دانشگاه صنعتی امیر کبیر
(پلی تکنیک تهران)

بسم الله الرحمن الرحیم

طراحی و پیاده‌سازی ابزاری به منظور اِعمال خط مشی امنیتیِ عدم تداخل مبتنی بر روش بازنویسی برنامه

سید محمدمهدی احمدپناه
smahmadpanah@aut.ac.ir

استاد راهنما: دکتر مهران سلیمان‌فلاح

دانشگاه صنعتی امیرکبیر
۲۵ مهر ۱۳۹۴

دانشکده مهندسی کامپیوتر
و فناوری اطلاعات

فهرست

- مقدمه
- خط مشی امنیتی عدم تداخل و اِعمال آن
- زبان WL و گراف وابستگی برنامه
- الگوریتم بازنویسی برنامه
- پیاده‌سازی و ایجاد رابط کاربری
- آزمون نرم‌افزار
- جمع‌بندی و کارهای آینده

مقدمه

- امنیت زبان‌مبنا [۱]
  - تحلیل ایستا
    - تحلیل جریان داده، تحلیل نوع‌مبنا، وارسی مدل، تفسیر انتزاعی
  - تحلیل پویا
    - نظارت اجرا

- تعریف مسئله
  - اعمال یک نیازمندی امنیتی به کد مبدأ برنامه

خط مشی امنیتی عدم تداخل

- خط مشی امنیتی
  - عدم تداخل

  - دسته‌بندی بر اساس وضعیت پیشرفت برنامه:
    - حساس به پیشرفت
    - غیرحساس به پیشرفت

مروری بر کارهای گذشته

- ارائه یک نوع‌سامانه [۲]
  - کارهای اولیه
  - محافظه‌کارانه
- یک روش تبدیل برنامه، ترکیبی از تحلیل پویا و ایستا [۶]
  - عدم توجه به رفتار خاتمه برنامه‌ها
  - عدم تشخیص جریان‌های ضمنی بدون انتساب
- چارچوبی برای ناظرهای امنیتی پویای درون‌برنامه‌ای [۷]
- طراحی یک ماشین به کمک رخدادهای انتزاعی زمان اجرا و ویرایش اجرا توسط اطلاعات ایستا [۸]

اِعمال عدم تداخل

- هیچ روش کاملاً پویایی برای اعمال عدم تداخلِ حساس به جریان وجود ندارد. [۹]
- مسئله تشخیص برنامه‌هایی که عدم تداخل را برآورده می‌کنند:
  - تصمیم‌پذیر نیست. [۲]
    - توسط روش‌های ایستا قابل اعمال نیست.
  - مکمل شمارش‌پذیر بازگشتی نیست. [۱۰]
    - توسط ناظرهای اجرا قابل اعمال نیست.
- **روش بازنویسی برنامه** [۱]

زبان WL

program ::= **program ;** clist

clist ::= c | clist **;** c

exp ::= b | n | x | exp == exp | exp < exp | exp <= exp | exp >= exp | exp > exp

| exp + exp | exp - exp | exp **or** exp | exp **and** exp | **!** exp

c ::= **NOP** | x = exp | **inL** varlist | **inH** varlist | **outL** x | **outH** x | **outL BOT**

| **outH BOT**

| **if** exp **then** clist **endif** | **if** exp **then** clist **else** clist **endif** | **while** exp **do** clist **done**

varlist ::= x | x **,** varlist

b ::= **true** | **false** | **TRUE** | **FALSE**

n ::= integer_number

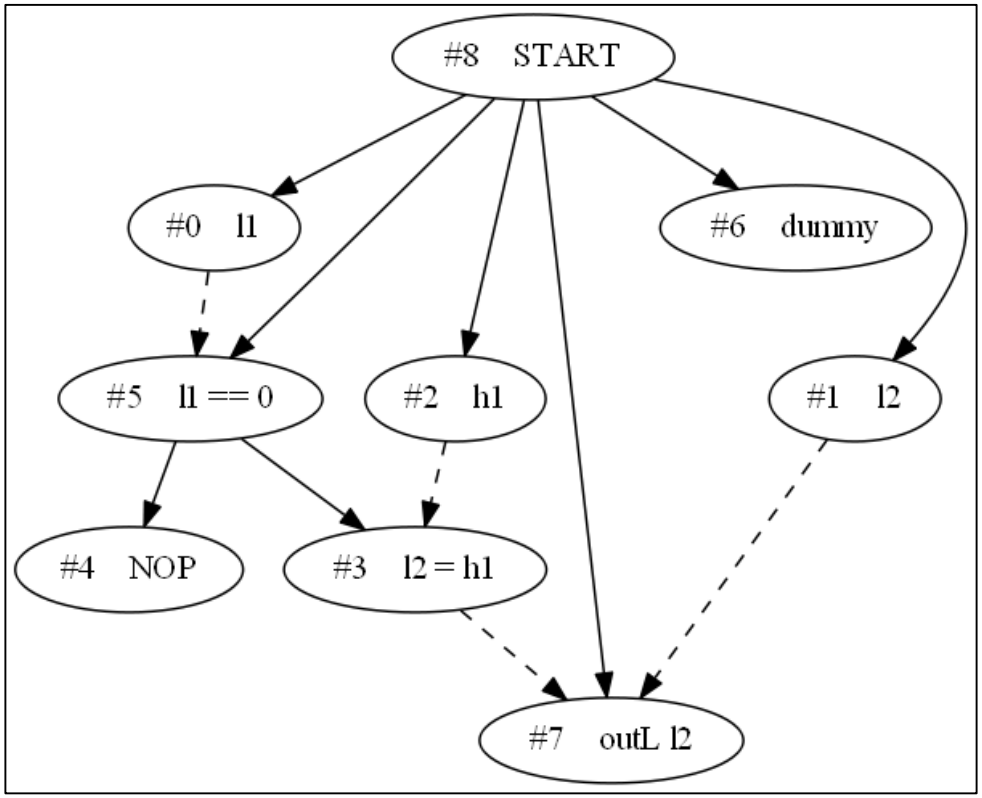
x ::= identifier

شکل ۱ – نحو زبان WL

گراف وابستگی برنامه

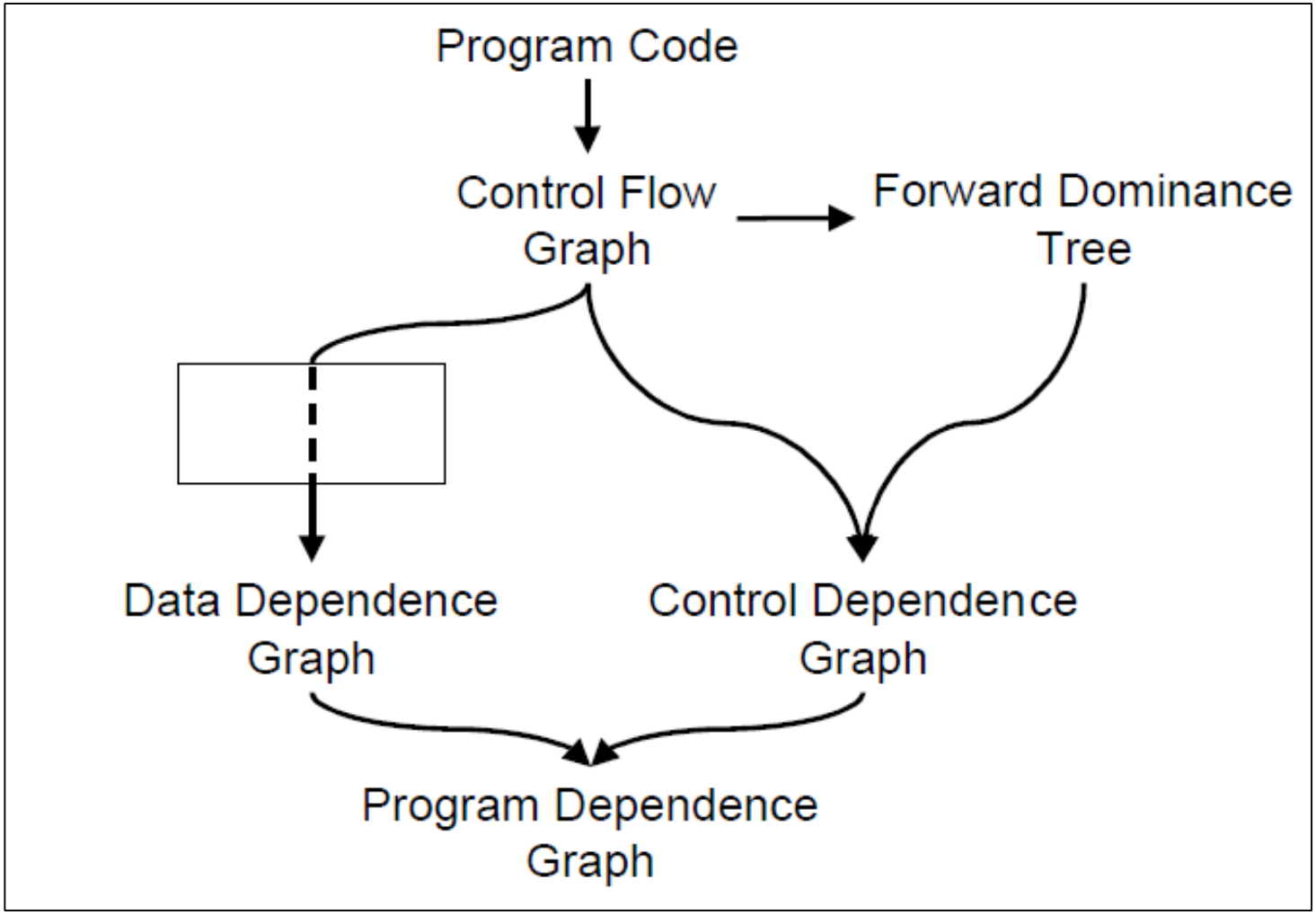
- تشخیص جریان‌های اطلاعات از ورودی‌های سطح بالا به خروجی‌های سطح پایین

```
program;
inL l1, l2;
inH h1;
if l1 == 0 then
        l2 = h1
else
        NOP
endif;
outL l2
```



شکل ۲ – برنامه‌ای به زبان WL و گراف وابستگی برنامه آن

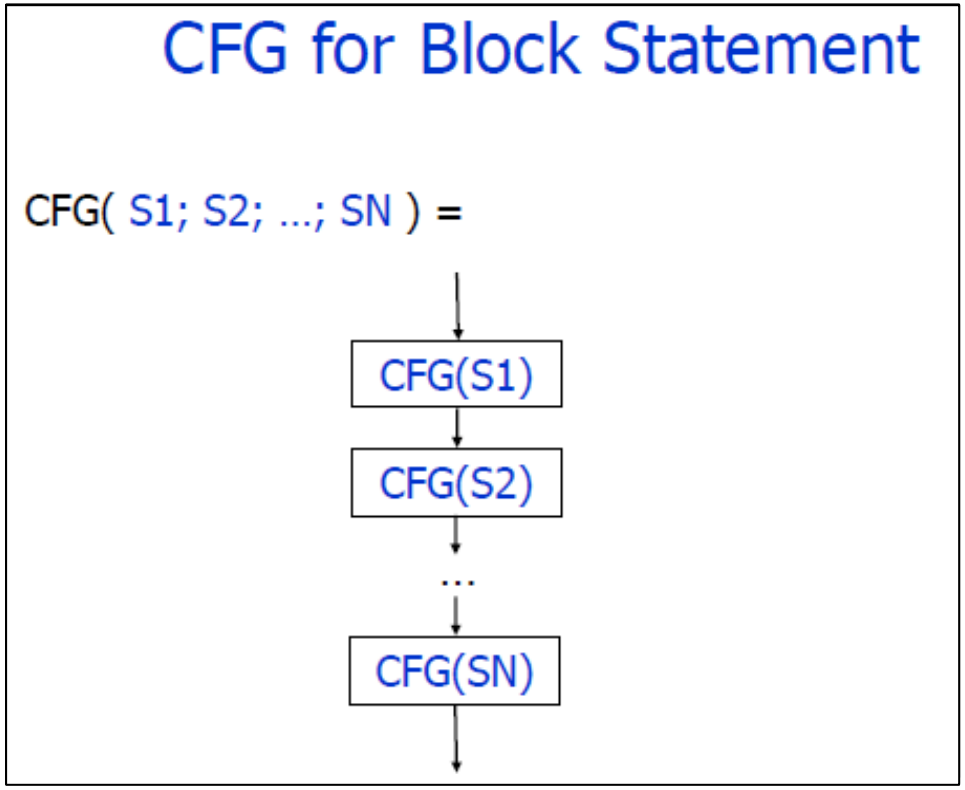
گراف وابستگی برنامه



شکل ۳ – نمودار کلی نحوه تولید گراف وابستگی برنامه از روی کد مبدأ برنامه [۱۵]

گراف جریان کنترل

- بلوک پایه
- تولید در هنگام تشکیل درخت تجزیه



شکل ۴ – نحوه تولید گراف جریان کنترل [۱۶]

گراف وابستگی کنترلی

- پس‌غلبه

- مرزهای پس‌غلبه
  - نزدیک‌ترین نقاط انشعابی که به گره الف منجر می‌شوند.

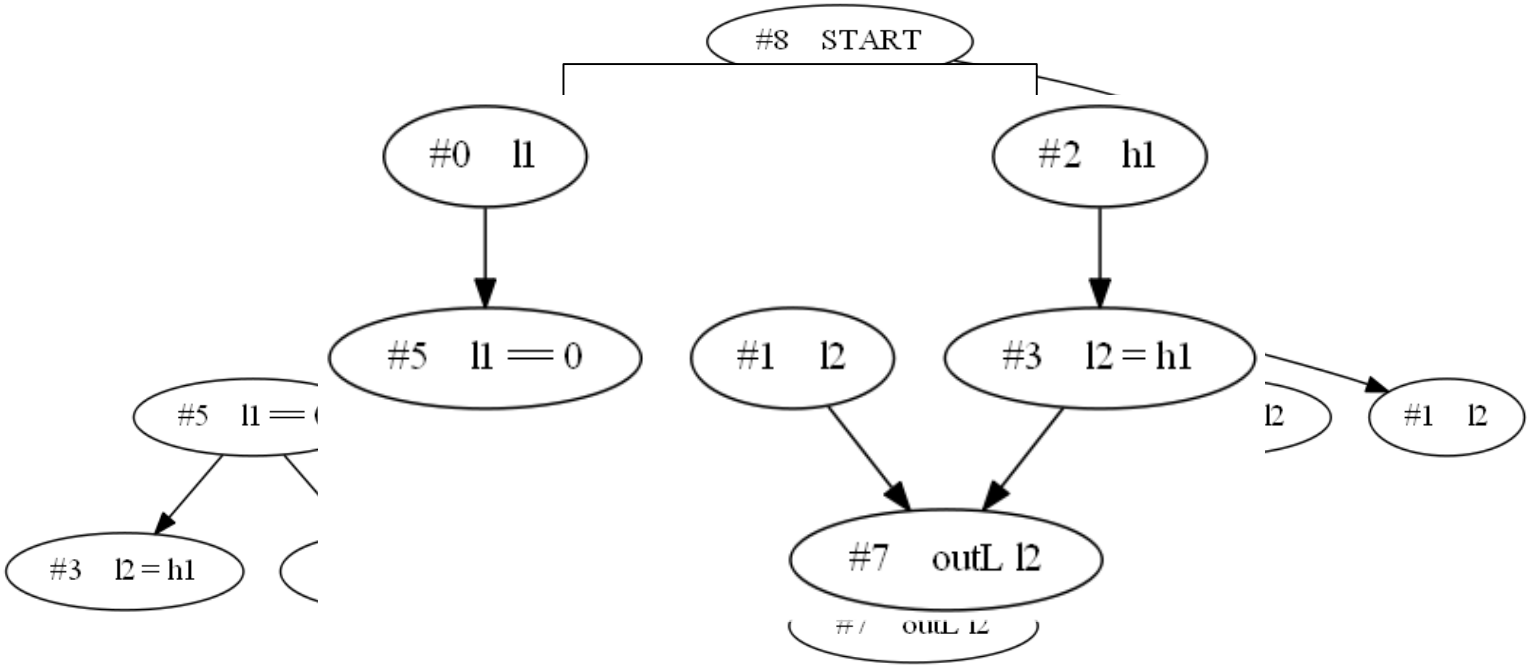
- وابستگی کنترلی

گراف وابستگی داده‌ای

- وابستگی داده‌ای
  - فقط نزدیک‌ترین گزاره‌ای که آن متغیر در آن مقداردهی شده است، وابستگی را خواهد داشت.

گراف وابستگی برنامه

- ترکیب گراف وابستگی کنترلی و داده‌ای
  - یال وابستگی کنترلی: خط ساده
  - یال وابستگی داده‌ای: خط‌چین



شکل ۲ – برنامه‌ای به زبان WL و گراف وابستگی برنامه آن

الگوریتم بازنویسی برنامه

- الگوریتم کلی
  - تابع affect

foreach statement X producing a high input event h_{in} **do**
  foreach statement Y producing a low observable event e_L **do**
    if $Y \in affect(X)$ **then**
      transform Y into Y′ such that
      $Y' \notin affect(X)$ in the new program
    end
  end
end

شکل ۷ – الگوریتم کلی بازنویسی برای اعمال خط مشی عدم تداخل [۵]

بازنویسی برای حالت غیر حساس به پیشرفت

- $inH\ h \rightsquigarrow \mathrm{outL}\ \mathrm{l}$
  - جایگزین کردن دستورات outL l متأثر از ورودی‌های سطح بالا با دستورات $\perp$ outL یا NOP
  - استفاده از شرط‌های مسیر
    - تعریف شرط مسیر
    - حاصل ترکیب عطفی شرط‌های اجرای گره‌های مسیر
      - تعریف شرط اجرای گره
  - جریان صریح و ضمنی

بازنویسی برای حالت غیر حساس به پیشرفت

RW_{PINI} (M, G):
Initialize F to the set of all paths $Start \hookrightarrow P \rightsquigarrow P'$ in the PDG *G* of *M* where *P* is the node representing a high input and P' is the node representing outL l for some l;
if $F = \emptyset$ then
    return M;
end
create a copy of M, name it M′, and change it as follows:
determine the type of flow indicated by each path $f \in F$;
foreach $f \in F$ do:
    Generate the path condition of f as the conjunction of the execution conditions of node N satisfying $f = Start \rightsquigarrow X \xrightarrow{d} N \rightsquigarrow P'$ if there are such nodes on the path and true otherwise;
end
foreach node n on G representing outL l for some l **do**
    let c be the disjunction of the path conditions of all $f' \in F$ which terminate at n;
    if all paths $f' \in F$ terminating at n indicate an explicit flow **then**
        replace outL l with the statement "if c then outL $\perp$ else outL l endif";
    else
        replace outL l with the statement "if c then NOP else outL l endif";
    end
end
return M′;

شکل ۸ – الگوریتم بازنویسی عدم تداخل حالت غیر حساس به پیشرفت [۵]

بازنویسی برای حالت حساس به پیشرفت

- وضعیت پیشرفت برنامه
  - خاتمه
  - واگرایی
- ساختار While در زبان WL
  - تحلیل حلقه‌های وابسته به مقدار سطح بالا
    - در حالت کلی، تصمیم‌ناپذیر

```
program;
inH h1;
inL l1;
while h1 < l1 do
        NOP;
        h1 = h1 - l1
done;
outL l1
```

Loop Analysis → “h1 >= l1 or l1 < 0”

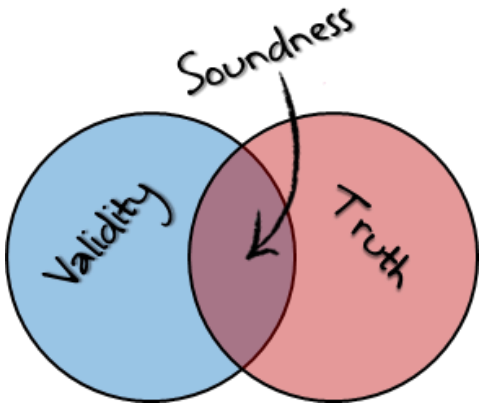
بازنویسی برای حالت حساس به پیشرفت

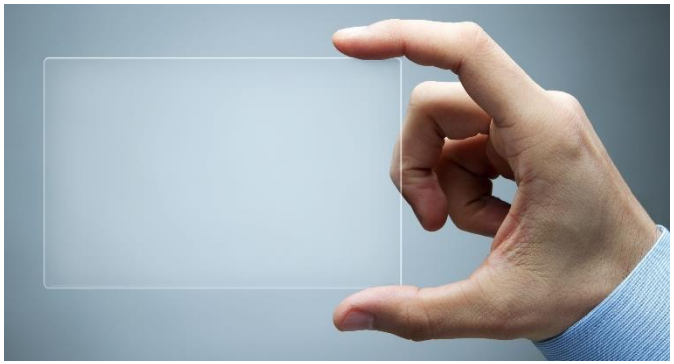
$RW_{PSNI}(M, G)$:
Initialize D to the set of all paths $Start \hookrightarrow P \hookrightarrow E^+$ in G where E^+ is a path terminating at a loop guard and P is the node representing a high input;
$M' = RW_{PINI}(M, G)$;
if $D = \emptyset$ then
    return M′;
end
H = max {*height*(n) | n is a node on G}, where *height* is a function that returns the height of a given node on the tree obtained by removing data dependence edges from G;
Change M′ as follows:
for $h = H$ to 1 **do**
    foreach node n with *height(n)* = h representing a loop on some path $f \in D$ **do**
        r = LoopAnalyzer(loop(n));
        if r = False **then**
            if $X \xrightarrow{c} n$ appears on at least one path $f \in D$ **do**
                replace *loop*(n) with the statement "if *guard*(n) then *body*(n) endif";
            end
        else
            if r ≠ True then
                replace *loop*(n) with the statement "if r then *loop*(n) endif";
            end
        end
end
h = h – 1;
end
return M′;

شکل ۹ – الگوریتم بازنویسی عدم تداخل حالت حساس به پیشرفت [۵]

الگوریتم‌های بازنویسی برنامه

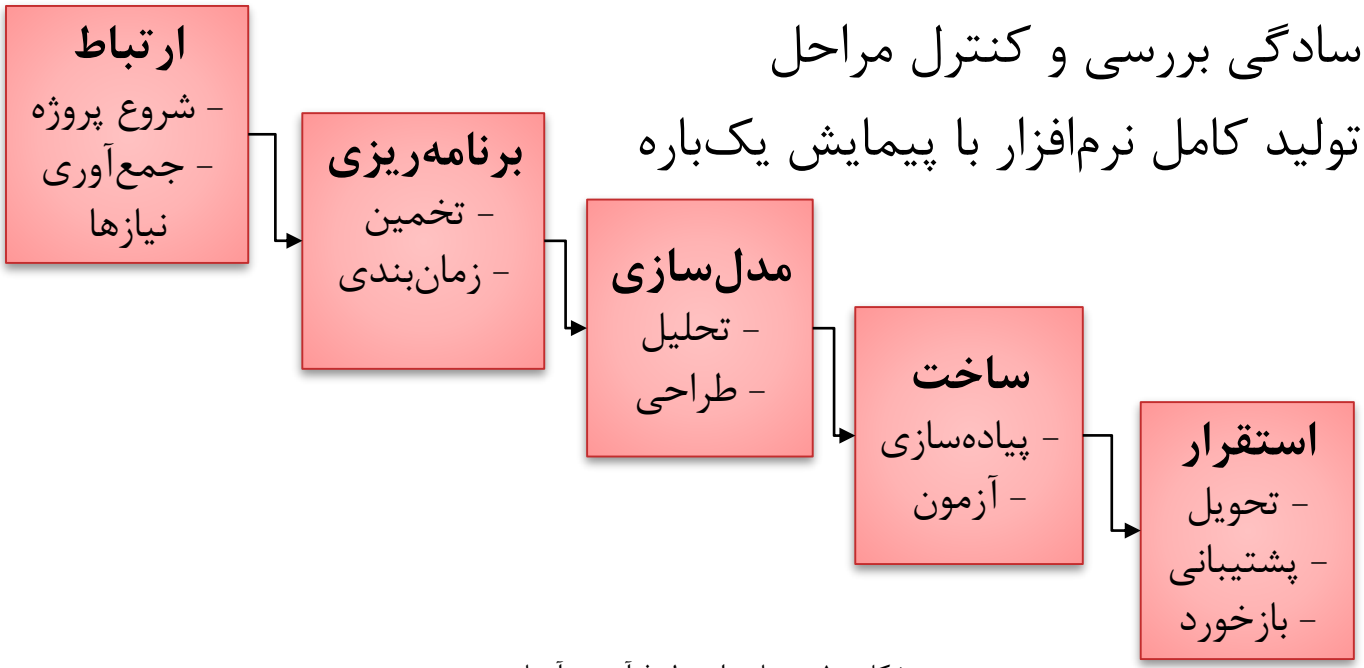
- صحت و شفافیت الگوریتم‌های مورد استفاده اثبات شده است. [۵]
- امکان تغییر معناشناخت برنامه، پس از بازنویسی در حالت حساس به پیشرفت
  - تبدیل حلقه به گزاره شرطی





تحلیل و طراحی نرم‌افزار

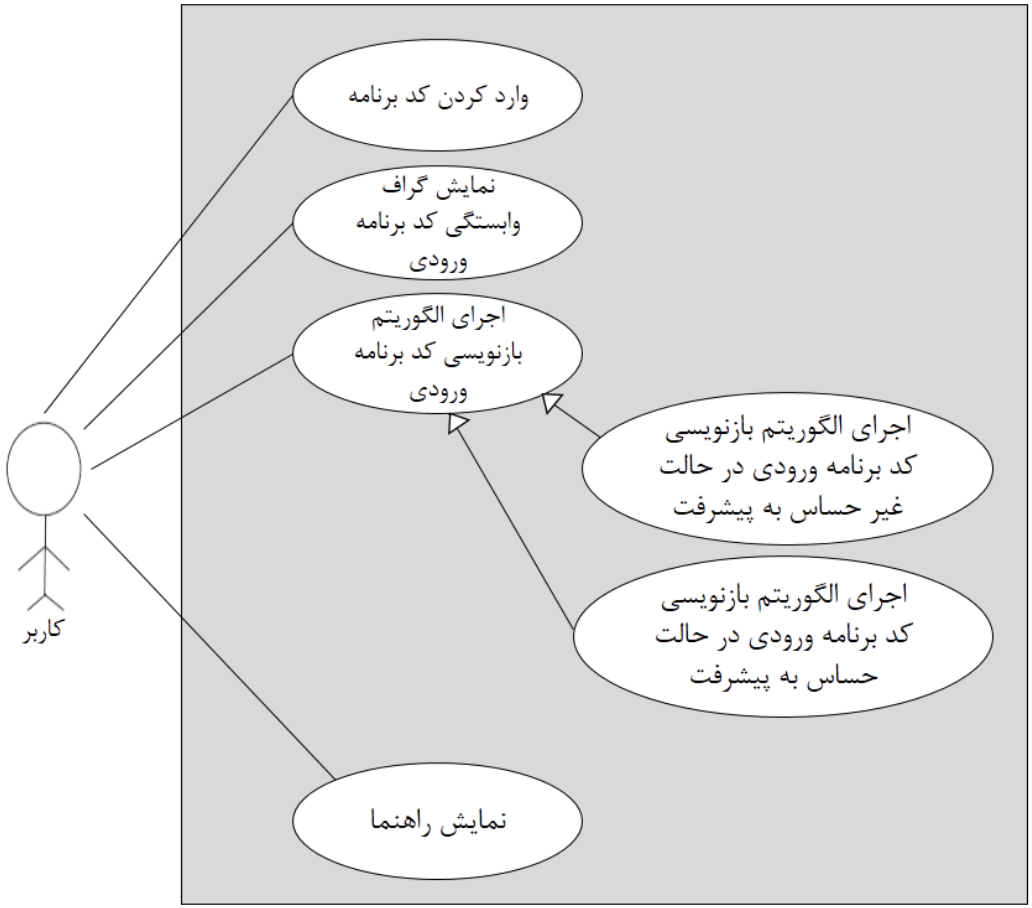
- مدل فرآیندی آبشاری
  - ثابت، مشخص و پایدار بودن نیازها
  - تولید مستندات آسان‌تر
  - سادگی استفاده و فهم
  - سادگی بررسی و کنترل مراحل
  - تولید کامل نرم‌افزار با پیمایش یک‌باره



شکل ۱۰ – نمایی از مدل فرآیندی آبشاری

تحلیل و طراحی نرم‌افزار

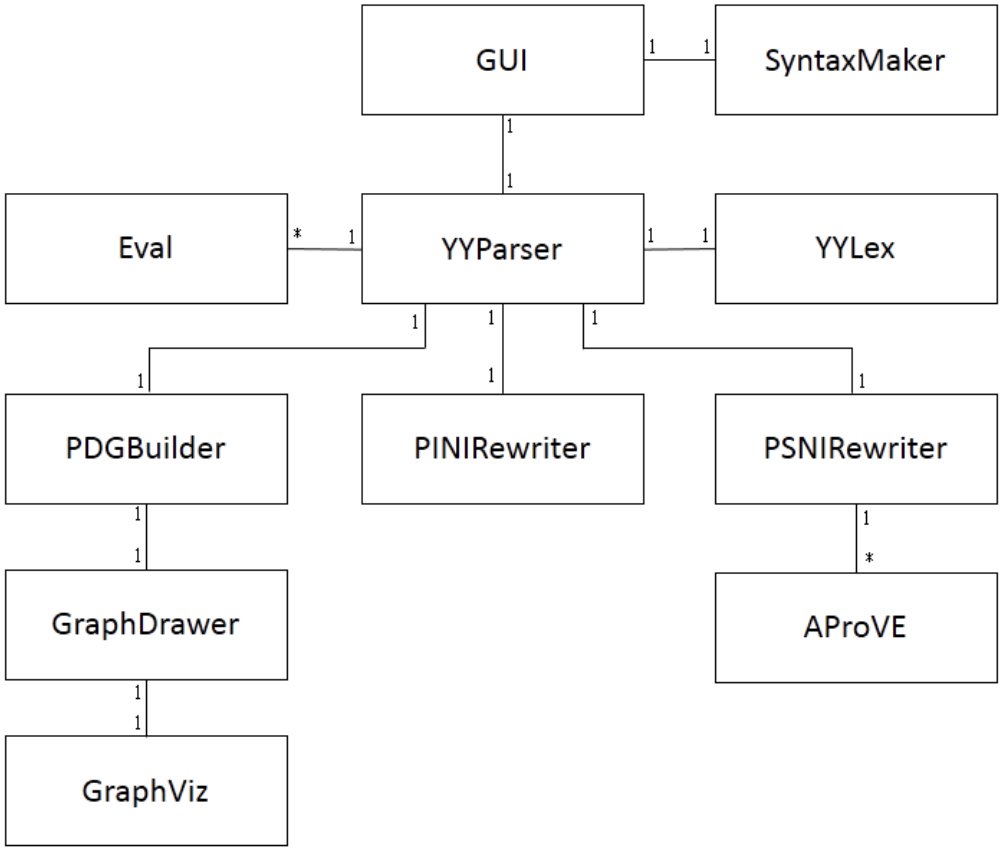
- نمودار درخواست سیستم



شکل ۱۱ – نمودار درخواست سیستم نرم‌افزار

تحلیل و طراحی نرم‌افزار

- نمودار کلاس

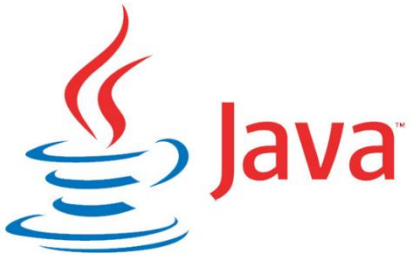


شکل ۱۲ – نمودار کلاس نرم‌افزار (بدون ذکر فیلدها و متدها)

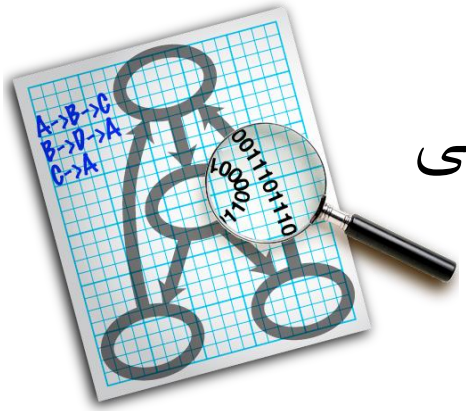
تحلیل و طراحی نرم‌افزار

- فازهای پیاده‌سازی
  - تحلیل‌گر لغوی و نحوی زبان WL
  - گراف جریان کنترل، درخت غلبه رو به جلو، گراف‌های وابستگی کنترلی، وابستگی داده‌ای و وابستگی برنامه
  - بازنویس برنامه برای حالت غیر حساس به پیشرفت
  - تحلیل‌گر حلقه و بازنویس برنامه برای حالت حساس به پیشرفت

شرح کلی پیاده‌سازی و ابزارهای مورد استفاده



- پیاده‌سازی به زبان جاوا
- تحلیل‌گر لغوی و نحوی زبان
  - ابزارهای JFlex و Bison
  - تغییر دستی کدهای تولید شده توسط ابزارها
  - تولید کدهای متناظر برنامه‌ها به زبان C



- نمایش گرافیکی گراف‌های وابستگی
  - ابزار GraphViz و زبان dot

شرح کلی پیاده‌سازی و ابزارهای مورد استفاده

- تحلیل‌گر حلقه

APROVE

- ویرایش‌گر پیشرفته کد مبدأ
  - کتابخانه RSyntaxTextArea و ابزار TokenMakerMaker

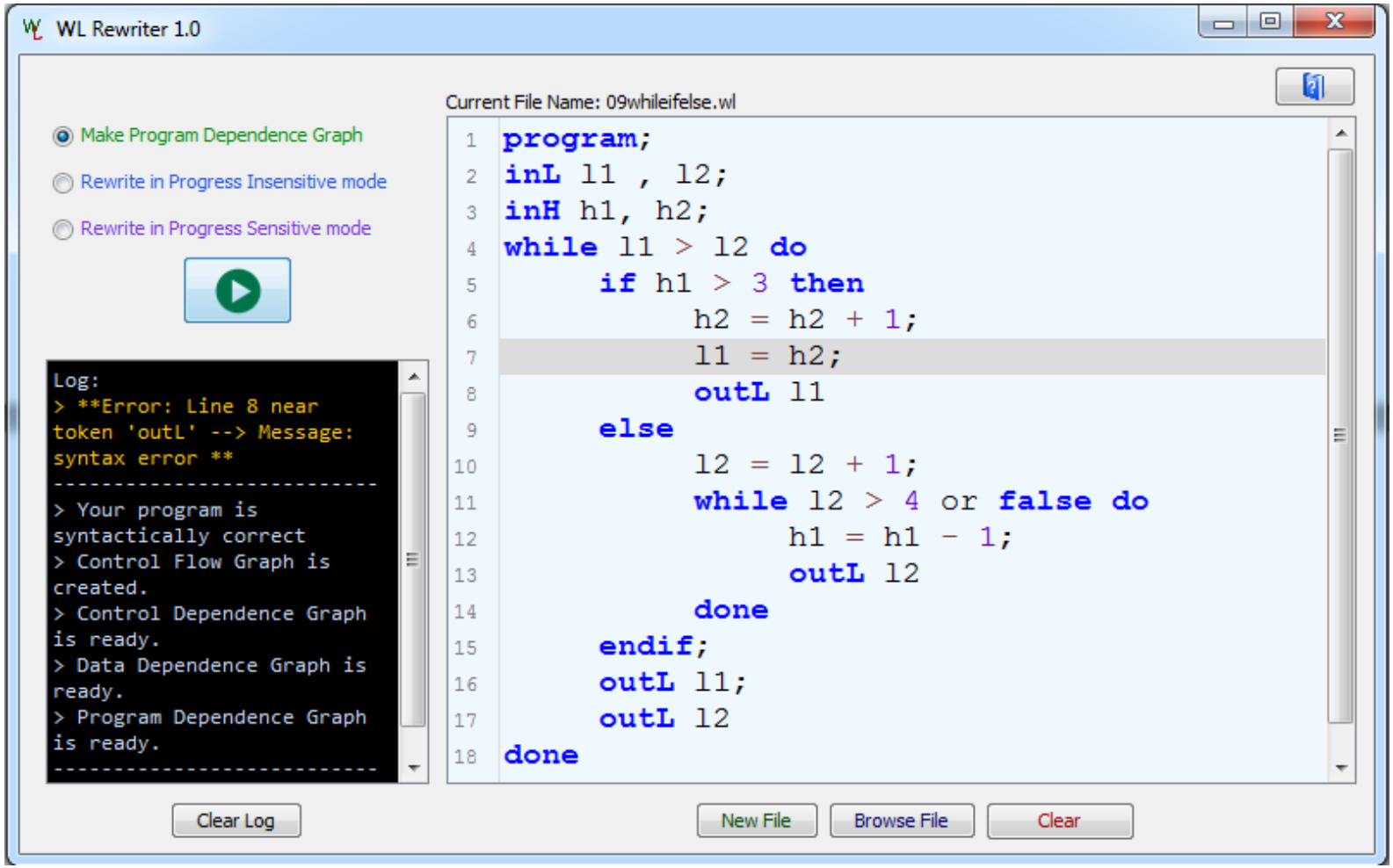
ایجاد رابط کاربری گرافیکی

- به حداقل رساندن پیچیدگی‌ها، با حفظ قابلیت‌های برنامه
- طراحی گرافیکی اولیه
- نظرسنجی و گرفتن بازخورد از تعدادی از برنامه‌نویسان به عنوان کاربران ابزار

ایجاد رابط کاربری گرافیکی

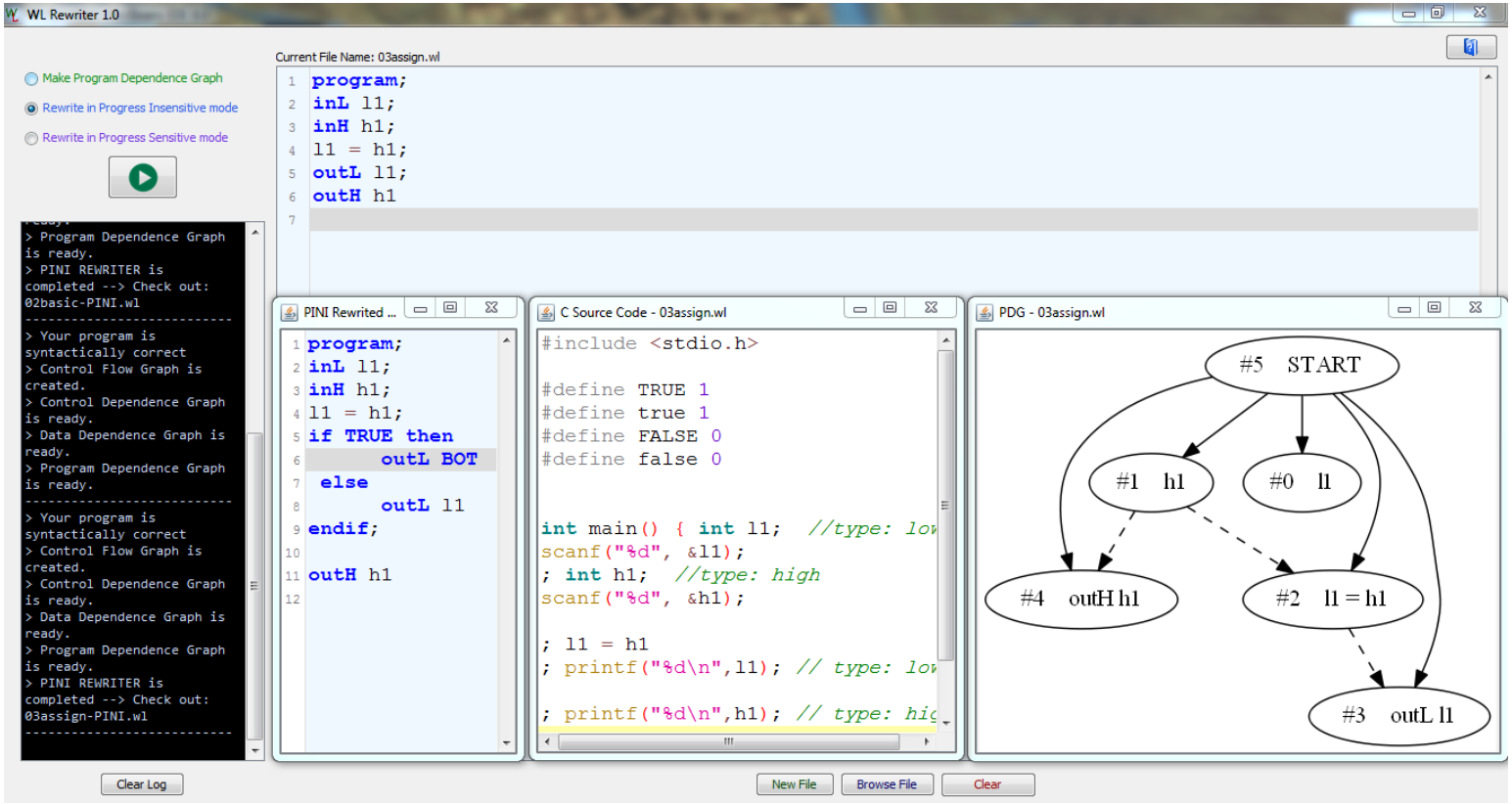
- برخی از نکات مورد استفاده در هنگام طراحی رابط کاربری
  - همگونی و سبک یکسان گزینه‌ها و دکمه‌ها
  - بازخورد مناسب برنامه به تغییرات و درخواست‌های کاربر
  - گروه‌بندی منطقی اجزای مرتبط صفحه
  - استفاده از رنگ‌ها و قالب‌های با معنا و متفاوت
  - ایجاد میان‌برها و یادمان‌ها
  - سازگار با تغییرات ابعاد صفحه

ایجاد رابط کاربری گرافیکی



شکل ۱۳– نمایی از رابط کاربری گرافیکی نرم‌افزار

ایجاد رابط کاربری گرافیکی



شکل ۱۴– نمایی از رابط کاربری گرافیکی نرم‌افزار

راستی‌آزمایی و آزمون نرم‌افزار

- استفاده از روش آزمون دامنه
  - طراحی و تولید موارد آزمون
    - ساختارهای مختلف زبان WL
    - حالت‌های مختلف نقض خط مشی عدم تداخل
    - جریان‌های صریح و ضمنی

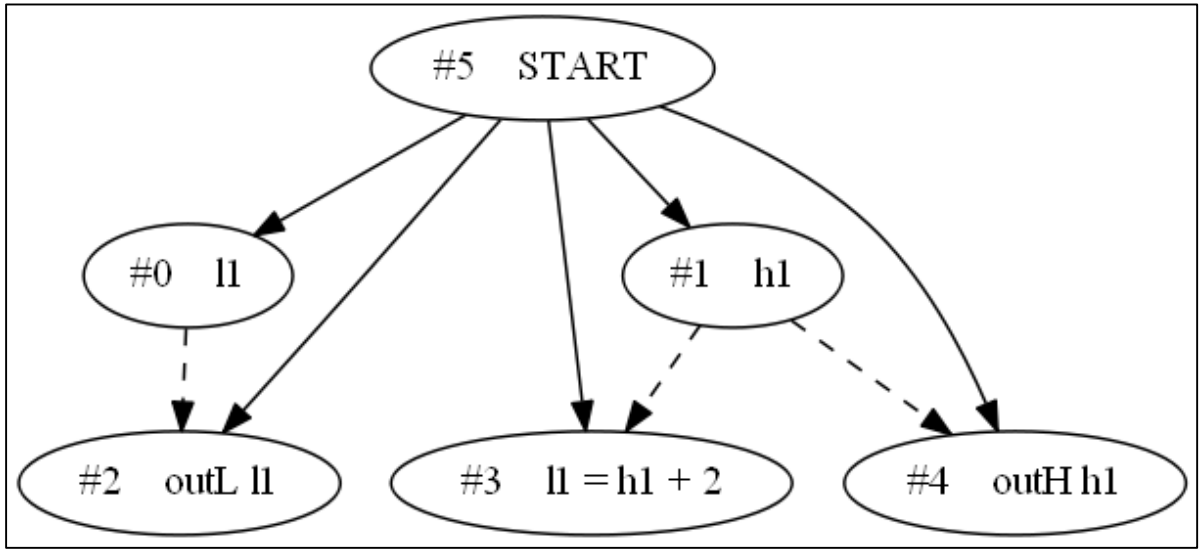
راستی‌آزمایی و آزمون نرم‌افزار

02basic.wl

```
program;
inL l1;
inH h1;
outL l1;
l1 = h1 + 2;
outH h1
```

PINI

```
program;
inL l1;
inH h1;
outL l1;
l1 = h1 + 2;
outH h1
```

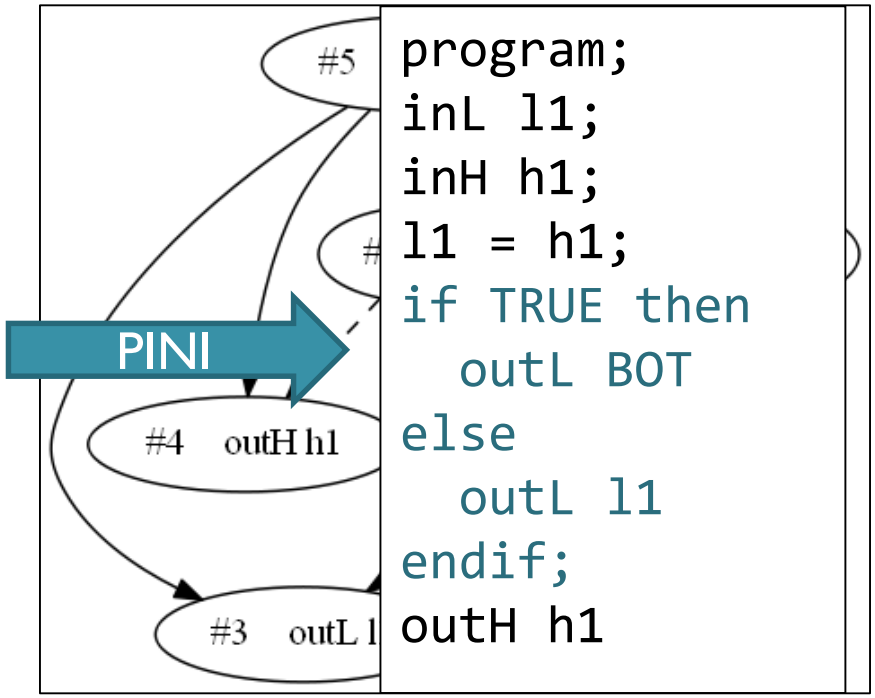


شکل ۱۵– مورد آزمون اول

راستی‌آزمایی و آزمون نرم‌افزار

03assign.wl

```
program;
inL l1;
inH h1;
l1 = h1;
outL l1;
outH h1
```



شکل ۱۶– مورد آزمون دوم

راستی‌آزمایی و آزمون نرم‌افزار

07Ifelseadvanced.wl

```
program;
inL l1 , l2;
inH h1 , h2;
if !(l1 == 0) then
    l1 = 2 + 4 + l1;
    outL l1;
    if h1 > 6 then
       l1 = 6;
       outL l1;
       outH h1
    endif
else
    if l2 > 3 then
       l1 = l1 + 1;
       outL l1;
       outH h2
    else
       l2 = 2 + h2;
       outL l2;
       outL l1
     endif
endif;
outL l1;
outL l2
```

PINI

```
program;
inL l1, l2; inH h1, h2;
if !((l1 == 0)) then
   l1 = 2 + 4 + l1;  outL l1;
   if h1 > 6 then
       l1 = 6;
       if ((!(l1 == 0)) or (!(l1 == 0) and (h1
> 6) and (!(l1 == 0))))  then
           NOP
       else outL l1
       endif;
       outH h1 endif
else
   if l2 > 3 then
       l1 = l1 + 1; outL l1; outH h2
   else
       l2 = 2 + h2;
       if (!(l2 > 3) and !(!(l1 == 0)))  then
           outL BOT else outL l2  endif;
       outL l1
   endif
endif;
if (!(l1 == 0))  then NOP else outL l1 endif;
if (!(l2 > 3) and !(!((l1 == 0))))  then
   outL BOT
else outL l2
endif
```

شکل ۱۷– مورد آزمون سوم

راستی‌آزمایی و آزمون نرم‌افزار

l1	l2	h1	h2	outL l1 (line# 6)	outL l1 (line# 9)	outH h1 (line# 10)	outL l1 (line# 15)	outH h2 (line# 16)	outL l2 (line# 19)	outL l1 (line# 20)	outL l1 (line# 23)	outL l2 (line# 24)	Violation
0	4	0	0	-	-	-	1	0	-	-	1	4	No
0	4	1	1	-	-	-	1	1	-	-	1	4	
0	2	0	0	-	-	-	-	-	2	0	0	2	Yes
0	2	1	1	-	-	-	-	-	3	0	0	3	
1	0	7	1	7	6	7	-	-	-	-	6	0	Yes
1	0	6	1	7	-	-	-	-	-	-	7	0	

جدول ۱ – نمونه ورودی‌ها و خروجی‌های برنامه مورد آزمون سوم

راستی‌آزمایی و آزمون نرم‌افزار

l1	l2	h1	h2	outL l1 (line# 6)	outL l1 (line# 9)	outH h1 (line# 10)	outL l1 (line# 15)	outH h2 (line# 16)	outL l2 (line# 19)	outL l1 (line # 20)	outL l1 (line# 23)	outL l2 (line# 24)	Violation
0	4	0	0	-	-	-	1	0	-	-	-	4	No
0	4	1	1	-	-	-	1	1	-	-	-	4	
0	2	0	0	-	-	-	-	-	BOT	0	0	BOT	No
0	2	1	1	-	-	-	-	-	BOT	0	0	BOT	
1	0	7	1	7	-	7	-	-	-	-	-	0	No
1	0	6	1	7	-	-	-	-	-	-	-	0	

جدول ۲ – نمونه ورودی‌ها و خروجی‌های برنامه بازنویسی‌شده حالت غیرحساس به پیشرفت مورد آزمون سوم

راستی‌آزمایی و آزمون نرم‌افزار

l l whilewhileconcat.wl

```
program;
inL l1;
inH h1 , h2;
while l1 > 0 do
   l1 = h2 + l1
done;
while h1 > l1 do
   l1 = l1 + 3;
   outL l1
done;
outL l1;
outH h1
```

شکل ۱۸– مورد آزمون چهارم

PSNI

```
program;
inL l1;
inH h1, h2;
if h2 < 0 then
    while l1 > 0 do
        l1 = h2 + l1
    done
endif;
while h1 > l1 do
    l1 = l1 + 3;
    if TRUE then NOP
    else outL l1
    endif
done;
if TRUE then NOP
else outL l1
endif;
outH h1
```

راستی‌آزمایی و آزمون نرم‌افزار

l1	h1	h2	outL l1 (line# 9)	outL l1 (line# 11)	outH h1 (line# 12)	Violation
0	1	0	3	3	1	Yes
0	5	1	3,6	6	5	
1	0	-2	2	2	0	Yes
1	5	-2	2,5	5	5	
1	1	0	diverge			Yes
1	1	-2	2	2	1	

جدول ۳ – نمونه ورودی‌ها و خروجی‌های برنامه مورد آزمون سوم

راستی‌آزمایی و آزمون نرم‌افزار

l1	h1	h2	outL l1 (line# 9)	outL l1 (line# 11)	outH h1 (line# 12)	Violation
0	1	0	-	-	1	No
0	5	1	-	-	5	
1	0	-2	-	-	0	No
1	5	-2	-	-	5	
1	1	0	diverge			No
1	1	-2	-	-	1	

جدول ۴ – نمونه ورودی‌ها و خروجی‌های برنامه بازنویسی‌شده حالت غیرحساس به پیشرفت مورد آزمون چهارم

راستی‌آزمایی و آزمون نرم‌افزار

l1	h1	h2	outL l1 (line# 9)	outL l1 (line# 11)	outH h1 (line# 12)	Violation
0	1	0	-	-	1	No
0	5	1	-	-	5	
1	0	-2	-	-	0	No
1	5	-2	-	-	5	
1	1	0	-	-	1	No
1	1	-2	-	-	1	

جدول ۵ – نمونه ورودی‌ها و خروجی‌های برنامه بازنویسی‌شده حالت حساس به پیشرفت مورد آزمون چهارم

جمع‌بندی

- خط مشی امنیتی عدم تداخل
  - حساس به پیشرفت
  - غیر حساس به پیشرفت
- زبان WL و بازنویسی برنامه
  - گراف وابستگی برنامه
  - الگوریتم‌های بازنویسی
    - صحت و شفافیت
    - پیاده‌سازی و آزمون ابزار

کارهای آینده

- زبان‌های برنامه‌نویسی پیشرفته‌تر و رایج
  - زبان‌های دارای ساختارهای پیچیده
    - پشتیبانی از تابع
    - کلاس و شیء
    - چندنخی
    - ویژگی‌های جدیدتر
- بهبود تحلیل‌گر حلقه
- بهبود ابزار پیاده‌سازی‌شده
  - سرعت
  - حافظه

منابع و مراجع

[۱] F.B. Schneider, J.G. Morrisett, and R. Harper, "A Language-Based Approach to Security", in *Informatics - 10 Years Back. 10 Years Ahead*, Springer-Verlag Berlin, Heidelberg, 2001, pp. 86-101.

[۲] D. Volpano and G. Smith, "A Type-Based Approach to Program Security", *TAPSOFT '97 Proceedings of the 7th International Joint Conference CAAP/FASE on Theory and Practice of Software Development*, 1997, pp. 607-621.

[۳] J.A. Goguen and J. Meseguer, "Security Policies and Security Models", in *Proceedings of IEEE Symposium on Security and Privacy*, Vol. 12, IEEE, 1982, pp. 11-18.

[۴] M.R. Clarkson and F.B. Schneider, "Hyperproperties", *Journal of Computer Security - 7th International Workshop on Issues in the Theory of Security (WITS'07)*, 2010, pp. 1157-1210.

[۵] A. Lamei and M. S. Fallah, "Rewriting-Based Enforcement of Noninterference in Programs with Observable Intermediate Values", submitted to *Journal of Universal Computer Science*, 2015.

[۶] V.N. Venkatakrishnan, W. Xu, D.C. DuVarney, and R. Sekar, "Provably Correct Runtime Enforcement of Non-interference Properties", in *Proceedings of the 8th International Conference on Information and Communications Security, ICICS'06*, Springer-Verlag Berlin, Heidelberg, 2006, pp. 332-351.

[۷] J. Magazinius, A. Russo, and A. Sabelfeld, "On-the-fly inlining of dynamic security monitors", *Computers and Security-Silver Linings in the Cloud*, 2012, pp. 827-843.

منابع و مراجع

[۸] G. Le Guernic, A. Banerjee, T. Jensen, and D.A. Schmidt, "Automata-based confidentiality monitoring", in *Proceedings of the 11th Asian computing science conference on Advances in computer science: secure software and related issues, ASIAN'06*, Vol. 4435, Springer-Verlag Berlin, Heidelberg, 2007, pp. 75-89.

[۹] A. Russo and A. Sabelfeld, "Dynamic vs. Static Flow-Sensitive Security Analysis", in *Proceedings of the 2010 23rd IEEE Computer Security Foundations Symposium, CSF '10*, IEEE, 2010, pp. 186-199.

[۱۰] G.M. Kevin W. Hamlen and F.B. Schneider, "Computability classes for enforcement mechanisms", *ACM Transactions on Programming Languages and Systems*, Vol. 28, 2006, pp. 175-205.

[۱۱] J. Ferrante, K.J. Ottenstein, and J.D. Warren, "The program dependence graph and its use in optimization", *ACM Transactions on Programing Languages and Systems*, Vol.9, 1987, pp. 319-349.

[۱۲] H. Mantel and H. Sudbrock, "Types vs. pdgs in information flow analysis", in *Logic-Based Program Synthesis and Transformation,* Springer, 2013, pp. 106-121.

[۱۳] "JFlex", Available: http://jflex.de/ [Sep. 10, 2015].

[۱۴] "Bison", Available: https://www.gnu.org/software/bison/ [Sep. 10, 2015].

منابع و مراجع

[۱۵] K. M. Anderson, Class Lecture, Topic: "Lecture 15: Control Dependence Graphs" CSCI 5828, University of Colorado at Boulder, Spring 2000, Available: http://www.cs.colorado.edu/~kena/classes/5828/s00/lectures/lecture15.pdf [Jul. 25 2015].

[۱۶] T. Teitelbaum, Class Lecture, Topic: "Lecture 24: Control Flow Graphs" Introduction to Compilers, Cornell University, 2008, http://www.cs.cornell.edu/courses/cs412/2008sp/lectures/lec24.pdf [Jul. 25 2015].

[۱۷] C. N. Fischer, Class Lecture, Topic: "The Program Dependence Graph: Control Flow and Control Dependences" S502 Compilers, Fall 2008, Available: http://pages.cs.wisc.edu/~fischer/cs701.f08/lectures/Lecture19.4up.pdf [Jul. 25 2015].

[۱۸] S. Moore, A. Askarov, and S. Chong, "Precise enforcement of progress-sensitive security", in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, ACM, 2012, pp. 881-893.

[۱۹] Roger S. Pressman, "Process Models" in *Software Engineering: A Practitioner's Approach*, 7th ed., Mc Graw-Hill Higher Education, 2010, pp. 39-41.

[۲۰] E. R. Gansner and S. C. North. "An Open Graph Visualization and Its Application to Software Engineering", *Software – Practice and Experience Journal*, vol. 30, No. 11, 2000, pp. 1203-1233, Available: www.graphviz.org [Aug. 12 2015].

[۲۱] "AProVE", Available: http://aprove.informatik.rwth-aachen.de/index.asp?subform=home.html [Aug. 25 2015].

[۲۲] "RSyntaxTextArea", Available: http://bobbylight.github.io/RSyntaxTextArea/ [Sep. 04 2015].

با سپاس از توجه شما! ☺