Compiler Engineering
CS6383 - Sep 2019
Mini Assignment 1:

A short on Scalar Evolution

INSTRUCTOR: Dr. Ramakrishna Upadrasta

Report By:
Vijay Tadikamalla
CS17BTECH11040

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

SCEV (Scalar Evolution)

In general, by **Scalar Evolution,** we mean how the value of **scalars changes** in a program with the execution of code.
Scalar Evolution in LLVM is an **analysis of the LLVM program** that helps its clients reason about **induction variables**. It does this by mapping SSA values into objects of a SCEV type, and implementing algebra on top of it.

Observations

LLVM uses Scalar Evolution as an analysis pass. This helps LLVM to prepare loops for advanced optimizations and efficient execution.
We try to optimize our LLVM programs using Scalar Evolution after analyzing the features like:

1. **How a scalar value is derived.**
2. **Dependence of a particular scalar value on a different scalar values.**
3. **Type of recurrence**

How a scalar value is derived

```
int foo(int n, int k) {
  int t = 0;
  for (int i = 0; i < n; i++) t = t + k;
  return t;
}
```

Upon applying optimizations, this code is simply reduced to the following equivalent code:

```
int foo(int n, int k) {
  return n * k;
}
```

Dependence of a particular scalar value on a different scalar values

```
void foo(int *a, int n, int k) {
  for (int i = 0; i < n; i++) a[i] = i*k;
}
```

In the above example, we can see the following recrussion:

$$a[i] = a[i-1] + k$$

That is, the value of each element can be easily derived from the value of the previous element.

So, on applying SCEV, we get a code equivalent to the following code:

```
void foo(int *a, int n, int k) {
  for (int i = 0; i < n; i++) a[i] = a[i-1] + k;
}
```

In the above code, we can see significant improvement because we replace each multiplication operation with a less costly addition operation.

Type of recurrence

SCEV can easily optimize large equations by rewriting/folding it into smaller chain recurrences.
We can easily prove the following rules after doing some simple math.

Expression		Rewrite	Example		
$G + \{e,+,f\}$	$\Rightarrow$	$\{G+e,+,f\}$	$12 + \{7,+,3\}$	$\Rightarrow$	$\{19,+,3\}$
$G * \{e,+,f\}$	$\Rightarrow$	$\{G*e,+,G*f\}$	$12 * \{7,+,3\}$	$\Rightarrow$	$\{84,+,36\}$
$\{e,+,f\} + \{g,+,h\}$	$\Rightarrow$	$\{e+g,+,f+h\}$	$\{7,+,3\} + \{1,+,1\}$	$\Rightarrow$	$\{8,+,4\}$
$\{e,+,f\} * \{g,+,h\}$	$\Rightarrow$	$\left\{\begin{matrix} e*g,+, \\ e*h+f*g+f*h, \\ +,2*f*h \end{matrix}\right\}$	$\{0,+,1\} * \{0,+,1\}$	$\Rightarrow$	$\{0,+,1,+,2\}$

For example:

```
void foo(int *p, int n){
  for( int x = 0; x < n; x++)
    p[x] = x*x*x + 2*x*x + 3*x + 7;
}
```

The equivalent SCEV of this code is

$$SCEV = \{7\ ,\ +\ ,\ 6\ ,\ +\ ,\ 10\ ,\ +\ ,\ 6\}$$

Proof:

$$p(x) = x^3 + 2x^2 + 3x + 7$$

$$= \{0,+,1\}^3 + 2 * \{0,+,1\}^2 + 3 * \{0,+,1\} + 7$$

$$= \{0,+,1,+,6,+,6\} + 2*\{0,+,1,+,2\} + \{0,+,3\} + 7$$

$$= \{0,+,1,+,6,+,6\} + \{0,+,2,+,4\} + \{7,+,3\}$$

$$= \{0,+,1,+,6,+,6\} + \{0,+,2,+,4\} + \{7,+,3\}$$

$$= \{0,+,3,+,10,+,6\} + \{7,+,3\}$$

$$= \{7,+,6,+10,+,6\}$$

References

- http://llvm.org/devmtg/2018-04/slides/Absar-ScalarEvolution.pdf
- https://doc.ecoscentric.com/gnutools/doc/gccint/Scalar-evolutions.html
- https://llvm.org/devmtg/2009-10/ScalarEvolutionAndLoopOptimization.pdf
- https://www.playingwithpointers.com/blog/scev-integer-overflow.html
- https://subscription.packtpub.com/book/application_development/9781785280801/5/ch05lvl1sec36/scalar-evolution