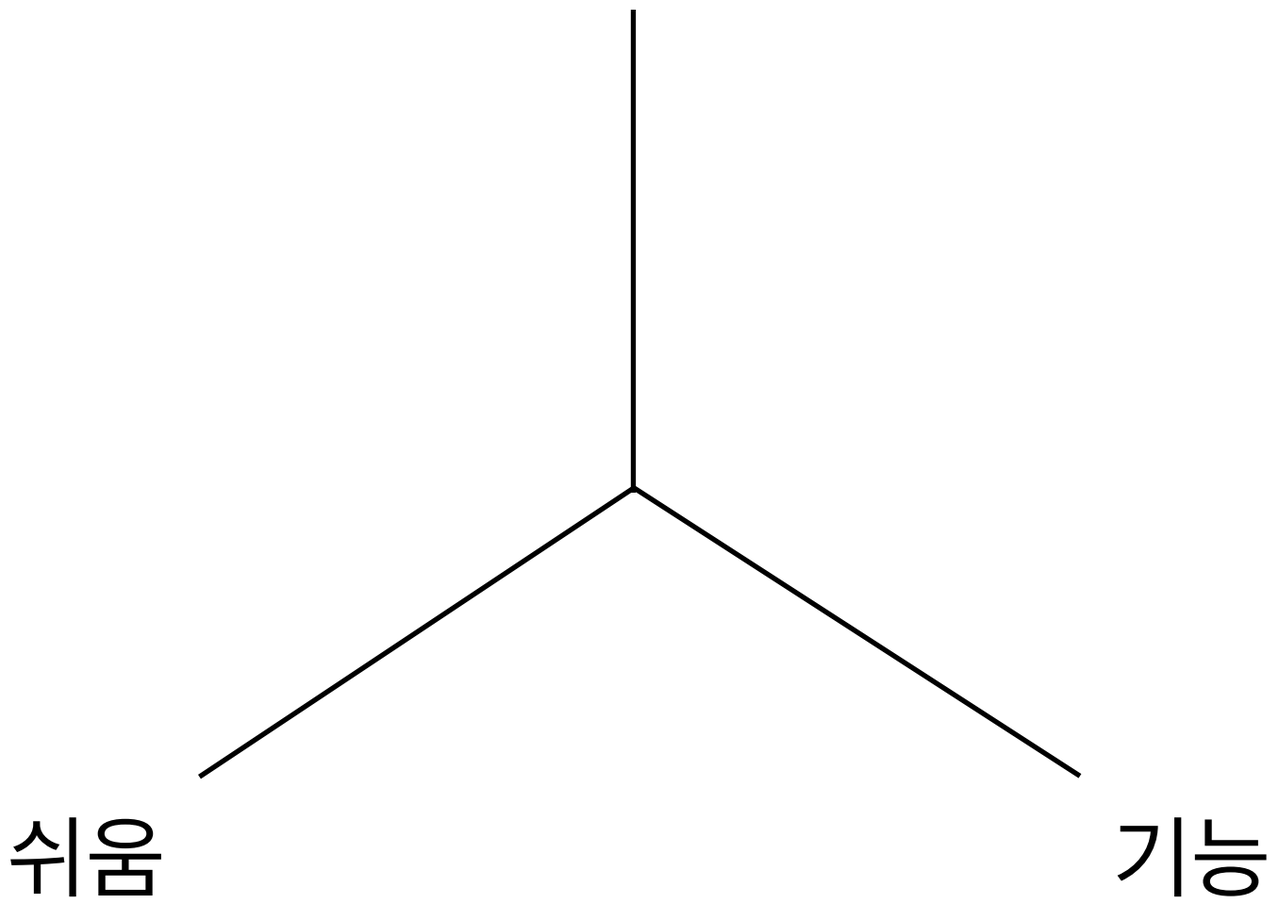
정수와 포인터간 변환을 지원하는 C/C++ 메모리 모델

Kang Hur Mansky Garbuzov Zdancewic Vafiadis

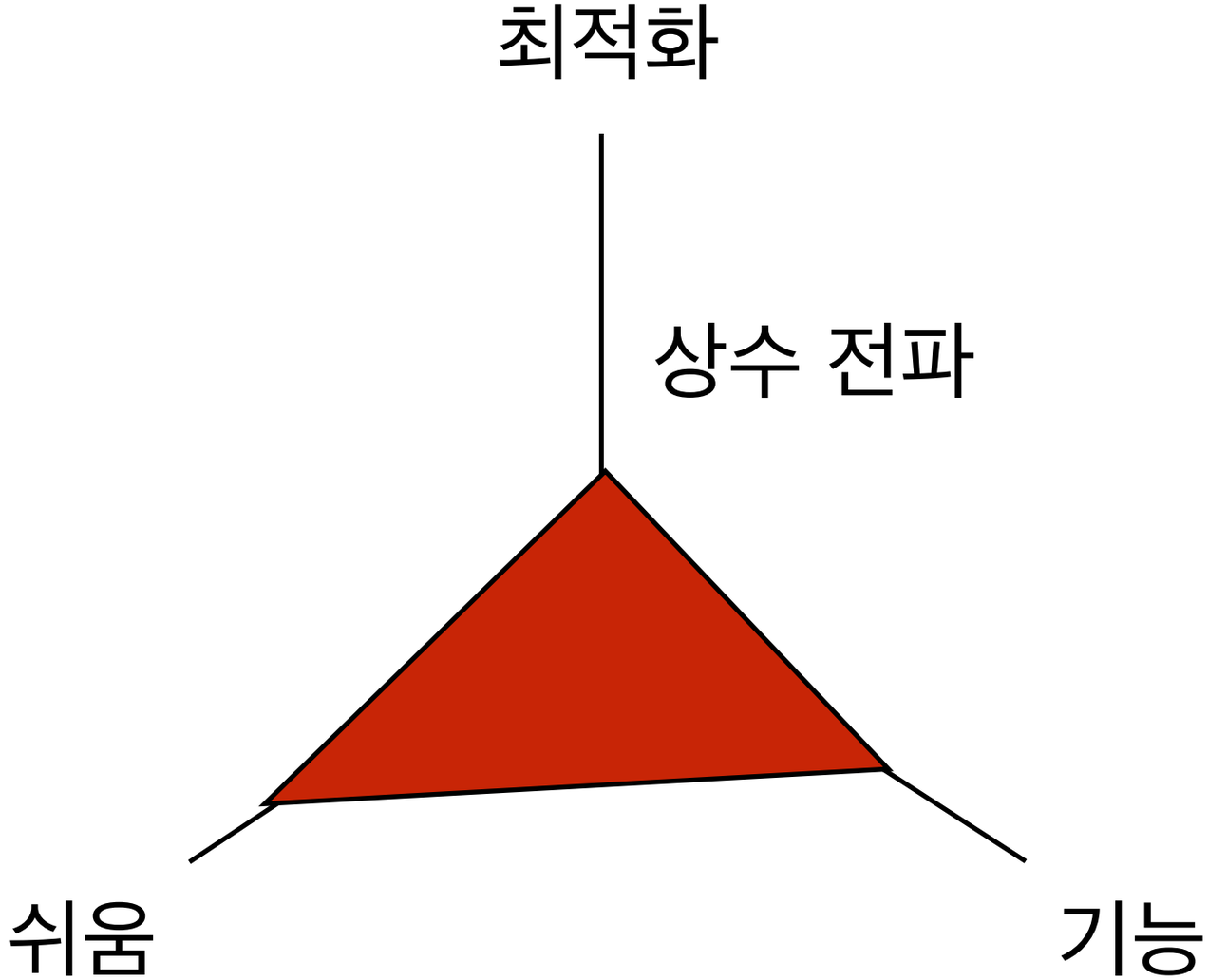
ROSAEC 웍샵
201501

메모리 모델 평가 기준



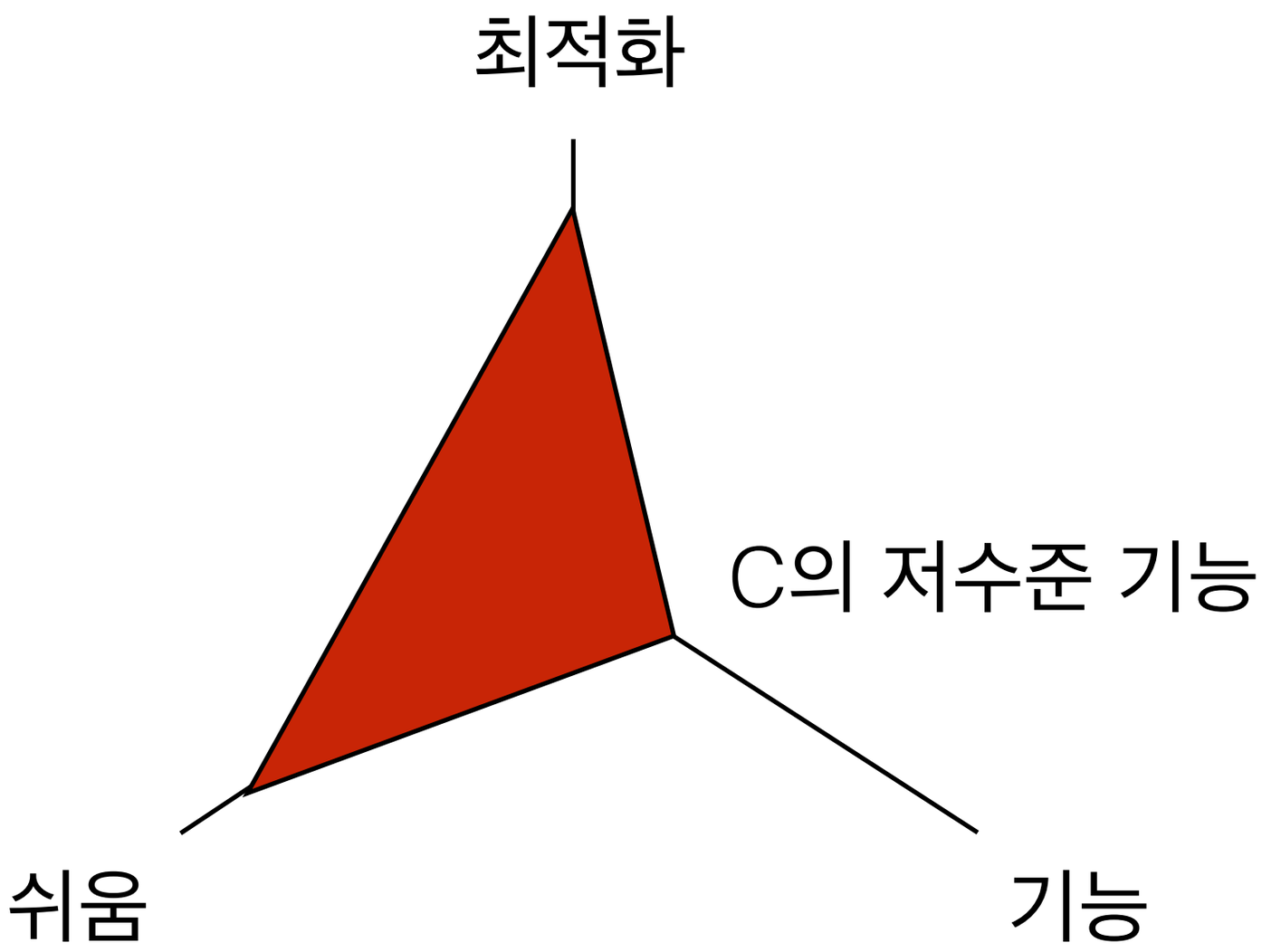
Hardware 메모리 모델

메모리 := int32 -> byte

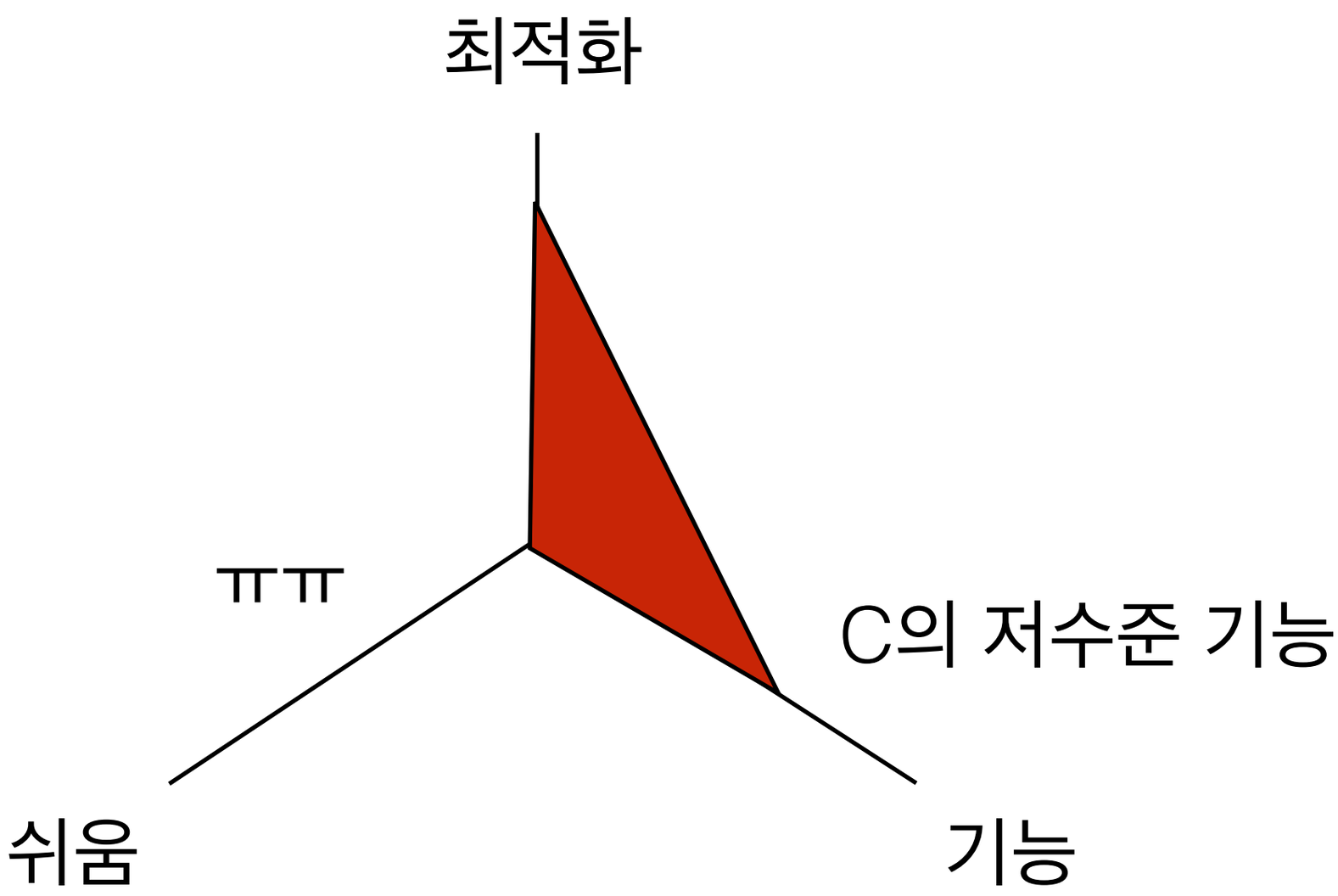


논리적인 메모리 모델

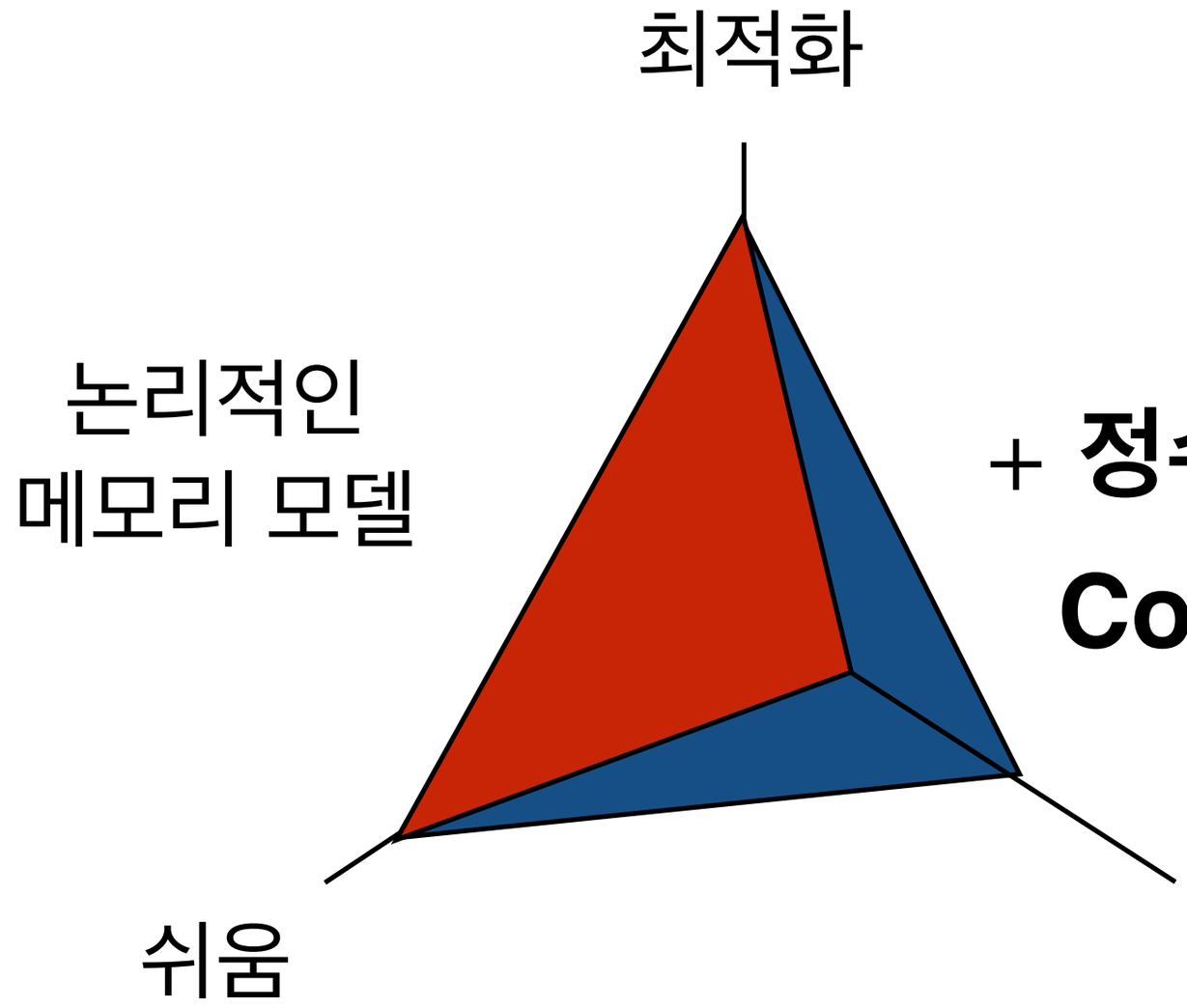
CompCert 메모리 모델



ISO 표준 메모리 모델



결론



+ 정수와 포인터간 변환

Coq으로 최적화 증명

학회에 제출중

“Memory model with **sensible tradeoffs**"

“The characterisation of casts to integers as side-effecting is **particularly compelling**”

좀 더 구체적으로

슬라이드는 총 20장

먼저, 왜 정수와 포인터간 변환?

- Linux, JVM 등 시스템 소프트웨어에 많이 사용됨
- Hash key로 포인터 사용
- 포인터 남는 bit에 마킹 (garbage collection, …)
- XOR linked list
  공간은 singly linked list, 기능은 doubly linked list
- 포인터 압축 (JVM): 64bit 포인터 -> 32bit 정수

예제

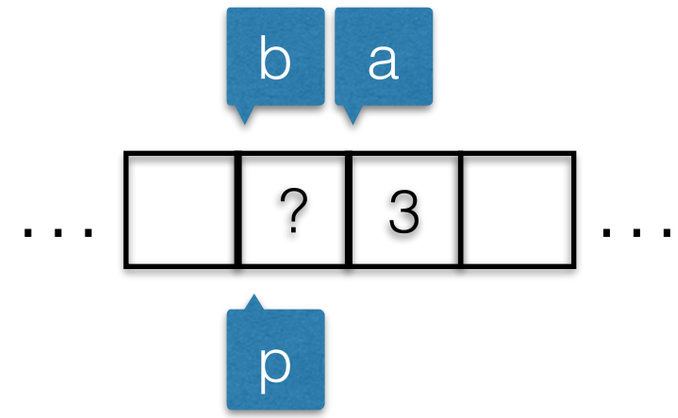
```
#include <stdio.h>

void foo(int *p) {
  *(p + 1) = 3;
}

int main() {
  int a = 0, b;
  foo(&b);
  printf("%d %d\n", a, (int) &a);
  return 0;
}
```

Hardware 모델:
a가 0이 아닐수도

논리적인 모델:
(int) &a가 undef

Hardware 메모리 모델의 문제

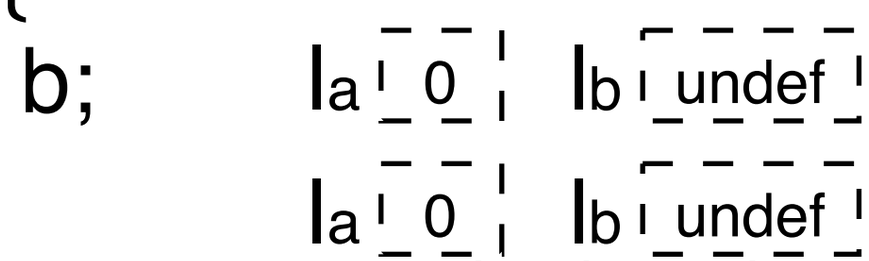
```
#include <stdio.h>

void foo(int *p) {
  *(p + 1) = 3;
}

int main() {
  int a = 0, b;
  foo(&b);
  printf("%d %d\n", a, (int) &a);
  return 0;
}
```



논리적인 메모리 모델의 문제

```
#include <stdio.h>

void foo(int *p) {
  *(p + 1) = 3;
}

int main() {
  int a = 0, b;
  foo(&b);
  printf("%d %d\n", a, (int) &a);
  return 0;
}
```

- 메모리 := 메모리 블락의 모임
- 메모리 블락 := (블락 이름, 블락에 담긴 값)
- 포인터 := (블락 이름, offset) $(l_b, 0)$



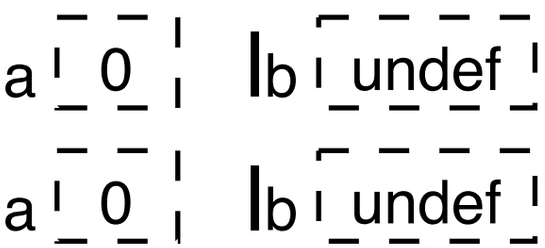
$(l_a, 0)$ 를 정수로?

우리 모델:
Hardware 모델과
논리적인 모델 섞기

포인터를 정수로 언제든지 변환 가능
정수로 변환되지 않은 포인터는 최적화 가능

우리 모델 예제

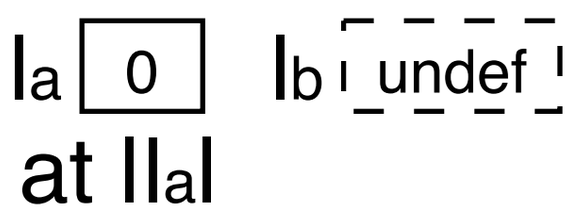
```
#include <stdio.h>

void foo(int *p) {
  *(p + 1) = 3;
}

int main() {
  int a = 0, b;
  foo(&b);
  printf("%d %d\n", a, (int) &a);
  return 0;
}
```



- **캐스팅할 때 블락에 정수 배정**
  (논리 블락 -> 물리 블락)

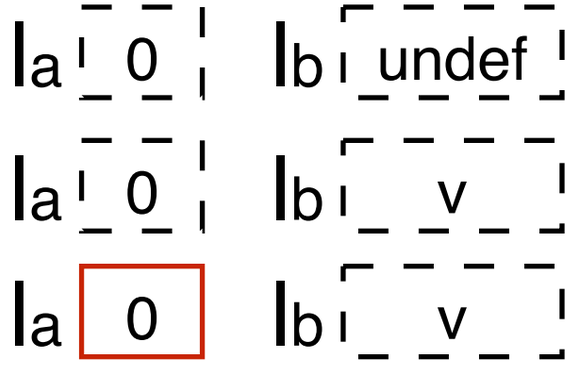
- 한번 물리는 평생 물리

- 포인터 (l_a, i)와
  정수 $|l_a|$ + i를 동일시

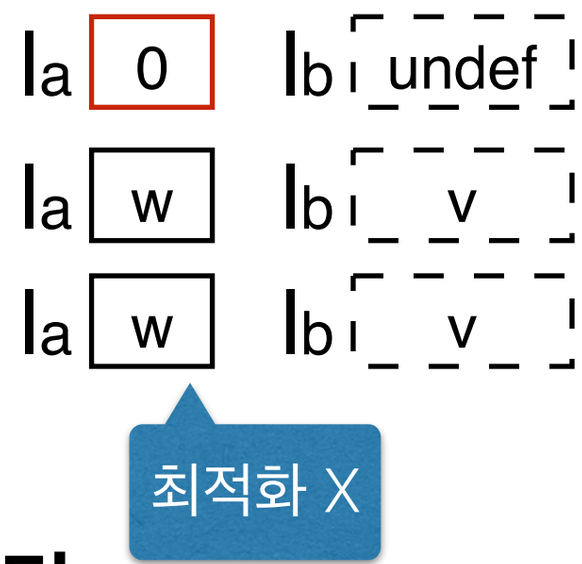


우리 모델 예제 2

```
int main() {
    int a = 0, b;
    foo(&b);
    printf(“%d %d\n”, a, (int) &a);
    return 0; }
```




```
int main() {
    int a = 0, b; (int) &a;
    foo(&b);
    printf("%d\n", a);
    return 0; }
```



암튼 이정도만 해도 충분히 최적화 잘됨

우리 모델은 **정말로** 최적화를 충분히 지원

GCC, Clang 컴파일러가 수행하는 중요한 최적화를
Coq으로 증명

증명한 최적화 예제 1

```
...                                  ...
p = malloc (1);                      p = malloc (1);
*p = 123;                            *p = 123;
bar();                        →      bar();
a = *p;                              a = *p;
hash_put(h, p, a);                   hash_put(h, p, 123);
...                                  ...
```

- p는 bar가 모르는 foo만의 비밀
  - 아직 정수로 캐스팅되지 않았기 때문에 bar가 접근할 길이 없음
- 따라서 bar를 갔다가 와도 여전히 p의 값은 123

증명한 최적화 예제 2

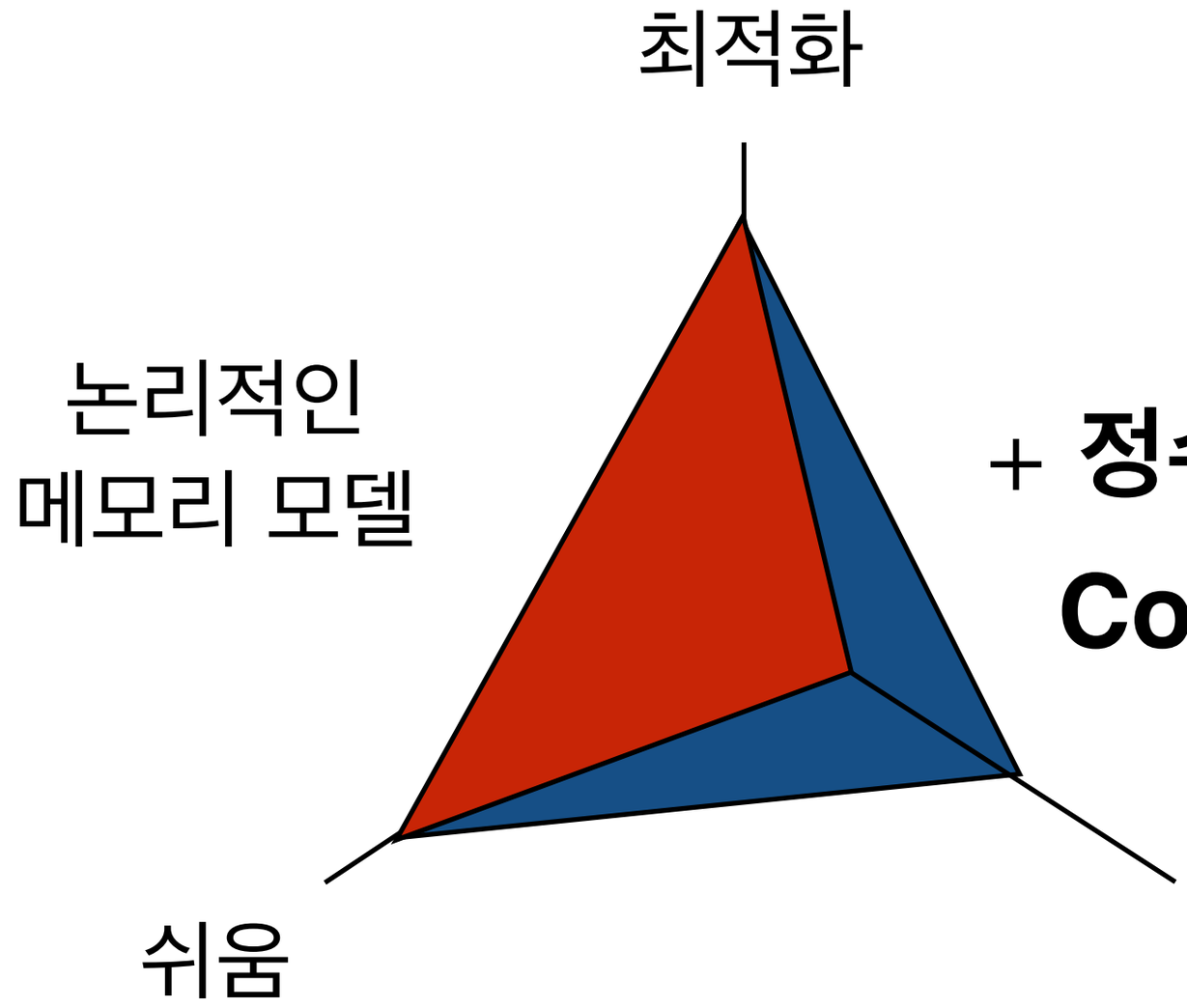
```
   foo(ptr p) {                          foo(ptr p) {
     var ptr q, int a;
1:   q = malloc (1);                                    // DAE
2:   *q = 123;                                          // DSE
3:   bar(p);                                             bar(p);
4:   a = *q;                                                        // DLE
5:   *p = a;                                             *p = 123; // CP
   }                                                   }
```

- q는 bar가 foo만의 모르는 비밀 (캐스팅 되지 않았으므로)
- 마치 bar(p)가 없는 것처럼 최적화 가능

증명한 최적화 예제 3

- 정수 연산 최적화
- 안쓰는 코드 지우기 (dead code elimination)
- 안쓰는 포인터->정수 캐스팅 지우기
  - 캐스팅이 논리 블락을 물리 블락을 바꿈에도 불구하고!
    자세한 내용은 논문에

결론



+ 정수와 포인터간 변환

Coq으로 최적화 증명

학회에 제출중

“Memory model with **sensible tradeoffs**"

“The characterisation of casts to integers as side-effecting is **particularly compelling**”


תודה
Dankie
Gracias
Спасибо
شكراً
Merci
Takk
Köszönjük
Terima kasih
Grazie
Dziękujemy
Dėkojame
Ďakujeme
Vielen Dank
Paldies
Kiitos
Täname teid
谢谢
Thank You
Tak
感謝您
Obrigado
Teşekkür Ederiz
감사합니다
Σας Ευχαριστούμ
ขอบคุณ
Bedankt
Děkujeme vám
ありがとうございます
Tack


?


!