접근 경로 분석

하동수
한양대학교 프로그램 분석 검증 연구실

2012.01.17
ROSAEC 워크샵

동기

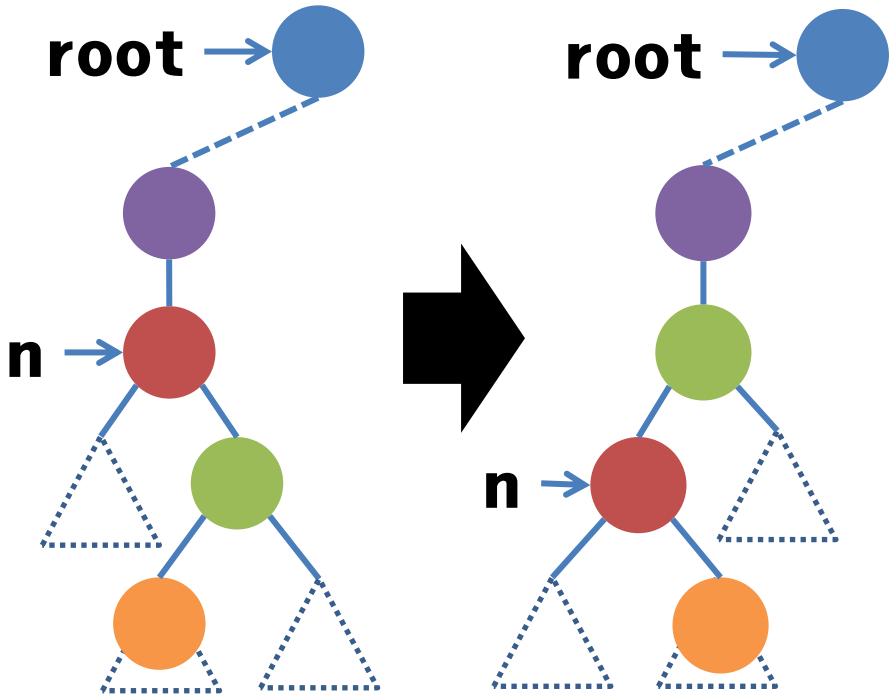
모양 분석에서 프로시져 분석시

인자로 **도달 가능한 셀**을 넘기는 대신
접근하는 셀만 넘긴다면

성능향상에 도움이 되지 않을까?

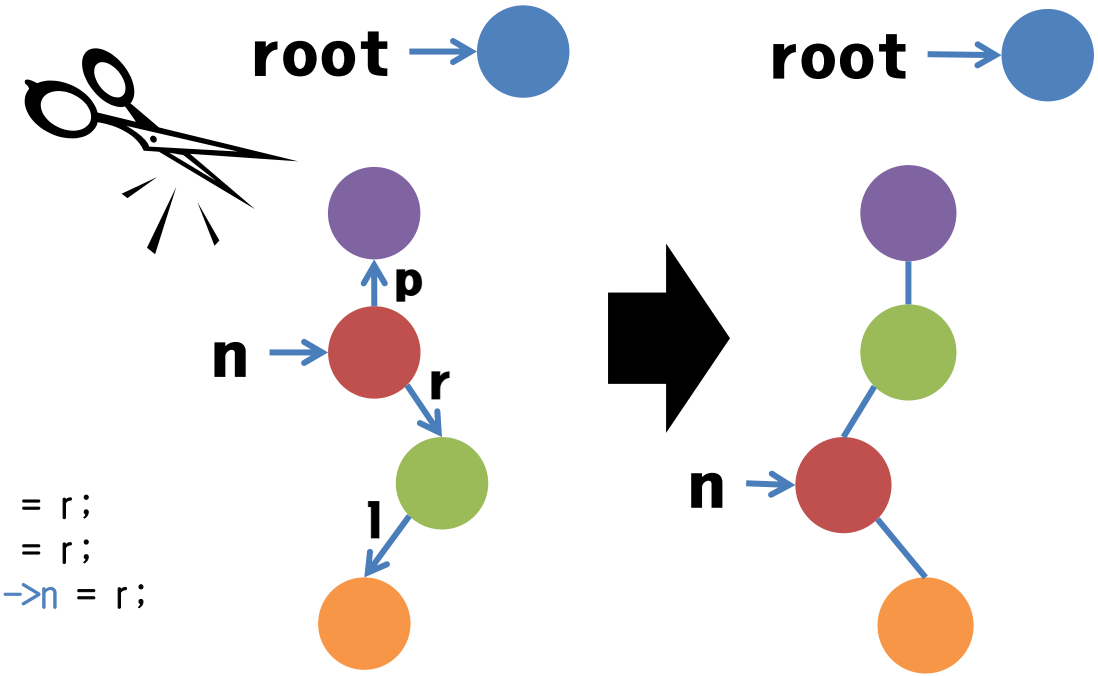
동기 예시

```
void _rotate_left(struct rb_node *n, struct rb_root *root)
{
    struct node *r;
    struct node *p;

    r = n->r;
    p = n->p;

    if(n->r) r->l->p = n;

    r->l = n;
    r->p = p;

    if(p & n == p->l)       p->l = r;
    else if(p & n != p->l) p->r = r;
    else                    root->n = r;

    n->p = r;
}
```



동기 예시

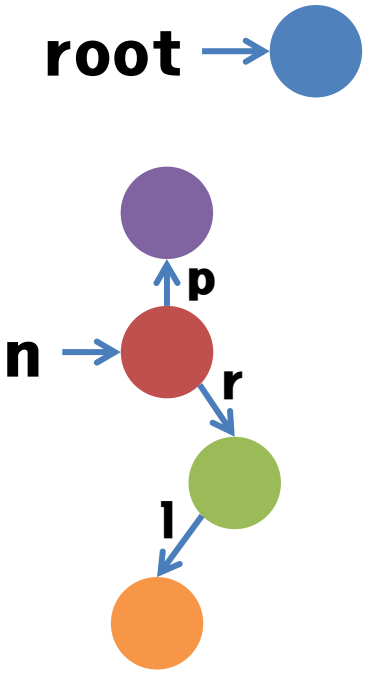
```
void _rotate_left(struct rb_node *n, struct rb_root *root)
{
    struct node *r;
    struct node *p;

    r = n->r;
    p = n->p;

    if(n->r) r->l->p = n;

    r->l = n;
    r->p = p;

    if(p & n == p->l)       p->l = r;
    else if(p & n != p->l) p->r = r;
    else                    root->n = r;

    n->p = r;
}
```



경로의 정의

- 스택 변수와 필드의 리스트로 정의
  - $x.f_1\ f_2\ f_3\ f_4\ f_5$

- 다음 예시의 접근 경로
  - {n.p, n.rl, root}



기본적인 경로 분석

- 상향식 접근 경로 분석

기본적인 경로 분석

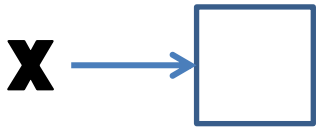
- 상향식 접근 경로 분석
- …

```
x->f = y;

x = x->f;

x = x->g;

x = x->h;
                {}
```

기본적인 경로 분석

- 상향식 접근 경로 분석
- …

```
x->f = y;

x = x->f;

x = x->g;
                {x}
x = x->h;
                {}
```



기본적인 경로 분석

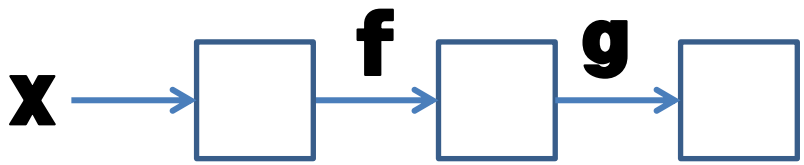
- 상향식 접근 경로 분석
- …

```
x->f = y;

x = x->f;
                {x.g}
x = x->g;
                {x}
x = x->h;
                {}
```

기본적인 경로 분석

- 상향식 접근 경로 분석
- …

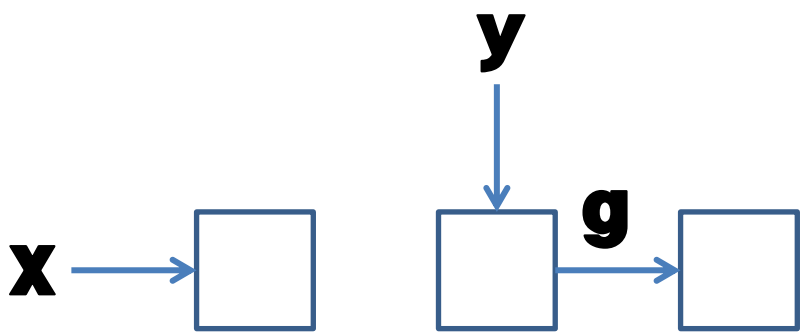
```
x->f = y;
                {x.fg}
x = x->f;
                {x.g}
x = x->g;
                {x}
x = x->h;
                {}
```



기본적인 경로 분석

- 상향식 접근 경로 분석
- …

```
                    {x, y.g}
x->f = y;
                    {x.fg}
x = x->f;
                    {x.g}
x = x->g;
                    {x}
x = x->h;
                    {}
```



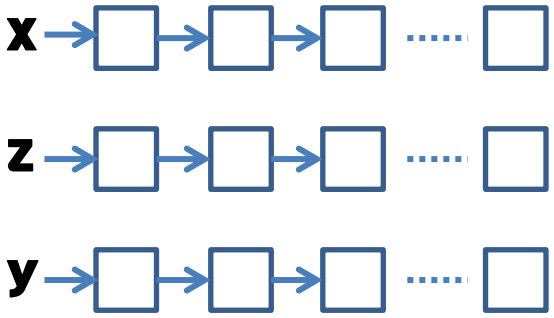
한계점

```
void bad_example(struct L *x, struct L *y, struct L *z)
{
    struct L *t;

    while(x != 0)
    {
        t = x;
        x = x->f;
        t->f = z;
    }

    while(y != 0)
        y = y->f;
}
```

한계점

```
void bad_example(struct L *x, struct L *y, struct L *z)
{
    struct L *t;

    while(x != 0)
    {
        t = x;
        x = x->f;
        t->f = z;
    }

    while(y != 0)
        y = y->f;
}
```

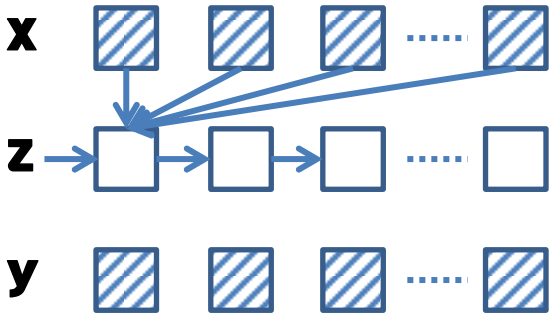
<alias가 없을 경우>



한계점

```
void bad_example(struct L *x, struct L *y, struct L *z)
{
    struct L *t;

    while(x != 0)
    {
        t = x;
        x = x->f;
        t->f = z;
    }

    while(y != 0)
        y = y->f;
}
```

<alias가 없을 경우>

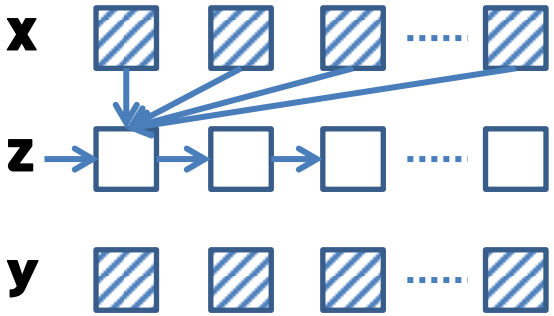


{xf*, yf*}

한계점

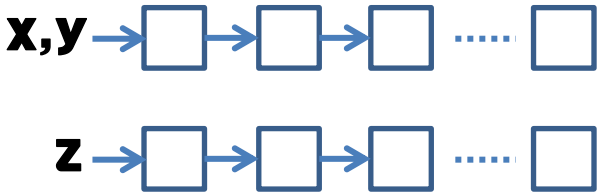
```
void bad_example(struct L *x, struct L *y, struct L *z)
{
    struct L *t;

    while(x != 0)
    {
        t = x;
        x = x->f;
        t->f = z;
    }

    while(y != 0)
        y = y->f;
}
```

<alias가 없을 경우>



{xf*, yf*}

<x,y가 alias일 경우>

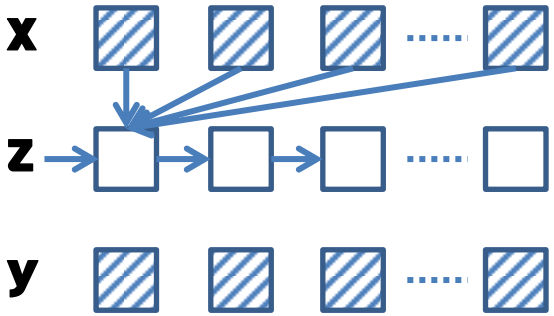


HANYANG UNIVERSITY

한계점

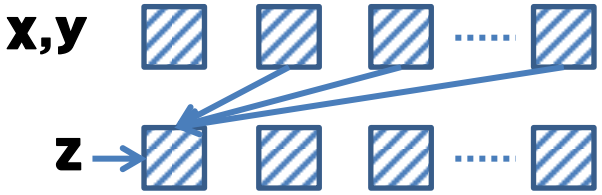
```
void bad_example(struct L *x, struct L *y, struct L *z)
{
    struct L *t;

    while(x != 0)
    {
        t = x;
        x = x->f;
        t->f = z;
    }

    while(y != 0)
        y = y->f;
}
```

<alias가 없을 경우>



{xf*, yf*}

<x,y가 alias일 경우>



{xf*, y, zf*}

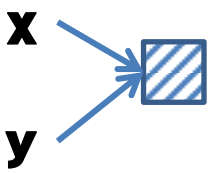
조건 경로

- Alias 상황을 다루기 위해 조건 추가
  - 조건 => 경로
  - sh(x,y) => $x.f_1f_2f_3$

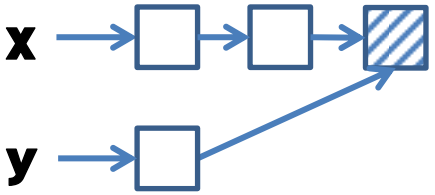
- "조건=>경로"의 의미
  - 해당 메모리 상태에서 "조건"을 만족한다면 해당 "경로"로 메모리 접근이 이뤄진다.

조건 분류

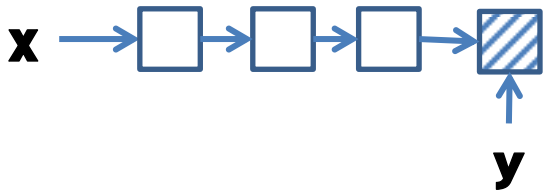
- x=y



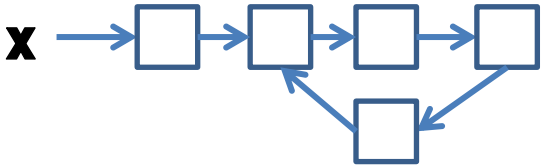
- sh(x,y)



- x->y

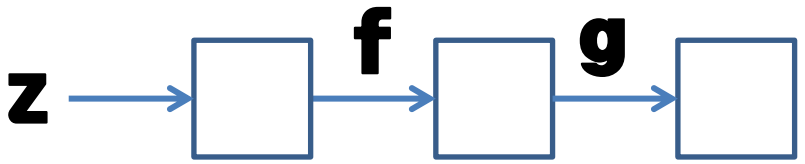


- cy(x)



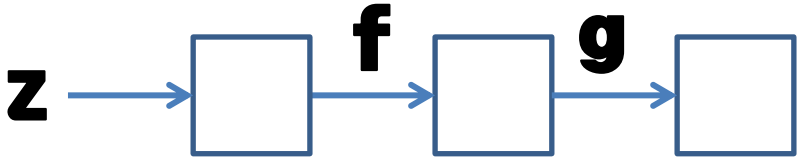
조건을 추가한 경로 분석

- { z.fg }




```
x->f = y;
```

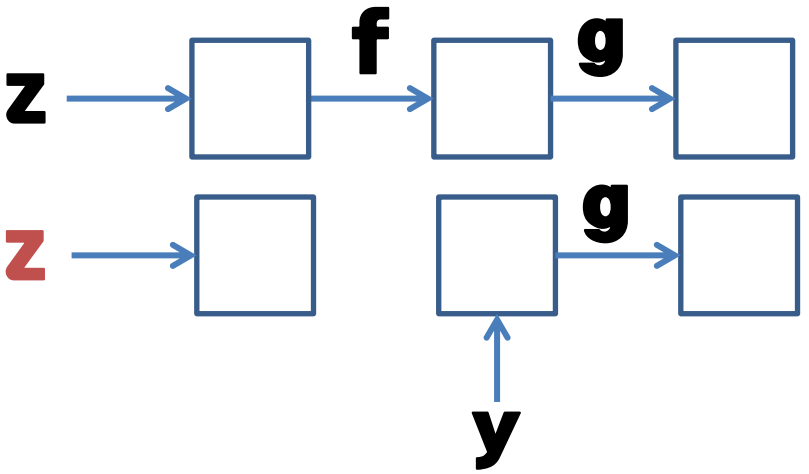
{ z.fg }



HANYANG UNIVERSITY

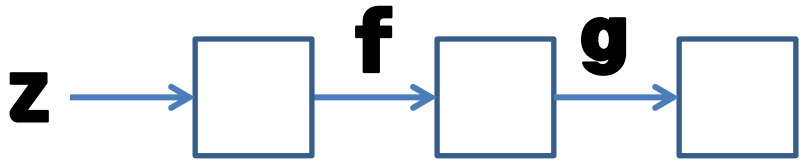
조건을 추가한 경로 분석

- { z.fg }
  ∪
  { x=z => y.g, … }




```
x->f = y;
```

{ z.fg }



정리

- 접근 경로 분석
  - 프로시져가 실제로 접근하는 힙 메모리 셀을 찾는 분석

- 분석 도메인의 형태
  - 조건 => 경로