به نام خدا

تبدیل خودکار کد منبع سیستم عامل از زبان C به زبان مدلسازی Promela جهت استفاده برای تست و ارزیابی خودکار مدل

مرضیه علیدادی

فهرست مطالب

مقدمه صفحه 3

سیستم‌های مشابه صفحه 6

روش پیشنهادی صفحه 11

نتیجه‌گیری صفحه 23

مقدمه

ضرورت

- کاربرد سیستم‌ها در مسائل روزمره
- اطمینان از درستی عملکرد سیستم‌های با کارهای حساس
- تست و ارزیابی رسمی
- فرایند حائز اهمیت مدل‌سازی کد از زبان C به Promela
- عملیاتی طاقت فرسا به صورت غیرخودکار

هدف

- شناسایی ابزار های موجود تولید مدل به صورت خودکار
- کار بر روی ابزار های موجود و رفع نقاط کاستی های آن‌ها؛ به صرفه تر از تولید ابزار جدید
- بنابراین، در ادامه خواهیم داشت:
  - مقایسه ی عملکرد این ابزارها
  - استخراج نقاط ضعف تولید کد در هر یک از آن ها
  - ارائه ی راه حل برای رفع کاستی های آن ها

سیستم های مشابه

ابزار Modex:

- هدف: استخراج خودکار مدل به زبان Promela از کد C
- استفاده از Promela به‌صورت محدود
  - اشاره‌گرها را اداره نمی‌کند و هیچ فراخوانی رویه‌ای ندارد.
- Promela به اندازه‌ی زبان C قدرتمند نیست. گاهاً نمی‌توان مستقیماً ترجمه کرد. از کدهای تعبیه شده ی C استفاده می‌شود.
- عدم امکان بررسی قطعات کد تعبیه‌شده توسط SPIN، ممکن است مخرب باشد.
  - مانند عملیات تقسیم بر صفر یا یک اشاره‌گر به فضای خالی

ابزار JPF(Java PathFinder):

- در ابتدا، توسعه به عنوان مترجمی از زیرمجموعه‌ای از Java به کد Promela
- استفاده از Spin برای بررسی مدل ترجمه شده
- چندسال بعد، توسعه به عنوان یک ماشین مجازی
- بررسی خودکار تمام مسیرهای اجرای احتمالی یک برنامه برای یافتن نقض ویژگی ها

ابزار SDL2PML:

- تولید مستقل و خودکار مدل Promela از SDL
- استفاده از Spin برای بررسی مدل ترجمه شده
- تولید خودکار مدل های قابل ارزیابی با Spin
- مزیت کدهای تعبیه‌شده ی C در کد Promela
- موفقیت در تولید مدل در کاربردهای دنیای واقعی
- عدم پشتیبانی برنامه‌های با اجرای چندنخی ، به عنوان کاستی قابل‌توجه

:ke thesis

- مقاله‌ای تحت عنوان Model Checking C Programs by Translating C to Promela
- روش‌هایی موثر برای استخراج کد Promela از کدC
- تمرکز بر روی ضعف‌های سایر ابزارها، برای رفع کاستی‌ها

نتیجه گیری:

- ابزار Modex بهترین ابزار دردسترس برای تبدیل کد C به Promela تشخیص‌داده‌شد.
- تصمیم بر بهبود برخی عملکردهای آن، با الهام از روش‌های مقاله فوق

روش پیشنهادی

شرح مسئله

- نقطه ضعف Modex مربوط به عدم فراخوانی هیچ گونه رویه ای

- عدم تطابق ترتیب اجرای رویه های کد Promela تولید شده، با ترتیب اجرای صحیح توابع کد، در برنامه‌های با فراخوانی‌های پیچیده‌تر توابع در زبان C (مثال در شکل ۱)

```c
// target.c
int main()
{
   while(1)
   f1();
   return 0;
}
int f1()
{
   int i;
   for(i = 1; i < 10; i++)
   f2();
   return 0;
}
int f2()
{
   f3();
   return 0;
}
int f3()
{
   f1();
   return 0;
}
```

شکل ۱: کد C

مدل سازی با Modex

حالت اول:

- عدم وجود فایل prx. / وجود فایل prx. با محتوای شکل ۲
- مدل Promela در شکل‌های ۳ و ۴

```
// target.prx
%x -xe
```

شکل ۲: محتوای فایل prx.

```
// model.pml
int res_p_f3;
bool lck_p_f3_ret;
bool lck_p_f3;
int res_p_f2;
bool lck_p_f2_ret;
bool lck_p_f2;
int res_p_f1;
bool lck_p_f1_ret;
bool lck_p_f1;
int res_p_main;
bool lck_p_main_ret;
bool lck_p_main;
chan ret_p_f3 = [1] of { pid };
chan exc_cll_p_f3 = [0] of { pid };
chan req_cll_p_f3 = [1] of { pid };
chan ret_p_f2 = [1] of { pid };
chan exc_cll_p_f2 = [0] of { pid };
chan req_cll_p_f2 = [1] of { pid };
chan ret_p_f1 = [1] of { pid };
chan exc_cll_p_f1 = [0] of { pid };
chan req_cll_p_f1 = [1] of { pid };
chan ret_p_main = [1] of { pid };
chan exc_cll_p_main = [0] of { pid };
chan req_cll_p_main = [1] of { pid };
active proctype p_main()
{
  pid lck_id;
  L_0:
  do
  :: true;
  atomic {
    lck_p_f1 == 0 && empty(req_cll_p_f1) -> req_cll_p_f1!_pid;
    exc_cll_p_f1!_pid;
  }
  ret_p_f1?eval(_pid);
  c_code { ; now.lck_p_f1_ret = 0; };
  goto L_0;
  :: c_expr { !1 }; -> break
  od;
  atomic { !lck_p_main_ret -> lck_p_main_ret = 1 };
  c_code { now.res_p_main = (int ) 0; }; goto Return;
  Return: skip;
}
active proctype p_f1()
{
  int i;
  pid lck_id;
  endRestart:
  atomic {
    nempty(req_cll_p_f1) && !lck_p_f1 -> lck_p_f1 = 1;
    req_cll_p_f1?lck_id; exc_cll_p_f1?eval(lck_id);
    lck_p_f1 = 0;
  };
  c_code { Pp_f1->i=1; };
  L_1:
  do
  :: c_expr { (Pp_f1->i<10) };
```

شکل ۳: فایل نتیجه Modex

مدل سازی با Modex (ادامه)

```
    atomic {
      lck_p_f2 == 0 && empty(req_cll_p_f2) -> req_cll_p_f2!_pid;
      exc_cll_p_f2!_pid;
    }
    ret_p_f2?eval(_pid);
    c_code { ; now.lck_p_f2_ret = 0; };
    c_code { Pp_f1->i++; };
    goto L_1;
    c_code { Pp_f1->i++; };
    :: c_expr { !(Pp_f1->i<10) }; -> break
    od;
    atomic { !lck_p_f1_ret -> lck_p_f1_ret = 1 };
    c_code { now.res_p_f1 = (int ) 0; }; goto Return;
    Return: skip;
    ret_p_f1!lck_id;
    goto endRestart
}
active proctype p_f2()
{
    pid lck_id;
    endRestart:
    atomic {
      nempty(req_cll_p_f2) && !lck_p_f2 -> lck_p_f2 = 1;
      req_cll_p_f2?lck_id; exc_cll_p_f2?eval(lck_id);
      lck_p_f2 = 0;
    };
    atomic {
      lck_p_f3 == 0 && empty(req_cll_p_f3) -> req_cll_p_f3!_pid;
      exc_cll_p_f3!_pid;
    }
    ret_p_f3?eval(_pid);
    c_code { ; now.lck_p_f3_ret = 0; };
    atomic { !lck_p_f2_ret -> lck_p_f2_ret = 1 };
    c_code { now.res_p_f2 = (int ) 0; }; goto Return;
    Return: skip;
    ret_p_f2!lck_id;
    goto endRestart
}
active proctype p_f3()
{
    pid lck_id;
    endRestart:
    atomic {
      nempty(req_cll_p_f3) && !lck_p_f3 -> lck_p_f3 = 1;
      req_cll_p_f3?lck_id; exc_cll_p_f3?eval(lck_id);
      lck_p_f3 = 0;
    };
    atomic {
      lck_p_f1 == 0 && empty(req_cll_p_f1) -> req_cll_p_f1!_pid;
      exc_cll_p_f1!_pid;
    }
    ret_p_f1?eval(_pid);
    c_code { ; now.lck_p_f1_ret = 0; };
    atomic { !lck_p_f3_ret -> lck_p_f3_ret = 1 };
    c_code { now.res_p_f3 = (int ) 0; }; goto Return;
    Return: skip;
    ret_p_f3!lck_id;
    goto endRestart
}
```

شکل ۴: ادامه فایل نتیجه Modex

- ۵ مسیر اجرایی:

i. f3 – f2 – f1 – main
ii. f3 – f2 – f1 – main – f1 – main
iii. f1 – main – f1 – main
iv. main– f1 – main
v. main

- برای مثال، مسیر چهارم در شکل ۵ نشان داده شده.

مدل سازی با Modex (ادامه)

- مشکلات:
  - i. تابع اصلی در کد C دارای یک حلقه‌ی بی‌نهایت است. و نباید هیچ‌گاه فراخوانی توابع به پایان برسد.
  - ii. دلیل: عدم امکان فراخوانی رویه‌ها در کد تولیدی Modex
  - iii. تمام رویه‌ها به صورت فعال تولید شده
- کنترل نسبی ترتیب اجرای تمام رویه‌های تولید شده ی فعال، با استفاده از کانال‌های هم‌گام‌ساز

```
Selected: 4
1: proc 0 (p_main:1) model-noprx.pml:33 (state 1) [(1)]
Selected: 4
2: proc 0 (p_main:1) model-noprx.pml:35 (state 2)
    [(((lck_p_f1==0)&&empty(req_cll_p_f1)))]
3: proc 0 (p_main:1) model-noprx.pml:35 (state 3) [req_cll_p_f1!_pid]
Selected: 3
4: proc 1 (p_f1:1) model-noprx.pml:53 (state 1) [((nempty(req_cll_p_f1)&&!(lck_p_f1)))]
5: proc 1 (p_f1:1) model-noprx.pml:53 (state 2) [lck_p_f1 = 1]
6: proc 1 (p_f1:1) model-noprx.pml:54 (state 3) [req_cll_p_f1?lck_id]
Selected: 4
7: proc 0 (p_main:1) model-noprx.pml:36 (state 4) [exc_cll_p_f1!_pid]
7: proc 1 (p_f1:1) model-noprx.pml:54 (state 4) [exc_cll_p_f1?eval(lck_id)]
8: proc 1 (p_f1:1) model-noprx.pml:55 (state 5) [lck_p_f1 = 0]
Selected: 3
c_code4: {  /* line 57 model-noprx.pml */
  Pp_f1->i=1; }
9: proc 1 (p_f1:1) model-noprx.pml:57 (state 7) [{c_code4}]
c_code5:  /* line 60 model-noprx.pml */
(Pp_f1->i<10)
c_code9:  /* line 70 model-noprx.pml */
!(Pp_f1->i<10)
Selected: 4
c_code9:  /* line 70 model-noprx.pml */
!(Pp_f1->i<10)
10:  proc 1 (p_f1:1) model-noprx.pml:70 (state 18) [({c_code9})]
Selected: 3
11:  proc 1 (p_f1:1) model-noprx.pml:59 (state 22) [break]
Selected: 3
12:  proc 1 (p_f1:1) model-noprx.pml:72 (state 23) [(!(lck_p_f1_ret))]
13:  proc 1 (p_f1:1) model-noprx.pml:72 (state 24) [lck_p_f1_ret = 1]
Selected: 3
c_code10: {  /* line 73 model-noprx.pml */
  14:  proc 1 (p_f1:1) model-noprx.pml:73 (state 26) [{c_code10}]
  Selected: 3
  15:  proc 1 (p_f1:1) model-noprx.pml:74 (state 28) [(1)]
  Selected: 3
  16:  proc 1 (p_f1:1) model-noprx.pml:75 (state 29) [ret_p_f1!lck_id]
  Selected: 4
  17:  proc 0 (p_main:1) model-noprx.pml:38 (state 6) [ret_p_f1?eval(_pid)]
  Selected: 4
  c_code1: {  /* line 39 model-noprx.pml */
    18:  proc 0 (p_main:1) model-noprx.pml:39 (state 7) [{c_code1}]
    c_code2:  /* line 41 model-noprx.pml */
    !1
    Selected: 5
    c_code2:  /* line 41 model-noprx.pml */
    !1
    19:  proc 0 (p_main:1) model-noprx.pml:41 (state 9) [({c_code2})]
    Selected: 4
    20:  proc 0 (p_main:1) model-noprx.pml:32 (state 13) [break]
    Selected: 4
    21:  proc 0 (p_main:1) model-noprx.pml:43 (state 14) [(!(lck_p_main_ret))]
    22:  proc 0 (p_main:1) model-noprx.pml:43 (state 15) [lck_p_main_ret = 1]
    Selected: 4
    c_code3: {  /* line 44 model-noprx.pml */
      23:  proc 0 (p_main:1) model-noprx.pml:44 (state 17) [{c_code3}]
      Selected: 4
      24:  proc 0 (p_main:1) model-noprx.pml:45 (state 19) [(1)]
```

شکل ۵: مسیر اجرایی چهارم

مدل سازی با Modex (ادامه)

- **حالت دوم:**
  - وجود فایل prx. با محتوای شکل ۶
  - عدم استفاده از کانال‌های هم‌گام‌ساز، با وجود فعال بودن رویه‌ها ← مسیرهای اجرایی بیشتر
  - مدل Promela در شکل ۷

```
// target.prx
%x -x
```

شکل ۶: محتوای فایل prx.

```
// model.pml
active proctype p_main()
{
   L_0:
   do
   :: true;
   c_code { f1(); };
   goto L_0;
   :: c_expr { !1 }; -> break
   od;
   goto Return;
   Return: skip;
}
active proctype p_f1()
{
   int i;
   c_code { Pp_f1->i=1; };
   L_1:
   do
   :: c_expr { (Pp_f1->i<10) };
   c_code { f2(); };
   c_code { Pp_f1->i++; };
   goto L_1;
   c_code { Pp_f1->i++; };
   :: c_expr { !(Pp_f1->i<10) }; -> break
   od;
   goto Return;
   Return: skip;
}
active proctype p_f2()
{
   c_code { f3(); };
   goto Return;
   Return: skip;
}
active proctype p_f3()
{
   c_code { f1(); };
   goto Return;
   Return: skip;
}
```

شکل ۷: فایل نتیجه Modex

اصلاح مدل خروجی Modex

- ایجاد تغییر برروی کد شکل ۷
- **گام اول:**
  - کد حاصل در شکل ۸
  - تغییرات:

i. حذف بلاک‌های C تعبیه شده

ii. فقط main فعال

```
// model.pml
active proctype p_main()
{
   L_0:
   do
   :: true;
   run p_f1();
   goto L_0;
   :: !1 -> break
   od;
   goto Return;
   Return: skip;
}
proctype p_f1()
{
   int i;
   i=1;
   L_1:
   do
   :: i<10;
   run p_f2();
   i++;
   goto L_1;
   i++;
   :: !(i<10) -> break
   od;
   goto Return;
   Return: skip;
}
proctype p_f2()
{
   run p_f3();
   goto Return;
   Return: skip;
}
proctype p_f3()
{
   run p_f1();
   goto Return;
   Return: skip;
}
```

شکل ۸: کد حاصل از گام اول

اصلاح مدل خروجی Modex (ادامه)

- همچنان بیش از یک مسیر اجرایی
- یکی از مسیرها، مسیر درست است.
- عدم قطعیت درباره‌ی اجرای دو رویه فراخوانی‌کننده و فراخوانی‌شونده، پس از فراخوانی رویه
- مسیر درست در شکل ۹

```
0: proc - (:root:) creates proc 0 (p_main)
1: proc 0 (p_main:1) model-prefer.pml:8 (state 1) [(1)]
Starting p_f1 with pid 1
2: proc 0 (p_main:1) creates proc 1 (p_f1)
2: proc 0 (p_main:1) model-prefer.pml:9 (state 2) [(run p_f1())]
Selected: 1
3: proc 1 (p_f1:1) model-prefer.pml:19 (state 1) [i = 1]
Selected: 1
4: proc 1 (p_f1:1) model-prefer.pml:22 (state 2) [((i<10))]
Selected: 1
Starting p_f2 with pid 2
5: proc 1 (p_f1:1) creates proc 2 (p_f2)
5: proc 1 (p_f1:1) model-prefer.pml:23 (state 3) [(run p_f2())]
Selected: 1
Starting p_f3 with pid 3
6: proc 2 (p_f2:1) creates proc 3 (p_f3)
6: proc 2 (p_f2:1) model-prefer.pml:34 (state 1) [(run p_f3())]
Selected: 1
Starting p_f1 with pid 4
7: proc 3 (p_f3:1) creates proc 4 (p_f1)
7: proc 3 (p_f3:1) model-prefer.pml:40 (state 1) [(run p_f1())]
Selected: 1
8: proc 4 (p_f1:1) model-prefer.pml:19 (state 1) [i = 1]
.
.
.
```

شکل ۹: مسیر اجرایی مربوط به کد حاصل از گام اول

اصلاح مدل خروجی Modex (ادامه)

- **گام دوم:**
  - کد حاصل در دو شکل ۱۰ و ۱۱
  - تغییرات:
    - i. تعریف دو متغیر عمومی، برای بلاک شدن مسیرهای غیرمنتظره، هنگامی که بدنه‌ی چند رویه به صورت هم‌زمان امکان اجرا دارند

```
// model.pml
int function_turn;
int function_prev;
active proctype p_main()
{
   function_turn = 0;
   function_prev = -1;
   L_0:
   do
   :: true;
   function_prev = 0;
   function_turn = 1;
   run p_f1();
   L_turn_0:
   do
   :: !(function_turn == 0);
   goto L_turn_0;
   :: function_turn == 0 -> break
   od;
   goto L_0;
   :: !1 -> break
   od;
   goto Return;
   Return: skip;
   function_turn = function_prev;
}
proctype p_f1()
{
   int i;
   i=1;
   L_1:
   do
   :: i<10;
   function_prev = 1;
   function_turn = 2;
   run p_f2();
   L_turn_1:
   do
   :: !(function_turn == 1);
   goto L_turn_1;
   :: function_turn == 1 -> break
   od;
   i++;
   goto L_1;
```

شکل ۱۰: کد حاصل از گام دوم

اصلاح مدل خروجی Modex (ادامه)

```
    i++;
    :: !(i<10) -> break
    od;
    goto Return;
    Return: skip;
    function_turn = function_prev;
}
proctype p_f2()
{
    function_prev = 2;
    function_turn = 3;
    run p_f3();
    L_turn_2:
    do
    :: !(function_turn == 2);
    goto L_turn_2;
    :: function_turn == 2 -> break
    od;
    goto Return;
    Return: skip;
    function_turn = function_prev;
}
proctype p_f3()
{
    function_prev = 3;
    function_turn = 1;
    run p_f1();
    L_turn_3:
    do
    :: !(function_turn == 3);
    goto L_turn_3;
    :: function_turn == 3 -> break
    od;
    goto Return;
    Return: skip;
    function_turn = function_prev;
}
```

شکل ۱۱: ادامه کد حاصل از گام دوم

- مشکلات:
  - i. عدم قطعیت و اشغال cpu
  - ii. مشکل هنگام فراخوانی مجدد یک رویه

اصلاح مدل خروجی Modex (ادامه)

- **گام سوم:**
  - کد حاصل در شکل ۱۲
  - تغییر:

    استفاده از کانال‌های هم‌گام‌ساز محلی، به جای متغیر عمومی
  - رفع دو مشکل گام قبل

```
// model.pml
active proctype p_main()
{
   L_0:
   do
   :: true;
   int flag;
   chan out_p_f1 = [0] of {int};
   run p_f1(out_p_f1);
   out_p_f1 ? flag;
   goto L_0;
   :: !1 -> break
   od;
   goto Return;
   Return: skip;
}
proctype p_f1(chan out_p_f1)
{
   int i;
   i=1;
   L_1:
   do
   :: i<10;
   int flag;
   chan out_p_f2 = [0] of {int};
   run p_f2(out_p_f2);
   out_p_f2 ? flag;
   i++;
   goto L_1;
   i++;
   :: !(i<10) -> break
   od;
   goto Return;
   Return: skip;
   out_p_f1 ! 1;
}
proctype p_f2(chan out_p_f2)
{
   int flag;
   chan out_p_f3 = [0] of {int};
   run p_f3(out_p_f3);
   out_p_f3 ? flag;
   goto Return;
   Return: skip;
   out_p_f2 ! 1;
}
proctype p_f3(chan out_p_f3)
{
   int flag;
   chan out_p_f1 = [0] of {int};
   run p_f1(out_p_f1);
   out_p_f1 ? flag;
   goto Return;
   Return: skip;
   out_p_f3 ! 1;
}
```

شکل ۱۲: کد حاصل از گام سوم

اصلاح مدل خروجی Modex (ادامه)

- تنها یک مسیر اجرایی
- مسیر در شکل ۱۳

```
0: proc - (:root:) creates proc 0 (p_main)
1: proc 0 (p_main:1) model-prefer.pml:8 (state 1) [(1)]
2: proc 0 (p_main:1) model-prefer.pml:10 (state 2) [flag = 0]
Starting p_f1 with pid 1
3: proc 0 (p_main:1) creates proc 1 (p_f1)
3: proc 0 (p_main:1) model-prefer.pml:11 (state 3) [(run p_f1(out_p_f1))]
Selected: 1
4: proc 1 (p_f1:1) model-prefer.pml:22 (state 1) [i = 1]
Selected: 1
5: proc 1 (p_f1:1) model-prefer.pml:25 (state 2) [((i<10))]
Selected: 1
6: proc 1 (p_f1:1) model-prefer.pml:27 (state 3) [flag = 0]
Selected: 1
Starting p_f2 with pid 2
7: proc 1 (p_f1:1) creates proc 2 (p_f2)
7: proc 1 (p_f1:1) model-prefer.pml:28 (state 4) [(run p_f2(out_p_f2))]
Selected: 1
Starting p_f3 with pid 3
8: proc 2 (p_f2:1) creates proc 3 (p_f3)
8: proc 2 (p_f2:1) model-prefer.pml:43 (state 1) [(run p_f3(out_p_f3))]
Selected: 1
Starting p_f1 with pid 4
9: proc 3 (p_f3:1) creates proc 4 (p_f1)
9: proc 3 (p_f3:1) model-prefer.pml:53 (state 1) [(run p_f1(out_p_f1))]
Selected: 1
10:  proc 4 (p_f1:1) model-prefer.pml:22 (state 1) [i = 1]
Selected: 1
11:  proc 4 (p_f1:1) model-prefer.pml:25 (state 2) [((i<10))]
Selected: 1
12:  proc 4 (p_f1:1) model-prefer.pml:27 (state 3) [flag = 0]
Selected: 1
Starting p_f2 with pid 5
13:  proc 4 (p_f1:1) creates proc 5 (p_f2)
13:  proc 4 (p_f1:1) model-prefer.pml:28 (state 4) [(run p_f2(out_p_f2))]
Selected: 1
Starting p_f3 with pid 6
14:  proc 5 (p_f2:1) creates proc 6 (p_f3)
14:  proc 5 (p_f2:1) model-prefer.pml:43 (state 1) [(run p_f3(out_p_f3))]
Selected: 1
Starting p_f1 with pid 7
15:  proc 6 (p_f3:1) creates proc 7 (p_f1)
15:  proc 6 (p_f3:1) model-prefer.pml:53 (state 1) [(run p_f1(out_p_f1))]
Selected: 1
16:  proc 7 (p_f1:1) model-prefer.pml:22 (state 1) [i = 1]
Selected: 1
17:  proc 7 (p_f1:1) model-prefer.pml:25 (state 2) [((i<10))]
Selected: 1
18:  proc 7 (p_f1:1) model-prefer.pml:27 (state 3) [flag = 0]
Selected: 1
Starting p_f2 with pid 8
19:  proc 7 (p_f1:1) creates proc 8 (p_f2)
19:  proc 7 (p_f1:1) model-prefer.pml:28 (state 4) [(run p_f2(out_p_f2))]
.
.
.
```

شکل ۱۳: مسیر اجرایی مربوط به کد حاصل از گام سوم

نتیجه گیری

پژوهش حاضر

- تمرکز بر روی مدل‌های با فراخوانی‌های پیچیده‌ی توابع
- بهبود در عملکرد ابزار Modex
- توسعه ماژول خودکار
  - ورودی: کد Promela حاصل از Modex
  - اعمال اصلاحات بیان شده

کارهای آتی

- بهبود کاستی های دیگر ابزار Modex
  - تعمیم فراخوانی توابع
    - i. توابع بازگشتی
    - ii. توابع دارای پارامتر ورودی
  - اشاره‌گرها

- بررسی ویژگی‌ها و کارایی مدل‌های حاصل از روش‌های پیشنهادی

متشکرم!

سوال؟

Marzieh.alidaadi@gmail.com