Initiation* và *Consecution*, sau đó đưQc n6i thành công th11c cu6i cùng b�ng các phép toán logic h<i và tuyJn tương 11ng.

**Thu�t toán 1:** Thui;t toán sinh công th11c đưang thijc thi

**Input :** Transition System ts

**Output:** String pathformula

**1** init *←* ts.getInitiation();

**2** initformula *←* createFormula(init);

**3** pathformula *←* initformula;

**4 for** *X ∈ ts.getConsecutions()* **do**

**5** /\* join path formula for consecutions \*/

**6** consformula *←* Or(createFormula(X.enable), createFormula(X.disable);

**7** pathformula *←* And(pathformula, consformula);

**8 end**

**Result:** pathformula

**9** /\* create formula from initiation and consecution matrix \*/

**10 Function** *createFormula(String[][] matrix) : String*

**11** formula *←* empty;

**12 for** *j ∈ matrix.getColumn()* **do**

**13 for** *i ∈ matrix.getRow()* **do**

**14** formula += sumByColumn(matrix[i],[j]);

**15 end**

**16 end**

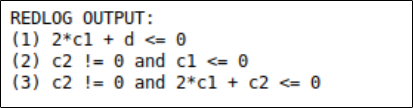
**17 end**

#### Giai công thuc ràng bu9c và sinh b.it bi�n

Công th11c ràng bu<c gifia *c*1*, c*2*, ...d* là m<t biJu th11c logic thu đưQc khi rút g9n và lo1i bo lưQng tlf tlf các công th11c đưang thijc thi. B< lo1i bo lưQng tlf đưQc su dvng là công cv REDLOG[3](#_bookmark32). Sau khi qua REDLOG, ta thu đưQc l&n lưQt các công th11c ràng

3[http://redlog.com](http://redlog.com/)

bu<c tương 11ng v6i tlfng *path formula*. V6i ví dv trên, ta có công th11c ràng bu<c cu6i cùng là h<i cua các công th11c (1), (2) và (3) như dư6i đây.



Hình 2.2: Ràng bu<c thu đưQc sau khi dùng REDLOG

Sau đó, công th11c ràng bu<c đưQc đưa vào b< giai Z3py đJ sinh ra các b< *c*1*, c*2*, ..., d* thoa mãn, đ6ng thai ta có các bat bi�n vòng li;p tương 11ng. Ý tưdng đJ giai công th11c ràng bu<c là tìm ki�m giá tri thoa mãn trên tlfng không gian cua b< nghi�m. V6i công th11c ràng bu<c g6m ba bi�n *c*1*, c*2*, d*, tlfng không gian nghi�m đưQc kiJm tra. Các không gian có thJ là *c*1 *<* 0*, c*2 *<* 0*, d <* 0 hoi;c *c*1 *≥* 0*, c*2 *<* 0*, d <* 0 hoi;c *c*1 *≥* 0*, c*2 *≥* 0*, d <* 0 ... Vì vi;y v6i ba bi�n, s6 không gian c&n kiJm tra là 23 = 8 . Trong ví dv trên, sau khi giai ràng bu<c su dvng ý tưdng trên b�ng Z3py, k�t qua đưa ra đưQc ba b< nghi�m thoa mãn.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *c*1 | *c*2 | *d* | *c*1*x* + *c*2*y* + *d* | *≤* 0 |
| 0 | -1 | 0 | *−y* | *≤* 0 |
| -1 | -1 | 2 | *−x − y* + 2 | *≤* 0 |
| -1 | 2 | 0 | *−x* + 2*y* | *≤* 0 |

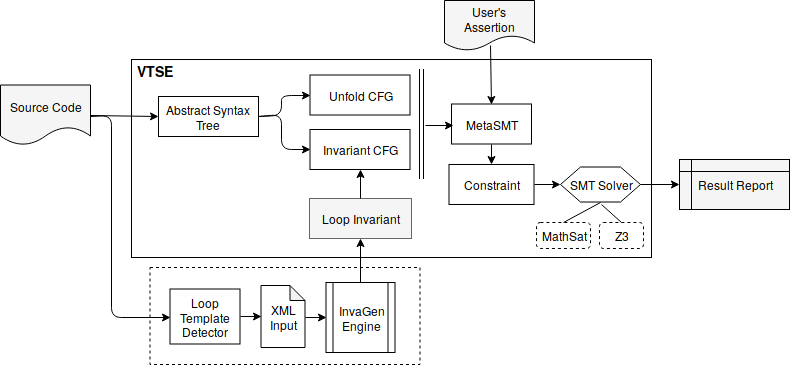
Cu6i cùng, ta thu đưQc công th11c bat bi�n vòng li;p

*ϕ* : (*y ≥* 0) *∧* (*x* + *y −* 2 *≥* 0) *∧* (*x −* 2*y ≥* 0)

**Chương 3**

# K�t h<p b.it bi�n trong kiJm chung chương trình

## Công cv kiJm chung chương trình VTSEinv

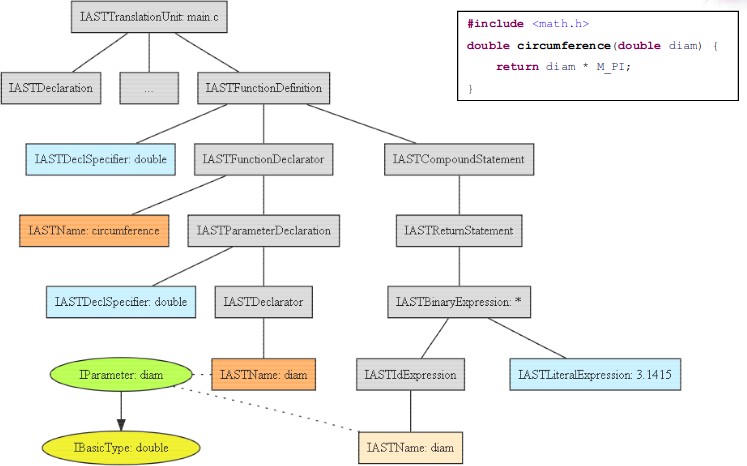


Hình 3.1: Ki�n trúc cua VTSEinv

Khóa lui;n xin đưQc đe xuat m<t phương pháp kiJm ch11ng chương trình dija trên sij k�t hQp cua kĩ thui;t thijc thi tưQng trưng và sinh bat bi�n vòng li;p. Đ6ng thai, công cv VTSEinv đưQc xây dijng trên phương pháp này đã đưQc cài đi;t. VTSEinv là phiên ban md r<ng cua công cv VTSE đã đưQc trình bày trong nghiên c11u trư6c đâ[y[15].](#_bookmark90)

#### Cây cú pháp truu tư<ng

Cây cú pháp trlfu tưQng là m<t cách th11c biJu dien có cau trúc cua mã ngu6n chương trình. ĐJ chuyJn tlf mã ngu6n C/C++ ve cây cú pháp trlfu tưQng, thư vi�n CDT [1](#_bookmark38) cua Eclipse đưQc su dvng. Nút ngu6n cua cây đưQc CDT đưa ra là m<t IASTTransla- tionUnit. Moi thành ph&n con cua cây đ1i di�n cho m<t cau trúc ngfi pháp cua chương trình.



Hình 3.2: Cây cú pháp trlfu tưQng trong CDT

1h[ttps://www.eclip](http://www.eclipse.org/cdt/)se.org/[cdt/](http://www.eclipse.org/cdt/)

#### Đ6 thj lu6ng đi�u khiJn

Đ6 thi lu6ng đieu khiJn *(Control Flow Graph)* là m<t đ6 thi có hư6ng thJ hi�n tat ca các đưang có thJ đưQc thijc thi cua chương trình. Đ6 thi đưQc sinh ra b�ng cách duy�t cây cú pháp trlfu tưQng tlf trên xu6ng. Lu6ng đieu khiJn đưQc xây dijng trong VTSEinv dija trên 4 lo1i nút chính:

–*Nút đơn* tương 11ng v6i các câu l�nh đơn bao g6m câu l�nh khdi t1o, câu l�nh gán và l�nh tra ve.

–*Nút đieu ki(!n* trong câu l�nh if-else, do-while, switch-case, và có ch11a biJu th11c đieu ki�n cùng hai nhánh tro đ�n hai ph&n then, else.

–*Nút đánh dau* đưQc dùng đJ đánh dau m<t s6 vi trí như bit đ&u vòng li;p, bit đ&u re nhánh, . . .

–*Nút khác* bao g6m nút label, nút goto, nút không xác đinh . . .

##### Xây d\fng đ6 thj lu6ng đi�u khiJn

Phương pháp su dvng là duy�t tlfng nút cua cây cú pháp trlfu tưQng, khi đó v6i tlfng cau trúc cú pháp, đ6 thi tương 11ng đưQc xây dijng. Ban đ&u, đ6 thi cua tlfng hàm con trong chương trình đưQc t1o. Sau đó, VTSEinv se n6i tat ca các đ6 thi con thành m<t đ6 thi hoàn chinh cua chương trình.

Cv thJ, sau khi có ti;p đ6 thi cua tlfng hàm, g9i là các *intraCFG*, đ6 thi lu6ng đieu khiJn chung *interCFG* đưQc xây dijng. Chương trình se tìm đ�n hàm *main*, hoi;c m<t hàm đưQc đinh nghĩa san, sau đó duy�t *intraCFG* cua hàm này đJ t1o *interCFG*. Khi gi;p nút trong đ6 thi đánh dau là lai g9i hàm, thì *intraCFG* tương 11ng cua hàm đưQc g9i se n6i vào *interCFG*. Trư6c khi n6i, các nút tương 11ng cho tham s6 cua hàm đưQc g9i cũng đưQc thêm vào *interCFG*. Nói chung, vi�c xây dijng đ6 thi lu6ng đieu khiJn *intraCFG* là tương tij v6i công cv VTSE phiên ban cũ, thui;t toán cv thJ đưQc trình bày t1i Thui;t toán [2.](#_bookmark41)

**Thuật toán 2: Thuật toán xây dựng đồ thị luồng điều khiển**

**Input :** Abstract Syntax Tree ast

**Output:** Control Flow Graph cfg

**1 Function** *buildCFG(ASTNode astNode) : ControlFlowGraph*

**2 if** *astNode is single node* **then**

**3** node *←* single node from astNode

**4** cfg *←* cfg connect with new cfg created from above node

**5 else if** *astNode is block of nodes* **then**

**6 foreach** *node n ∈ block of nodes* **do**

**7** subCfg *←* buildCfg(n)

**8** n6i subCfg vào cfg chính;

**9 end**

**10 else if** *astNode là câu l(!nh If-else* **then**

**11** thenBranch *←* buidCFG(astNode.then)

**12** elseBranch *←* buildCFG(astNode.else)

**13** decisionNode *←* nút biJu th11c đieu ki�n

**14** gán decisionNode v6i thenBranch và elseBranch

**15** t1o nút beginIf và endIf;

**16** cfg *←* new CFG(beginIf, endIf)

**17 else if** *astNode là câu l(!nh For* **then**

**18** t1o nút beginFor và endFor

**19** initNode *←* ch11a câu l�nh bit đ&u vòng li;p

**20** decisionNode *←* ch11a đieu ki�n li;p

**21** thenBranch *←* buildCFG(astNode.forBody)

**22** elseBranch *←* k�t thúc for

**23** iterationNode *←* bư6c nhay /\* vd: i ++; i = i + 2; ... \*/

**24** cfg *←* new CFG(beginFor, endFor)

**25 else if** *astNode là loi g9i hàm* **then**

**26** functionCallNode *←* câu l�nh ch11a lai g9i hàm

**27** /\* cfg ban đ�u chi chua loi g9i, cfg cua tung hàm đư�c n6i vào sau \*/

**28**

**29 else if** *astNode là câu l(!nh While-do/Do-while* **then**

**30** tương tij câu l�nh for

**31** . . .

**32 else if** *astNode là câu l(!nh Switch-case* **then**

**33 foreach** *n ∈ switch tu dưdi lên* **do**

**34 if** *n là kh8i default* **then**

**35** subCfg *←* buildCFG(n)

**36** lastNode *←* subCfg.start

**37 else if** *n là kh8i case* **then**

**38** moi case t1o m<t kh6i CFG If-else

**39** lastNode *←* conditionNode trên cùng

**40** t1o beginSwitch, endSwitch

**41** cfg *←* new CFG(beginSwitch, endSwitch)

**42 return** *cfg*

##### Ga vòng l�p trong đ6 thj lu6ng đi�u khiJn

V6i vòng li;p trong đ6 thi lu6ng đieu khiJn, có hai chi�n lưQc đưQc su dvng là ga vòng li;p v6i s6 l&n nhat đinh và su dvng bat bi�n. V6i phương pháp ga (còn g9i là *unfolding*), moi vòng li;p đưQc chuyJn ve d1ng m<t ti;p các câu l�nh *if-else* n6i nhau. S6 l&n ga vòng li;p đưQc đinh nghĩa bdi ngưai dùng hoi;c mi;c đinh là 100 l&n ga. V6i tlfng l&n ga, m<t ban sao cua nút đieu ki�n và đo1n đ6 thi thân vòng li;p đưQc t1o ra và gin vào đ6 thi m6i dư6i d1ng *if-else*. ĐJ sao chép đ6 thi, thư vi�n Kryo [2](#_bookmark42) phát triJn bdi Google đưQc su dvng. Phương pháp ga vòng li;p đưQc trình bày b�ng Thui;t toán [4](#_bookmark43) dư6i đây.

2https://github.com/EsotericSoftware/kryo

**Thu�t toán 3:** Thui;t toán ga vòng li;p

**Input :** ControlFlowGraph loopGraph

**Output:** ControlFlowGraph unfoldGraph

**1** /\* create copy of then clause \*/

**2** decisionNode *←* loopGraph.decision

**3** cpyThenClause *←* decisionNode.then

**4** lastNode *←* loopGraph.end

**5** /\* create multiple if-else from bottom to up \*/

**6 for** *i ∈ number\_of\_loops* **do**

**7** condition *←* copy(decisionNode)

**8** condition.else *←* endNode

**9** condition.then *←* cpyThenClause

**10** cpyThenClause.exit.next *←* lastNode

**11** lastNode *←* condition

**12 end**

**13** unfoldGraph *←* new ControlFlowGraph(lastNode, endNode)

#### K�t h<p b.it bi�n vòng l�p vào VTSEinv

ĐJ sinh biJu th11c bat bi�n, mã ngu6n ch11a vòng li;p đưQc kiJm tra xem có phù hQp v6i biJu mau *(template)* đưQc đinh nghĩa trư6c. Do hi�n nay phương pháp sinh bat bi�n là b6 đe Farkas, vì vi;y nên *template* vòng li;p đưQc xu lý cũng c&n phù hQp v6i đ&u vào cua công cv InvaGen đã đưQc trình bày d Chương [2.](#_bookmark20) Các *template* đưQc đinh nghĩa trư6c là các vòng li;p *for*, *while* đơn hoi;c có ch11a câu l�nh re nhánh trong vòng li;p. Trưang hQp vòng li;p l6ng nhau *(nested loop)* chưa đưQc xu lý. Đo1n Mã ngu6n

[3.1](#_bookmark45) thJ hi�n cho m<t chương trình ch11a vòng li;p và đ&u vào tương 11ng cua InvaGen đJ sinh bat bi�n. ĐJ chuyJn tlf mã ngu6n C ve m<t h� th6ng chuyJn đ6i tr1ng thái, ban đ&u *template* cua vòng li;p đưQc kiJm tra và tlf đay nhfing thành ph&n cua câu l�nh li;p đưQc lưu l1i dư6i d1ng *Transition System*. Sau khi bat bi�n đưQc sinh ra tlf InvaGen, biJu th11c bat bi�n đưQc thêm vào đ6 thi lu6ng đieu khiJn đJ dijng lên InvariantCFG. Thui;t toán [4](#_bookmark46) trình bày các bư6c dijng đ6 thi InvariantCFG.

Mã ngu6n 3.1: Ví dv chương trình C Mã ngu6n 3.2: TransitionSystem.xml

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | int main() { | |  | 1 | <TransitionSystem> | | | |
| 2 | int x = 0; | |  | 2 | <Variables>x</Variables> | | | |
| 3 | while (x < | | 1000) { | 3 | <Initiation> x = 0 </Initiation> | | | |
| 4 | if (x < | | 100) { | 4 | <Consecutions> | | | |
| 5 | x += 1; | | | 5 | <Transition> x < 100; x’ = x + 1 | | | |
| 6 | } else { | | |  | </Transition> | | | |
| 7 |  | x += | 2; | 6 | <Transition> x >= 100; | x’ | = | x + 2 |
| 8 |  | }} |  |  | </Transition> |  |  |  |
| 9 |  | return 0; |  | 7 | </Consecutions> |  |  |  |
| 10 | } |  |  | 8 | </TransitionSystem> |  |  |  |

**Thu�t toán 4:** Thui;t toán dijng đ6 thi xu lý bat bi�n

**Input :** ControlFlowGraph inter\_cfg

**Output:** ControlFlowGraph invariant\_cfg

**1 Function** *iterateCFG(CFGNode inter\_cfg.start): invariant\_cfg*

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**10**

**11**

**12**

**13**

**if** *node là beginFor || beginWhile* **then**

/\* thêm Invariant node

decisionNode *←* begin.next

notCondition *←* getNotCondition(decisionNode) invariantNode.next *←* notCondition

invariantSubCFG *←* new CFG(invariantNode,notCondition) invariantSubCFG.exit *←* iterateCFG(endNode)

**else if** *node là beginIf* **then**

condition *←* begin.next; condition.setThen(iterateCFG(condition.then)) condition.setElse(iterateCFG(condition.else))

**else if** *node là plainNode || labelNode || endNode* **then**

node.next *←* iterateCFG(node.next)

\*/

#### Đánh chi s6 và sinh biJu thuc ràng bu9c

Ve cơ ban, phương pháp đánh chi s6 và duy�t ra biJu th11c trlfu tưQng hóa chương trình trong VTSEinv tương tij phiên ban trư6c đó. Tuy nhiên, VTSEinv có sua l1i đ&u ra cho phù hQp theo chudn cua SV-COMP và k�t n6i thêm v6i m<t s6 b< giai SMT khác ngoài Z3 như MathSat, raSat, ...

ĐJ t1o ra công th11c ràng bu<c tlf đ6 thi lu6ng đieu khiJn, đ6 thi đưQc duy�t tlf trên xu6ng dư6i và đánh chi s6 cho tlfng l&n xuat hi�n cua bi�n. Vi�c đánh chi s6 là c&n thi�t do trong b< giai SMT không có khái ni�m bi�n mà chi xu lý các h�ng s6. Vì vi;y đJ thJ hi�n sij thay đ6i cua bi�n s6, ta phai đánh chi s6 cho tlfng bi�n tương 11ng trong quá trình thijc thi chương trình. Bi�n đưQc khdi t1o v6i chi s6 index -1, v6i moi l&n ci;p nhi;t giá tri thì chi s6 cua bi�n tăng thêm m<t đơn vi.

Công th11c trlfu tưQng hóa cua chương trình sau khi đưQc sinh ra se đưQc k�t hQp v6i đieu ki�n ngưai dùng đJ t1o thành biJu th11c ràng bu<c cu6i cùng. G9i *Ffinal* là biJu th11c cu6i cùng, *Fprogram* là biJu th11c trlfu tưQng hóa cua chương trình và *Fassertion* là đieu ki�n ngưai dùng. Ta có

*Ffinal* = *Fprogram ∧ ¬Fassertion*

Công th11c này se đưQc chuyJn ve đinh d1ng SMT-LIB2 và thêm vào b< giai SMT. VTSEinv ho trQ hai b< giai là Z3[3](#_bookmark48) và Mathsat[4](#_bookmark49). Tlf đ&u ra cua b< giai, công cv đưa k�t qua kiJm ch11ng chương trình tương 11ng.

* K�t qua do SMT tra ve là **sat** (satisfaction): t6n t1i giá tri đJ làm thoa mãn

*Ffinal*, nghĩa là đieu ki�n ngưai dùng không phai luôn đúng v6i mã ngu6n và có đưang thijc thi bi vi ph1m. Khi này, VTSEinv tra ve k�t qua **false** và đưa ra phan ví dv tương 11ng.

* K�t qua do SMT tra ve là **unsat** (unsatisfaction): không t6n t1i giá tri đJ thoa mãn *Ffinal*, nghĩa là đieu ki�n ngưai dùng thoa mãn so v6i mã ngu6n. VTSEinv trong trưang hQp này se tra ve **true**.
* K�t qua cua SMT là **timeout**: b< giai không đưa ra đưQc k�t qua trong thai

gian quy đinh. VTSEinv tra ve k�t qua **timeout** tương tij.

3https://rise4fun.com/z3/ 4<http://mathsat.fbk.eu/>

## Minh ho9- ho9-t đ9ng cu.a VTSEinv

Trong ph&n này, khóa lui;n xin minh h9a quy trình kiJm ch11ng cua VTSEinv b�ng hai phương pháp ga vòng li;p và su dvng bat bi�n.

##### Mã ngu6n chương trình

Xét bài toán kiJm ch11ng mã ngu6n C dư6i đây:

1 int multivar\_1\_1(int x) {//pre-condition x = 0;

2 int y = x;

3 while (x < 1024) {

4 x = x + 1;

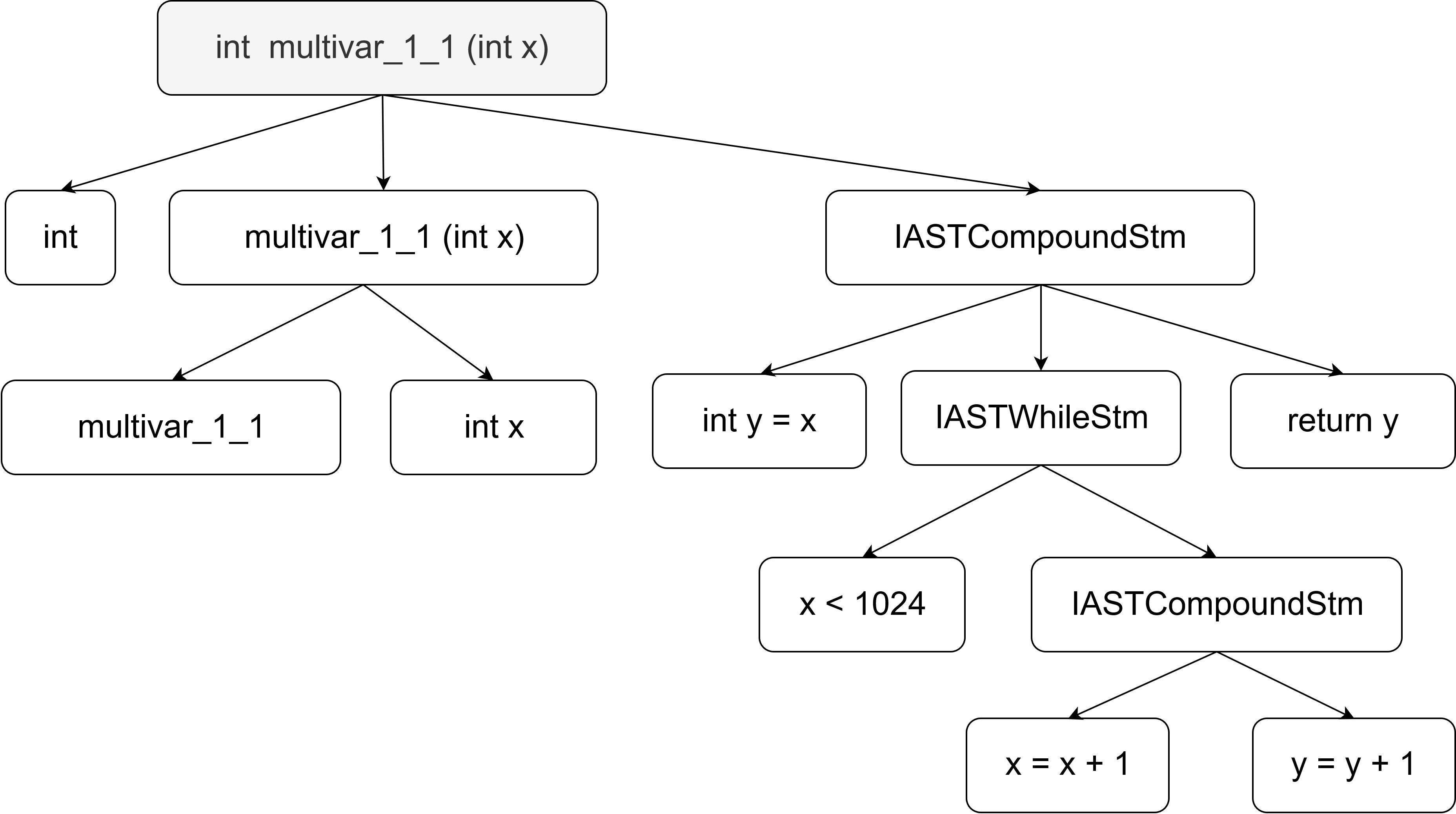
5 y = y + 1;}

6 return y;

7 VERIFIER\_assert(return < 1024); //assertion user want to verify}

##### Cây cú pháp truu tư<ng

Thư vi�n CDT đưQc su dvng đJ sinh ra AST v6i cau trúc như Hình **??**.



Hình 3.3: Cây cú pháp trlfu tưQng tlf mã ngu6n

##### Sinh b.it bi�n vòng l�p

Mã ngu6n C đưQc xu lý chuyJn ve d1ng Transition System tương 11ng và đưa vào InvaGen.

Mã ngu6n 3.3: multival\_1\_1.xml

1 <TransitionSystem>

2 <Variables> x; y </Variables>

3 <Initiation> y = x </Initiation>

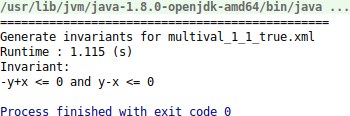
4 <Consecutions>

5 <Transition> x’ = x + 1; y’= y + 1 </Transition>

6 </Consecutions>

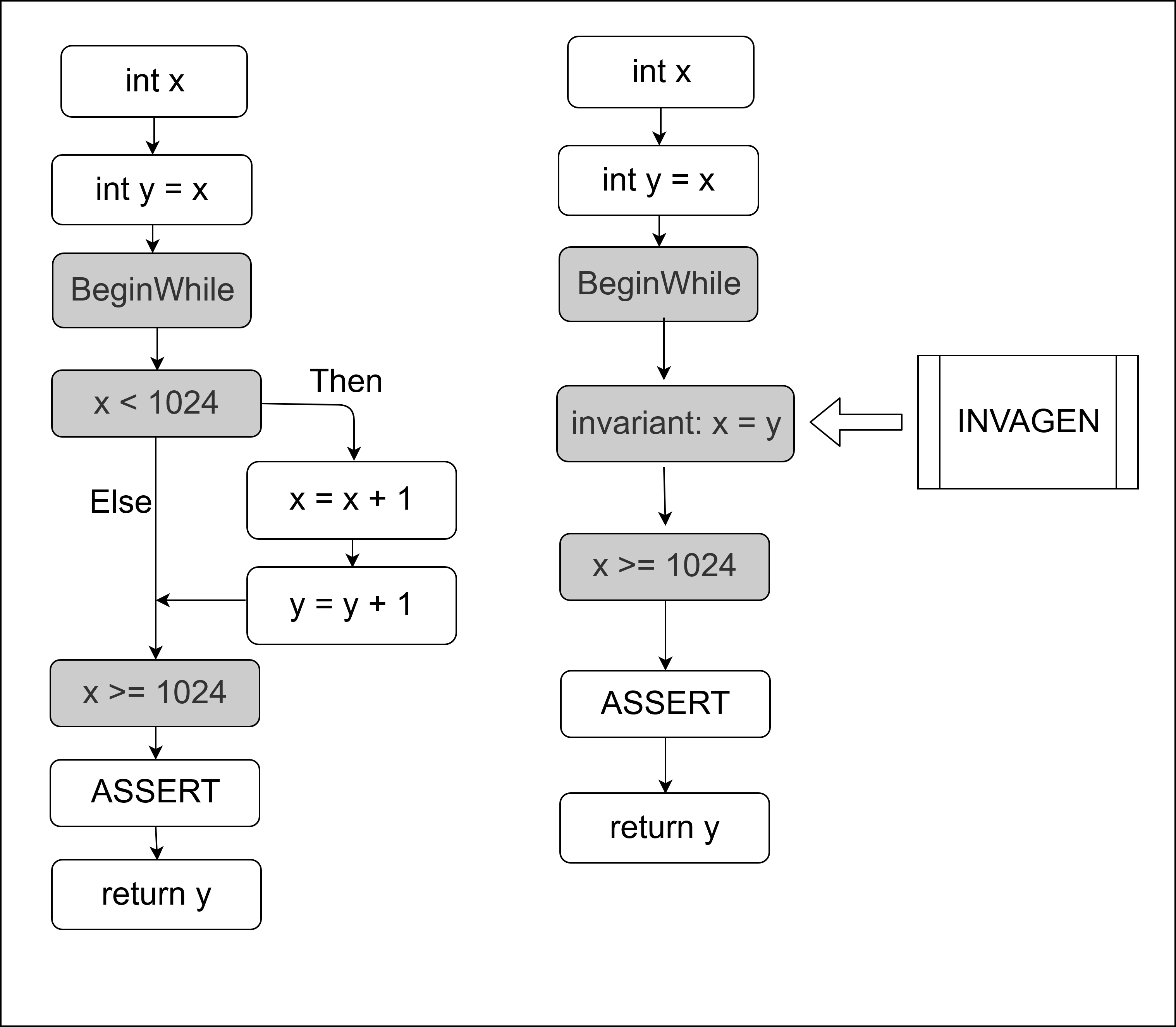
7 </TransitionSystem>

V6i file đ&u vào xml d trên, InvaGen xu lý và đưa ra k�t qua là công th11c bat bi�n cua vòng li;p như sau:



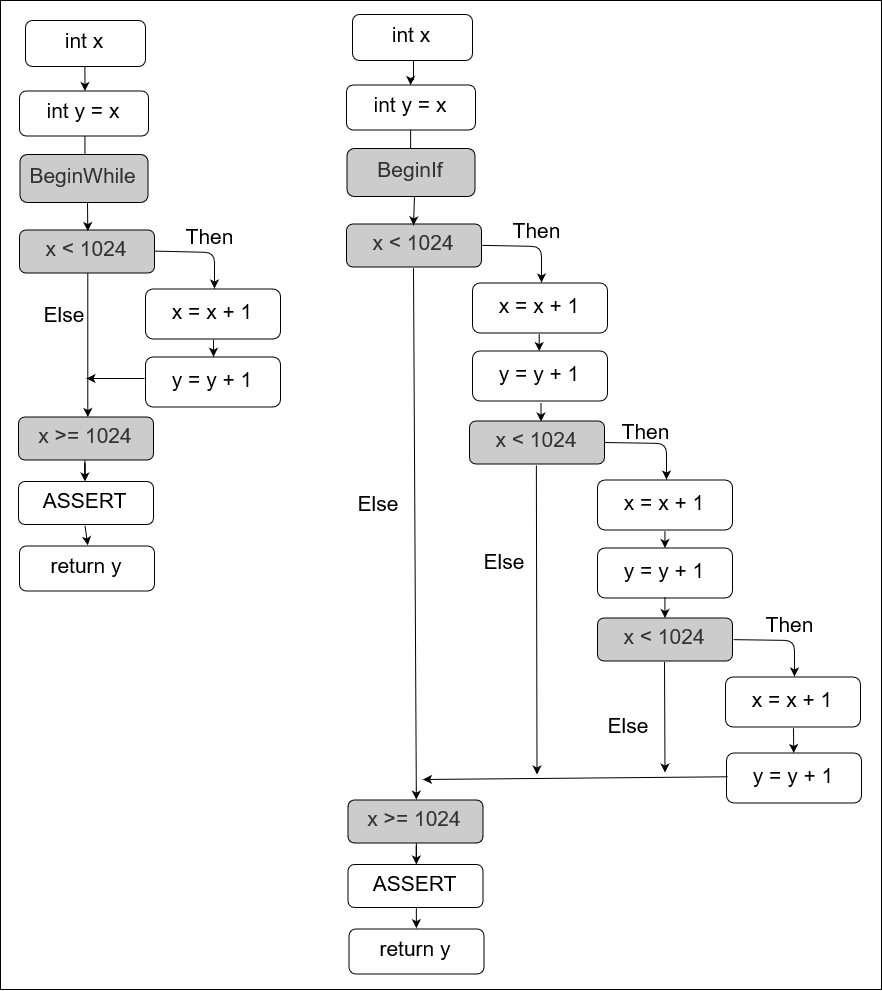
##### Đ6 thj lu6ng đi�u khiJn

Có hai đ6 thi lu6ng đieu khiJn đưQc xây dijng riêng bi�t cho hai chi�n lưQc xu lý vòng li;p khác nhau. V6i phương pháp dùng bat bi�n vòng li;p, công th11c bat bi�n sau khi thu tlf InvaGen se đưQc gin th�ng vào CFG. Trong đ6 thi xu lý bat bi�n vòng li;p, công th11c bat bi�n đưQc gin thay cho thân vòng li;p. Khi này, bat bi�n là m<t xap xi trên, thJ hi�n các tính chat luôn đưQc thoa mãn xuyên su6t các l&n thijc thi cua vòng li;p. Công th11c bat bi�n càng sát v6i vòng li;p thijc t� thì khi su dvng đJ kiJm ch11ng chương trình se càng cho k�t qua chính xác.



Hình 3.4: CFG chèn bat bi�n vòng li;p

Phương pháp ga vòng li;p se chuyJn câu l�nh li;p thành m<t ti;p các l�nh *if-else* v6i s6 l&n ga nhat đinh. Con s6 này đưQc đinh nghĩa bdi ngưai dùng trư6c khi bit đ&u kiJm ch11ng, hoi;c đưQc ga theo mi;c đinh là 100 l&n. ĐJ de dàng biJu dien, ví dv minh h9a trên se đưQc ga v6i s6 l&n li;p là 3, tương đương thân cua vòng *while* đưQc thijc thi 3 l&n. Đ6 thi lu6ng đieu khiJn khi ga vòng li;p đưQc trình bày d Hình [3.5.](#_bookmark53) V6i moi l&n ga vòng li;p, s6 lưQng bi�n đưQc tăng lên đáng kJ. Vì vi;y, v6i nhfing bài toán có gi6i h1n li;p l6n, kích thư6c đ6 thi lu6ng đieu khiJn khi ga se tăng nhanh, dan đ�n làm ph11c t1p hóa công th11c ràng bu<c.



Hình 3.5: CFG phương pháp ga vòng li;p

##### Đánh chi s6 và sinh công thuc ràng bu9c

V6i đ6 thi invariantCFG hay unfoldCFG, đ6 thi đeu đưQc đánh chi s6 và duy�t tlf trên xu6ng đJ thu đưQc công th11c trlfu tưQng cua mã ngu6n chương trình *Fprogram*. K�t hQp v6i công th11c đieu ki�n ngưai dùng *Fassertion*, VTSEinv t1o ra file đinh d1ng SMT-LIBv2, là đ&u vào cua b< giai SMT.

Mã ngu6n 3.4: File *multival\_1\_1.smt* phương pháp bat bi�n vòng li;p

1 (declare-fun x\_multivar\_1\_1\_-1 () Int)

2 ...

3 (declare-fun y\_multivar\_1\_1\_0 () Int)

4 (declare-fun return\_multivar\_1\_1\_0 () Int)

5 (assert (and (and (=> (< x\_multivar\_1\_1\_0 1024)

6 (and (= x\_multivar\_1\_1\_1 y\_multivar\_1\_1\_0)

7 (= x\_multivar\_1\_1\_1 1024)))

8 (=> (not (< x\_multivar\_1\_1\_0 1024))

9 (= x\_multivar\_1\_1\_1 x\_multivar\_1\_1\_0)))

10 (= return\_multivar\_1\_1\_0 y\_multivar\_1\_1\_0)))

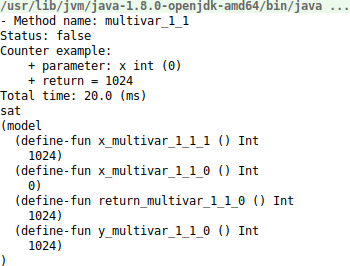
11 (assert (= x\_multivar\_1\_1\_0 0))

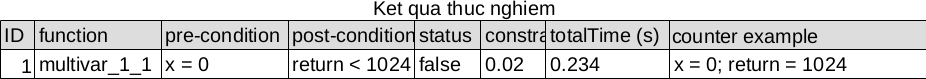
12 (assert (not (< return\_multivar\_1\_1\_0 1024)))

13 (check-sat)

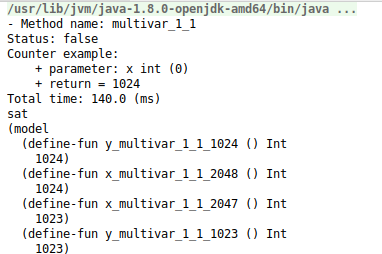
14 (get-model)

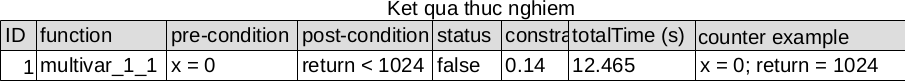
Cu6i cùng, VTSEinv dija vào k�t qua cua b< giai SMT đJ đưa ra k�t lui;n. Đ6ng thai, m<t báo cáo Excel thJ hi�n k�t qua kiJm ch11ng đưQc t1o ra. Dư6i đây thJ hi�n hai k�t qua cua VTSEinv khi l&n lưQt su dvng bat bi�n vòng li;p và ga vòng li;p 100 l&n.





Hình 3.6: K�t qua tlf Z3 (trên) và báo cáo Excel (dư6i) cua VTSEinv dùng Invariant





Hình 3.7: K�t qua tlf Z3 (trên) và báo cáo Excel (dư6i) cua VTSEinv dùng Unfold

Dùng ca hai phương pháp, công cv đưa ra k�t qua **false** và v6i tham s6 đ&u vào (parameter) *x = 0* thì giá tri cua k�t qua tra ve là *return = 1024*. Có thJ thay, v6i bài toán trên, ca hai phương pháp đeu có thJ đưa ra k�t qua kiJm ch11ng chương trình là **false** và đưa ra phan ví dv chính xác. Tuy thai gian sinh ra công th11c ràng bu<c là không chênh l�ch nhieu, nhưng t6ng thai gian giai khi su dvng bat bi�n cho t6c đ< t6t hơn so v6i ga vòng li;p.

**Chương 4**

# Nghiên cuu liên quan

Trong chương này, khóa lui;n xin trình bày m<t s6 nghiên c11u liên quan đ�n các công cv kiJm ch11ng ph&n mem và phương pháp sinh bat bi�n vòng li;p khác. Hai công cv đưQc nêu ra là CBMC và VeriAbs, đây là hai công cv đưQc ti�n hành thijc nghi�m và so sánh v6i VTSEinv. Sau đó, hai kĩ thui;t sinh bat bi�n đang phát triJn trong thai gian g&n đây, đã đưQc thJ hi�n là cho k�t qua t6t hơn so v6i vi�c dùng b6 đe Farkas se đưQc trình bày.

## M9t s6 công cv kiJm chung phfin m�m

CBMC là m<t công cv đi đ&u trong kiJm ch11ng ph&n mem v6i kĩ thui;t Bounded Model Checking. Trong khi đó, Veriabs là công cv md r<ng trên nen tang CBMC khi áp dvng thêm phương pháp *k-induction* và bat bi�n vòng li;p.

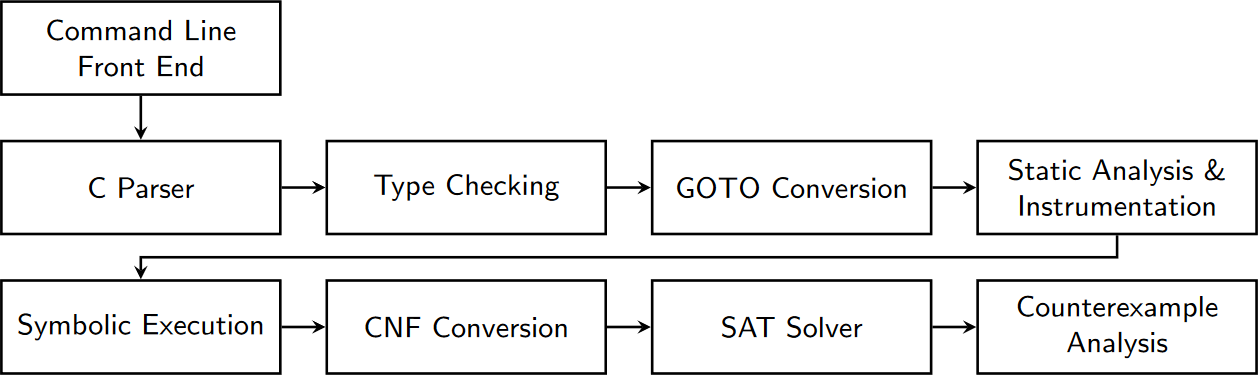
### CBMC

Công cv C Bounded Model Checker (CBMC) [[11]](#_bookmark86) có thJ tìm ra sij vi ph1m cua m<t biJu th11c đieu ki�n *assertion* trong chương trình C hoi;c ch11ng minh biJu th11c là đúng v6i m<t gi6i h1n cho trư6c. CBMC cài đi;t m<t b< dich *bit-percise* cua m<t đ&u vào chương trình C, nghĩa là bi�n đ6i chương trình v6i vòng li;p đưQc ga m<t s6 l&n

nhat đinh ve công th11c bit-vector v6i kích thư6c c6 đinh. N�u công th11c là thoa mãn thì có nghĩa là chương trình dan đ�n m<t *assertion* bi vi ph1m. B< giai đJ xét tính thoa đưQc cua công th11c đưQc su dvng là MiniSat 2.2.0 [[6].](#_bookmark81)

##### Ki�n trúc cu.a CBMC

*Bounded model checker* như CBMC chuyJn bài toán phân tích đưang thijc thi cua chương trình ve công th11c ràng bu<c có thJ đưQc giai b�ng đ< giai SAT hay SMT. V6i vi�c su dvng b< giai SAT, CBMC có thJ nhi;n đ&u vào là m<t chương trình C v6i các đieu ki�n *assertion* c&n kiJm ch11ng. CBMC cho ra m<t biJu th11c CNF biJu dien các đưang thijc thi cua chương trình dan đ�n vi ph1m *assertion*. Quy trình CBMC bi�n đ6i đưang thijc thi đưQc trình bày d Hình [4.1.](#_bookmark59) Ki�n trúc cua CBMC bao g6m b6n ph&n chính.



Hình 4.1: Ki�n trúc cua CBMC

*Front end.* Ban đ&u, CBMC cho phép ngưai dùng nhi;p m<t s6 tùy ch9n ví dv như đ< r<ng cua bit, s6 vòng li;p, ... M<t *C parser* xây dijng cây cú pháp tlf mã ngu6n r6i chuyJn qua b< *Type checking*. Moi kí tij trong cây đưQc gán v6i m<t kiJu thông tin d d1ng bit.

*Intermediate Representation.* CBMC su dvng *GOTO programs* như m<t biJu dien trung gian. V6i biJu dien trung gian này, m9i lu6ng thijc thi phi tuy�n tính ví dv như if, switch, vòng li;p, ... đưQc chuyJn ve các câu l�nh goto tương 11ng.

*Middle end.* CBMC áp dvng thijc thi tưQng trưng b�ng cách ga vòng li;p m<t s6 l&n hfiu h1n, con s6 này đưQc quy�t đinh bdi ngưai dùng. Sau khi thijc hi�n ga vòng li;p, CBMC chuyJn chương trình d1ng GOTO ve d1ng *static single asignment (SSA)*.

*Back end.* CBMC chuyJn chương trình tlf d1ng SSA ve biJu th11c CNF b�ng cách li;p mô hình bit cua tat ca các câu l�nh kèm thêm các ràng bu<c Boolean. Sau đó, công th11c CNF đưQc đưa vào MiniSat 2.2.0 đJ kiJm ch11ng và đưa ra phan ví dv n�u có.

##### ĐiJm m9-nh và h9-n ch�

CBMC có điJm m1nh ve t6c đ< và s6 bài toán có thJ giai đưQc. Ve mi;t h1n ch�, CBMC không thJ xu lý các bài toán mà vòng li;p là không gi6i h1n. Tuy nhiên, v6i bài toán ch11a vòng li;p, CBMC đi;t m<t m11c thai gian nhat đinh và bit bu<c chương trình cham d11t v6i đ&u ra TRUE/FALSE đJ đưa ra m<t k�t qua kha thi. Vì vi;y, trong b< thijc nghi�m xu lý vòng li;p, nhieu bài toán khi giai b�ng CBMC cho ra k�t qua là *unsound*.

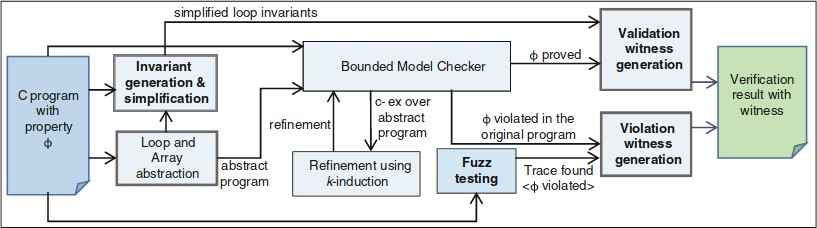
#### VeriAbs

VeriAbs [[5]](#_bookmark80) là m<t công cv kiJm ch11ng chương trình C/C++ su dvng kĩ thui;t k- induction đJ md r<ng phương pháp Bounded Model Checking cho chương trình v6i vòng li;p l6n hoi;c không xác đinh đưQc gi6i h1n li;p. VeriAbs trlfu tưQng hóa nhfing vòng li;p như vi;y thành các vòng li;p xác đinh v6i s6 l&n nho hơn và có thJ đưQc ch11ng minh bdi BMC. Ý tưdng này dija trên khái ni�m thu g9n vòng li;p *(loop shrinkability)* [[2].](#_bookmark77)

##### Ki�n trúc và quy trình kiJm chung cu.a VeriAbs

Quy trình kiJm ch11ng cua VeriAbs đưQc thJ hi�n d Hình [4.2.](#_bookmark62) VeriAbs chuyJn file mã ngu6n C qua m<t Tata Consultancy Services (TCS)[1](#_bookmark61) đJ sinh ra biJu dien trung

1h[ttps://www.tcs.c](http://www.tcs.com/)om/



Hình 4.2: Ki�n trúc VeriAbs

gian *(intermediate representation)* cua chương trình. Sau đó, ban IR này đưQc phân tích su dvng PRISM đJ thijc hi�n trlfu tưQng hóa và tính toán. VeriAbs su dvng C Bounded Model Checker (CBMC) [[11]](#_bookmark86) phiên ban 5.8 v6i MiniSat [[6]](#_bookmark81) đJ kiJm ch11ng trlfu tưQng hóa hoi;c chương trình ban đ&u v6i s6 vòng li;p đã bi�t. VeriAbs sinh ra *correct witnesses* cua chương trình b�ng cách t1o ra các bat bi�n vòng li;p tlf hi;u đieu ki�n. ĐJ t1o bat bi�n vòng li;p, công cv su dvng Z3.4.5.1[2](#_bookmark63) đJ lo1i bo lưQng tlf, sau đó bat bi�n đưQc thêm vào *control flow automaton* sinh bdi CPAchecker ban 1.6.1 [[1].](#_bookmark76) ĐJ sinh ra *error witnesses*, VeriAbs su dvng CBMC ban 5.8.

##### ĐiJm m9-nh và h9-n ch�

ĐiJm m1nh chính cua VeriAbs là tính an toàn *(sound)*. Tat ca các bi�n đ6i thijc hi�n d công cv là xap xi trên. Khi su dvng CBMC, công cv cho phép m<t tùy ch9n **unwinding-assertions**, đJ đam bao n�u công cv thông báo là tính chat này thoa mãn thì đieu đó là chic chin. M<t lQi th� khác cua VeriAbs là công cv bi�n đ6i tat ca các vòng li;p trong chương trình ve vòng li;p trlfu tưQng v6i s6 lưQng l&n li;p là xác đinh, vì vi;y, *bounded model checker* có thJ đưQc su dvng đJ kiJm ch11ng.

H1n ch� cua Veriabs là không có quy trình dien giai trlfu tưQng đJ có thJ tìm ra loi. Tuy nhiên, công cv có thJ tìm đưQc loi thông qua *testing*. Veriabs phv thu<c vào Z3 đJ lo1i bo lưQng tlf và sinh ra bat bi�n vòng li;p, phv thu<c vào CPAChecker đJ sinh ra *automata* cua chương trình.

2https://rise4fun.com/z3/

## Phương pháp sinh b.it bi�n vòng l�p

#### Sinh b.it bi�n su' dvng N9i suy Craig

##### N9i suy Craig

Cho m<t ci;p công th11c (A,B) trong đó *A ∧ B* là không thoa mãn, m<t công th11c n<i suy cho (A, B) là m<t công th11c I v6i các tính chat sau[[13](#_bookmark88)]:

* + - * A *→* I
      * I *∧* B là *unsat*
      * Ti;p ngôn ngfi cua I thu<c giao cua A và B.

B6 đe n<i suy Craig nói r�ng m<t n<i suy luôn t6n t1i v6i A và B là *unsat*.

##### Sinh b.it bi�n vòng l�p su' dvng n9i suy

H� th6ng công th11c đưQc su dvng đJ thJ hi�n h� th6ng chuyJn đ6i tr1ng thái dư6i d1ng công th11c vi tlf cap m<t. Ta g9i S là m<t tr1ng thái cua h� th6ng. V6i m9i kí hi�u s *∈* S, ta có kí hi�u s’ đ1i di�n cho các giá tri s t1i m<t đơn vi tr1ng thái ti�p theo. Máy sinh bat bi�n vòng li;p G là m<t thu tvc nhi;n đ&u vào M và đ&u ra là m<t dãy các công th11c bat bi�n.

Thui;t toán sinh bat bi�n sau đưQc dija trên *unfolding* h� th6ng chuyJn đ6i tr1ng thái trong Bounded Model Checking. V6i k *≥* 0, khi ga M k-bư6c, ta có công th11c:

*Uk*(*M* ) = *I, T, T* (1)*, ..., T* (*k−*1)*, ¬Pk*

trong đó *I - initiation* là tr1ng thái md đ&u, *T - transition* là các bi�n đ6i cua chương trình và *P - postcondition* là hi;u đieu ki�n.Công th11c này thJ hi�n m<t ti;p các thijc thi cua h� th6ng chuyJn đ6i tr1ng thái khi ch1y k-bư6c và k�t thúc d tr1ng thái *unsafe*. H� th6ng M là *safe* khi *Uk*(*M* ) là không thoa mãn v6i *∀k*.

M<t trong nhfing công cv đ&u tiên su dvng n<i suy Craig đưQc cài đi;t là SPASS[3](#_bookmark66).

3h[ttps://www.spass](http://www.spass-prover.org/)-prov[er.org](http://www.spass-prover.org/)

Thui;t toán [5](#_bookmark67) trình bày phương pháp sinh bat bi�n vòng li;p trong SPASS.

**Thu�t toán 5:** Thui;t toán sinh bat bi�n su dvng n<i suy Craig **Input :** H� th6ng M=(I,T,P) có m<t công th11c bat bi�n **Output:** M<t chuoi các công th11c bat bi�n cua M

**1** i *←* 1; k *←* 1

**2** Li;p:

**3** Su dvng b< giai cho *Uk*(*M* )

**4 if** *b(j giiii trii ve m(jt n(ji suy I* **then**

**5 for** *j = 1 → k+1* **do**

**6** Xây dijng V*l*=1*...j* (*Il*)

**7** k++

**8 end**

**9 else**

**10**

**11**

//b< giai không tra ve đưQc n<i quy

i ++

Thui;t toán trên su dvng chi�n lưQc ga ti�n d&n d&n tlfng tr1ng thái cua h� th6ng. Vì vi;y, bat bi�n sinh ra thưang là xap xi trên rat r<ng cua chương trình và không đu m1nh đJ ch11ng minh tính chat cua h� th6ng. M<t chi�n lưQc ga tr1ng thái m6i hơn đưQc gi6i thi�u t1i ASE 2017 [[17].](#_bookmark92) Trong báo cáo này, thay vì chi dùng *forward unfolding*, các tác gia k�t hQp ca *backward unfolding* đJ rút g9n đưQc bat bi�n là sát v6i h� th6ng nhat. Phương pháp này có ưu th� so v6i b6 đe Farkas do *backward unfolding* sinh ra đưQc nhfing bat bi�n khác nhau tùy vào đieu ki�n c&n kiJm tra. Vì vi;y, có thJ đam bao bat bi�n là đu m1nh đJ kiJm ch11ng chương trình v6i tlfng đieu ki�n ngưai dùng cv thJ

#### Sinh và làm mjn b.it bi�n b�ng phương pháp l.iy m§.u chon loc

Như đã nói d trên, vi�c tìm đưQc bat bi�n đu m1nh đJ phân tích và kiJm ch11ng chương trình là bài toán không quy�t đinh. M<t trong nhfing thui;t toán làm min bat bi�n đã đưQc Jiajying Li và c<ng sij [[12]](#_bookmark87) đưa ra t1i h<i nghi ASE 2017. Tác gia đã đe

xuat phương pháp sinh bat bi�n vòng li;p b�ng cách k�t hQp H9c máy và KiJm ch11ng chương trình, đ6ng thai gi6i thi�u công cv ZILU [4](#_bookmark69) su dvng phương th11c này. V6i đ&u vào là m<t chương trình ch11a vòng li;p tuân thu Logic Hoare, công cv ZILU se ngau nhiên kiJm tra chương trình *(testing)*, thu thi;p tr1ng thái chương trình t1i thai gian ch1y và phân lo1i tr1ng thái xem chúng có thoa mãn bat bi�n đưQc tìm thay. Ti�p theo, ky thui;t phân lo1i đưQc su dvng đJ t1o ra bat bi�n vòng li;p tij đ<ng. Cu6i cùng, ZILU làm min bat bi�n su dvng Lay mau có ch9n l9c. Đ�n khi bat bi�n không thJ làm min hơn, công cv kiJm tra li�u bat bi�n đó có thJ đưQc dùng đJ ch11ng minh b< ba Hoare.

##### L.iy m§.u chon loc – Selective sampling

Mau có thJ đưQc thu thi;p b�ng ba chi�n lưQc khác nhau. Ban đ&u, các mau đưQc ch9n ngau nhiên đJ làm ti;p mau khdi đ&u. Chi phí đJ sinh dfi li�u ngau nhiên thưang là thap, tuy nhiên thưang c&n m<t lưQng l6n dfi li�u ngau nhiên đJ có thJ đưa ra k�t qua chính xác. Chi�n lưQc th11 hai, lay mau ch9n l9c tiêu t6n nhieu hơn, tuy nhiên, mau ch9n l9c thu đưQc nhieu lQi ích tlf lay mau ngau nhiên. Cách cu6i cùng là ch9n mau b�ng kiJm ch11ng. Khi m<t bat bi�n tiem năng không thoa mãn m<t trong ba tính chat cua bat bi�n, b< kiJm ch11ng đưa ra phan ví dv r6i thêm vào như m<t mau m6i.

##### Phương pháp d\f đoán và kiJm tra - Guess and check

Công cv Zilu đưQc xây dijng dija trên phương pháp *“guess and check”*. Vi�c này bit đ&u v6i sinh ra m<t ti;p giá tri V (hay còn g9i là mau - *sample*) và phân lo1i chúng vào các nhóm khác nhau, có thJ là m<t nhóm thoa mãn bat bi�n vòng li;p và nhóm còn l1i g6m các giá tri không thoa mãn. M<t thui;t toán h9c ti;p sau đay đưQc su dvng đJ khái quát hóa các giá tri thoa mãn tlf đó đoán ra công th11c bat bi�n vòng li;p có thJ. Các công th11c đó se đưQc kiJm tra su dvng kĩ thui;t kiJm ch11ng chương trình (ví dv như thijc thi tưQng trưng) đJ xem li�u chúng có thoa mãn ba đieu ki�n cua bat bi�n vòng li;p đã đưQc nêu d Chương [1.](#_bookmark2) N�u có m<t đieu ki�n bi vi ph1m, ta thu đưQc phan ví dv thJ hi�n r�ng bat bi�n thu đưQc là không chính xác.

4ZILU repo. https://github.com/lijiaying/zilu, 2017

**Chương 5**

# K�t qua th\fc nghi�m

## Đi�u ki�n th\fc nghi�m

Trong chương này, khóa lui;n se trình bày k�t qua thijc nghi�m cua VTSEinv v6i các phương pháp xu lý vòng li;p cũ và các công cv khác trên b< dfi li�u đưQc lay tlf cu<c thi SV-COMP là **ReachSafety/Loop-acceleration**. SV-COMP (Competition on Software Verification) [1](#_bookmark72) là m<t cu<c thi uy tín đưQc t6 ch11c thưang niên dành cho các công cv kiJm ch11ng nh�m tìm ra kĩ thui;t và công cv có kha năng giai quy�t đưQc nhieu bài toán trong thai gian ngin nhat. B< dfi li�u đưQc su dvng đJ so sánh bao g6m các chương trình vi�t b�ng ngôn ngfi C. Các bài toán h&u h�t là các chương trình ch11a m<t vòng li;p *while* đơn, ti;p trung kiJm tra các tính chat cua vòng li;p. Đây là b< thijc nghi�m đJ SV-COMP kiJm tra kha năng xu lý vòng li;p cua các công cv tham gia.

Moi bài toán là m<t chương trình bao g6m mã ngu6n và đieu ki�n ngưai dùng c&n kiJm tra, có đinh d1ng tương tij như đưQc trình bày d ph&n [3.2.](#_bookmark50) K�t qua đưa ra cua moi bài có thJ là m<t trong ba trưang hQp:

* + - *True*: Đieu ki�n ngưai dùng đưQc thoa mãn, không có đưang thijc thi bi vi ph1m
    - *False*: Đieu ki�n nhi;p vào bi vi ph1m và có m<t phan ví dv đưQc đưa ra.
    - *Timeout/Overflow* : Công cv không thJ đưa ra k�t qua trong thai gian quy đinh

1https://sv-comp.sosy-lab.org/2019/

hoi;c tràn b< nh6.

Các công cv tham gia thijc nghi�m bao g6m VTSEinv, VTSE [[15](#_bookmark90)], VeriAbs [[5]](#_bookmark80) và CBMC [[11].](#_bookmark86) Trong đó, VeriAbs là công cv kiJm ch11ng mã ngu6n C/C++ giành giai B1c SV-COMP 2018 và giai Vàng SV-COMP 2019 d h1ng mvc ReachSafety. CBMC là m<t trong nhfing công cv đi đ&u trong kiJm ch11ng ph&n mem su dvng kĩ thui;t *Bounded Model Checking*. CBMC cũng tham gia SV-COMP trong nhieu năm và đ1t giai Vàng năm 2015. Sau đó, rat nhieu các công cv kiJm ch11ng đã đưQc phát triJn dija trên nen tang cua CBMC như JBMC, ESBMC, 2LS, ...

## K�t qua th\fc nghi�m

K�t qua thijc nghi�m đưQc trình bày d Bang [5.1](#_bookmark74) g6m các ph&n so sánh v6i phiên ban VTSE cũ và công cv khác trên b< bài toán **Loop-acceleration**. Thijc nghi�m đưQc thijc hi�n trên máy Intel i5-5200U 2.2GHz x 4, 8GB RAM, v6i thai gian thijc hi�n bài toán t6i đa là 500 giây.

Trong bang thijc nghi�m, k�t qua đúng cua bài toán dija theo SV-COMP và gi6i h1n s6 l&n vòng li;p thijc thi đưQc thJ hi�n l&n lưQt d c<t *ER - Expected Result* và *Loop bound*. Moi công cv khi giai se cho đ&u ra d c<t *R* và kèm theo là thai gian thijc hi�n tính theo giây d *T*. VTSE (phiên ban trư6c đây) và CBMC đưQc ch1y thijc nghi�m v6i hai phương án dư6i đây:

* + - *Unfold I* : Su dvng phương pháp ga vòng li;p v6i s6 l&n ga li;p là 100 l&n.
    - *Unfold II* : Su dvng phương pháp ga vòng li;p v6i s6 l&n ga li;p đưQc bài toán đưa ra *(loop bound)*. Trong trưang hQp vòng li;p không có gi6i h1n *(unbounded loop)*, mi;c đinh s6 l&n ga là 100.

Trong khi đó, VTSEinv su dvng tùy ch9n *Invariant* - kiJm ch11ng dùng bat bi�n và VeriAbs khi xu lý vòng li;p dùng ky thui;t *k-induction* trong BMC k�t hQp v6i bat bi�n vòng li;p.

Ve k�t qua, so v6i VTSE trư6c đây và CBMC, VTSEinv có s6 lưQng bài toán giai đúng và thai gian thijc hi�n t6t hơn. Khi thijc hi�n ga đúng s6 vòng li;p, VTSE luôn ra k�t qua đúng (trlf v6i bài toán không có gi6i h1n li;p), tuy nhiên se gi;p phai

van đe *Timeout* khi s6 l&n ga là quá l6n. Giam b6t s6 l&n ga giai quy�t đưQc van đe quá thai gian, tuy nhiên l1i đưa ra k�t qua không chính xác *(unsound)*, do khi này công th11c ràng bu<c đưQc sinh ra là xap xi dư6i cua chương trình. Trong khi đó, VTSEinv có s6 bài giai đúng là **14/17** bài, t6t hơn nhieu VTSE và CBMC khi hai công cv này chi giai đưQc 7 hoi;c 8 bài toán. Vì công th11c bat bi�n là xap xi trên, chi có 3 bài toán đưa ra k�t qua báo đ<ng gia *(false alarm)*, còn l1i đeu đưQc giai v6i t6c đ< nhanh, t6ng thai gian đJ kiJm ch11ng ca 17 chương trình chi là **0.385(s)**. V6i bài toán khi s6 l&n ga li;p không cao, và tat ca đeu có thJ đưa ra k�t qua đúng như *multival\_1\_1.c*, VTSEinv cũng chi mat **0.014(s)** khi VTSE và CBMC l&n lưQt c&n

**12.267(s)** và **56.248(s)**. Có thJ nói, VTSEinv đã cai thi�n đưQc k�t qua cua VTSE

khi kiJm ch11ng các bài toán có vòng li;p.

Khi so sánh v6i VeriAbs, có thJ thay VTSEinv kém ve s6 lưQng bài giai đưQc, đ6ng thai VeriAbs luôn đưa ra k�t qua đúng khi kiJm ch11ng. Tuy nhiên, thai gian đJ giai m<t bài toán cua VTSEinv là nhanh hơn rat nhieu VeriAbs. Ví dv, trong bài *mono4\_1.c*, VTSEinv chi mat **0,013(s)** trong khi VeriAbs không thJ đưa ra k�t qua trong thai gian **500(s)**. Đieu này dan đ�n t6ng thai gian kiJm ch11ng cua VTSEinv chi là **0.385(s)** so v6i **3154.8(s)** cua VeriAbs.

Có thJ thay, su dvng bat bi�n vòng li;p luôn cho k�t qua chính xác khi ch11ng minh đưQc chương trình là đúng đin (bài *multivar\_1\_1.c* và *simple\_2\_2.c*). Tuy nhiên VTSEinv có h1n ch� là đưa ra k�t qua *false alarm*, nghĩa là báo có loi trong khi chương trình đúng. Tuy nhiên, trong kiJm ch11ng ph&n mem, vi�c đưa ra k�t qua là *unsound*, nghĩa là ch11ng minh chương trình đay đúng trong khi có loi, có hi;u qua ni;ng hơn rat nhieu so v6i *false alarm*. VTSEinv đã xu lý đưQc van đe này cua CBMC và VTSE khi hai công cv này su dvng phương pháp ga vòng li;p. VTSEinv cũng có sij khác bi�t l6n ve t6c đ< so v6i VeriAbs. ĐJ không đưa ra các k�t qua *false alarm*, bat bi�n vòng li;p sinh ra c&n đưQc làm min đJ càng sát v6i công th11c ràng bu<c đúng cua chương trình càng t6t. V6i m<t phương pháp sinh bat bi�n vòng li;p đu t6t, VTSEinv se có thJ vlfa chi�m ưu th� ve t6c đ<, vlfa cho k�t qua chính xác hơn.

Bang 5.1: K�t qua b< thijc nghi�m Loop-acceleration

**Loop**

**VTSEinv**

**VTSE**

**CBMC**

**VeriAbs**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Chương trình** | **ER** | **bound** | **Inv** | **ariant** | **Unfold I** | | **Unfold II** | | **Unfold I** | | **Unfo** | **ld II** |  |  |
|  |  |  |  | **R** | **T (s)** | **R** | **T (s)** | **R** | **T (s)** | **R** | **T (s)** | **R** | **T (s)** | **R** | **T (s)** |
|  |  |  |  |  | | | | | | | | | | | |
| 1 | const\_1\_1 | T | 1024 |
| 2 | functions\_1\_1 | F | 10000 |
| 3 | functions\_1\_2 | T | 10000 |
| 4 | mono1\_1 | F | 1E+06 |
| 5 | mono3\_1 | F | 1E+06 |
| 6 | mono4\_1 | F | 1E+06 |
| 7 | mono5\_1 | F | - |
| 8 | mono6\_1 | F | - |
| 9 | multivar\_1\_1 | T | 1024 |
| 10 | multivar\_1\_2 | F | 100 |
| 11 | phases\_1\_2 | F | 1000 |
| 12 | simple\_1\_1 | F | 10000 |
| 13 | simple\_2\_1 | T | 10000 |
| 14 | simple\_2\_2 | F | 10000 |
| 15 | underapp\_1\_1 | F | 6 |
| 16 | underapp\_2\_1 | F | 6 |
| 17 | underapp\_2\_2 | T | 6 |
|  | **S6 bài đúng T6ng thoi gian** |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **F** | **0.123** | T | 3.042 | T | 6.072 | T | 0.143 | T  F T | 0.24 | T  F | 15.301 |
| F | 0.023 | **T** | **1.344** | O | *500* | **T** | **0.197** | 1.729 | 66.014 |
| **F** | **0.013** | T  F | 0.958 | O | *500* | T | 0.154 | 1.861 | O | *500* |
| F | 0.012 | 0.956 | O | *500* | **T** | **1.91** | **T** | 89.492 | F | 361.686 |
| F | 0.024 | **T** | **1.006** | O | *500* | **T** | **1.928** | **T** | 131.062 | F | 362.33 |
| F | 0.013 | F | 1.327 | O | *500* | **T** | **1.853** | **T** | 100.396 | O | 500 |
| F | 0.013 | **T** | **2.63** | **T** | **2.704** | **T** | **1.293** | **T** | 78.772 | F | 364.858 |
| F | 0.024 | **T** | **0.981** | **T** | **1.003** | **T** | **1.387** | **T** | 82.862 | F | 366.051 |
| **T** | **0.014** | T | 1.039 | T F  F | 12.267 | T  F | 0.743 | T  F | 56.248 | T | 8.509 |
| F | 0.03 | **T** | **1.002** | 3.594 | 0.482 | 0.368 | F | 7.945 |
| F | 0.012 | **T** | **0.969** | 11.919 | **T** | **0.143** | **T** | **0.884** | F | 360.001 |
| F | 0.012 | **T** | **1.438** | O | *500* | **T** | **0.146** | **T** | **0.792** | F | 205.325 |
| **T** | **0.022** | T | 1.03 | O | *500* | T F  F | 0.354 | O | *500* | T | 7.067 |
| F | 0.012 | **T** | **1.097** | O | *500* | 0.323 | O | *500* | F | 6.105 |
| F | 0.012 | F F T **9** | 0.831 | F F T  **7** | 0.807 | 0.143 | F F T  **7** | 0.1 | F | 9.173 |
| F | 0.013 | 1.372 | 0.754 | **T** | **0.139** | 0.107 | F | 6.091 |
| **F** | **0.013** | 1.265 | 0.672 | T | 0.138 | 0.1 | T | 8.352 |
| **14** | **0.385** | **22.287** | **4039.79** | **8** | **11.476** | **1545.01** | **15** | **3154.81** |

False alarm

Unsound

Timeout/Overflow

43

*ER (Expected result): Ket quii đúng, Loop bound: Gidi h¤n vòng lrJp, R: Ket quii cõa công cV, T(s): Thoi gian(s) T: True, F: False, O: Time out/Overflow*

# K�t lu�n

Khóa lui;n đã tìm hiJu và trình bày phương pháp sinh bat bi�n vòng li;p cũng như cách th11c k�t hQp bat bi�n v6i thijc thi tưQng trưng đJ kiJm ch11ng chương trình C/C++. Đ6ng thai, công cv VTSEinv, phiên ban cai ti�n cua VTSE đã đưQc cài đi;t đJ tăng kha năng kiJm tra các chương trình có ch11a vòng li;p. VTSEinv bao g6m các thành ph&n trlfu tưQng hóa như trư6c đây tích hQp v6i công cv con InvaGen đJ sinh bat bi�n vòng li;p. Chi�n lưQc sinh bat bi�n đưQc nghiên c11u và cài đi;t là b6 đe Farkas k�t hQp v6i giai ràng bu<c.

V6i b< thijc nghi�m *Loop-acceleration*, VTSEinv đã cho k�t qua kha quan khi so sánh v6i hai công cv tham gia thi SV-COMP có chi�n lưQc xu lý vòng li;p khác là CBMC và VeriAbs. So v6i CBMC, VTSEinv giai đưQc g&n gap đôi s6 bài toán trong thai gian nhanh hơn rat nhieu. Không chi vi;y, phương pháp dùng bat bi�n đã khic phvc đưQc nhưQc điJm sinh ra k�t qua *unsound* do vi�c đánh giá xap xi dư6i cua CBMC. So v6i VeriAbs, công cv cũng áp dvng bat bi�n vào m<t ph&n cua quá trình xu lý vòng li;p thì VTSEinv giai đưQc ít hơn hai bài toán, tuy nhiên van chi�m ưu th� ve t6c đ< thijc thi.

Trong tương lai, đJ cai thi�n k�t qua cua VTSEinv thì vi�c tìm ki�m hoi;c thit chi;t bat bi�n vòng li;p là rat quan tr9ng. Hai phương pháp sinh bat bi�n đang n6i bi;t hi�n nay đã đưQc tìm hiJu và trình bày trong Chương [4](#_bookmark56). Đ6ng thai, VTSE cũng có thJ md r<ng kha năng 11ng dvng b�ng cách t6i thiJu hóa các ràng bu<c cua chương trình, ti�p tvc phát triJn đJ xu lý các tính chat khác cua mã ngu6n C/C++ như dfi li�u mang, dfi li�u v6i b< nh6 đ<ng, . . .

# Tài li�u tham khao

1. Beyer, D., Keremoglu, M.E. (2011) *CPAchecker: a tool for configurable software verifi-cation.* In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp.184–190. Springer, Heidelberg.
2. Chimdyalwar, B., Kumar, S., Shrotri, U. (2013) Precise range analysis on large industrycode. In: Proceedings of the 2013 9th Joint Meeting on Foundations of SoftwareEngineering, pp. 675–678. ACM
3. Cousot, P., Halbwachs, N. (1978) *Automatic Discovery of Linear Restraints among Variables of a Program.* In: POPL, ACM 84–96
4. Clarke E.M., Henzinger T.A., Veith H. (2018) *Introduction to Model Checking*. In: Clarke E., Henzinger T., Veith H., Bloem R. (eds) Handbook of Model Checking.

Springer, Cham

1. Darke P. et al. (2018) *VeriAbs: Verification by Abstraction and Test Generation.* In: Beyer D., Huisman M. (eds) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2018. Lecture Notes in Computer Science, vol 10806.

Springer, Cham

1. E en, N., Soensson, N. (2004) *An extensible SAT-solver*. In: Giunchiglia, E., Tac- chella,A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg
2. Furia, C. A., Meyer, B., and Velder, S. (2012). *Loop invariants: analysis, classifi- cation, and examples*. ACM Comput. Surv. V, N, Article A ( YYYY), 51 pages.
3. Gupta, A., Rybalchenko, A. (2009) *Invgen: An efficient invariant generator.* In:

Computer Aided Verification, Springer 634–640

1. Henzinger, T., Jhala, R., Majumdar, R., Sutre, G. (2003) *Software verification with BLAST*. In: International conference on Model checking software. 235–239
2. King, J.C. (1976) *Symbolic execution and program testing*. Commun. ACM 19(7), 385–394
3. Kroening, Daniel & Tautschnig, Michael. (2014). CBMC – C Bounded Model Checker. 8413. 389-391. 10.1007/978-3-642-54862-8\_26.
4. Li, Jiaying & Sun, Jun & Li, Li & Le, Quang Loc & Lin, Shang-Wei. (2017). *Automatic loop-invariant generation and refinement through selective sampling*. 782-792. 10.1109/ASE.2017.8115689.
5. McMillan, K. (2003) *Interpolation and sat-based model checking*. In: Computer Aided Verification, Springer 1–13
6. Michael Colon, Sriram Sankaranarayanan, and Henny Sipma (2003) *Linear In- variant Generation using Non-linear Constraint Solving*, in Computer-aided Veri- fication (CAV).
7. N Thi Van Anh, T Van Khanh, N Thi Thuy (2018) *VTSE – Verifi- cation Tool based on Symbolic Execution*. In: VNU-UET Technical Report <http://eprints.uet.vnu.edu.vn/eprints/3053/>
8. Păsăreanu, C.S. & Visser, W. Int J, (2009) *A survey of new trends in symbolic execution for software testing and analysis*, Software Tools Technology Transfer 11: 339. https://doi.org/10.1007/s10009-009-0118-1
9. Shang-Wei Lin, Jun Sun, Hao Xiao, Yang Liu, David Sanán, Henri Hansen (2017) *FiB: squeezing loop invariants by interpolation between Forward/Backward predi- cate transformers.* ASE 2017: 793-803