CS107: Dataflow Analysis

Guannan Wei
guannan.wei@tufts.edu

March 10, 2026

Spring 2026

Tufts University

Dataflow Analysis

Dataflow analysis computes value-flow information at each program point in the control-flow graph.

- Intra-procedural: analyses that work on a single function, and whose results are only valid for that function.
- Inter-procedural: analyses that work on multiple functions with function calls and returns.
  - Higher-order program analysis where functions are permitted as first-class values.

A First Example: Available Expressions

The following C program fragment sets `r` to x^y for $y > 0$.

```
1     int y1 = 1;
2     int r = x;
3     while (y1 != y) {
4       int t = y1 * 2;
5       if (t <= y) {
6         r = r * r;
7         y1 = y1 * 2;
8       } else {
9         r = r * x;
10        y1 = y1 + 1;
11      }
12    }
```

Example: `x=2` and `y=5`, then `r=2^5=32` at the end of the program.

A First Example: Available Expressions

The following C program fragment sets `r` to x^y for $y > 0$.

```
1     int y1 = 1;
2     int r = x;
3     while (y1 != y) {
4       int t = y1 * 2;
5       if (t <= y) {
6         r = r * r;
7         y1 = y1 * 2;
8       } else {
9         r = r * x;
10        y1 = y1 + 1;
11      }
12    }
```

How can it be (slightly) optimized?

CSE

The following C program fragment sets `r` to x^y for $y > 0$.

```
1     int y1 = 1;
2     int r = x;
3     while (y1 != y) {
4       int t = y1 * 2;
5       if (t <= y) {
6         r = r * r;
7         y1 = y1 * 2; // y1 * 2 can be replaced by t
8       } else {
9         r = r * x;
10        y1 = y1 + 1;
11      }
12    }
```

How can it be (slightly) optimized?

Available Expressions

Why is the previous optimization valid?

Available Expressions

Why is the previous optimization valid?

- At line 7, where expression `y1*2` appears for the second time, it is **available**.
- That is, no matter how we reach line 7, `y1*2` will have been computed previously at line 4.
- The computation of line 4 is still valid at line 7 because no redefinition of `y1` appears between those two points.

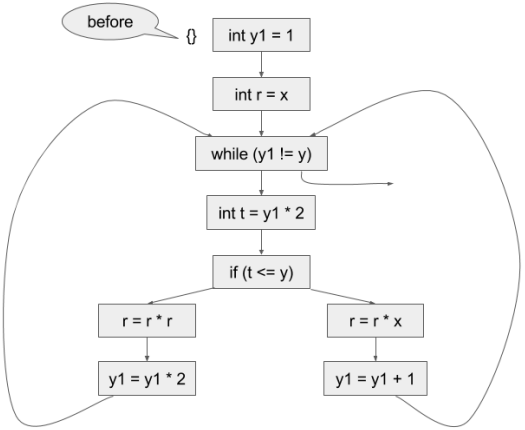
Available Expressions

Why is the previous optimization valid?

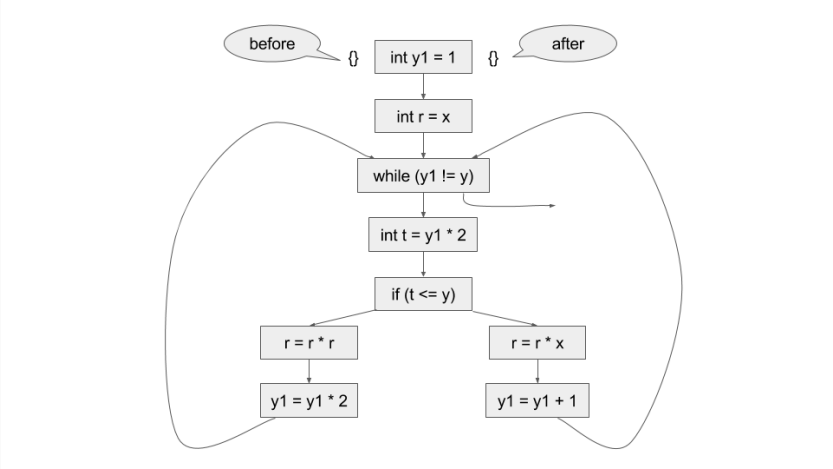
- At line 7, where expression `y1*2` appears for the second time, it is **available**.
- That is, no matter how we reach line 7, `y1*2` will have been computed previously at line 4.
- The computation of line 4 is still valid at line 7 because no redefinition of `y1` appears between those two points.

Generally speaking, we can define for every program point the set of **available expressions**, which is the set of all *nontrivial expressions* whose value has already been computed at that point.

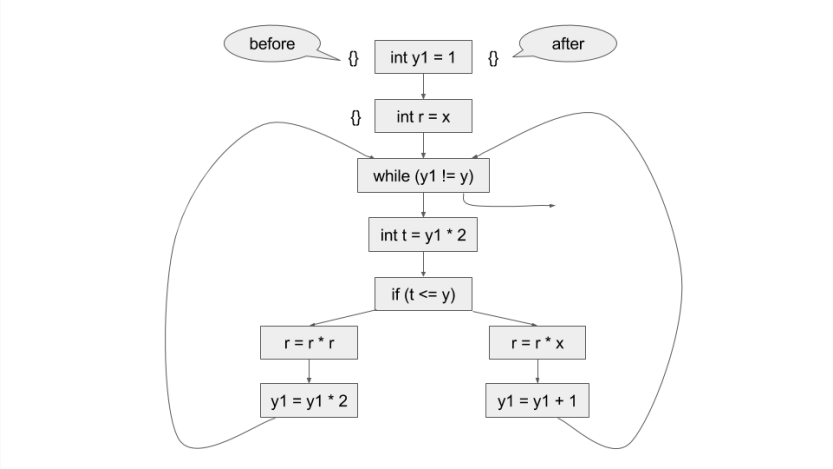
Available Expressions



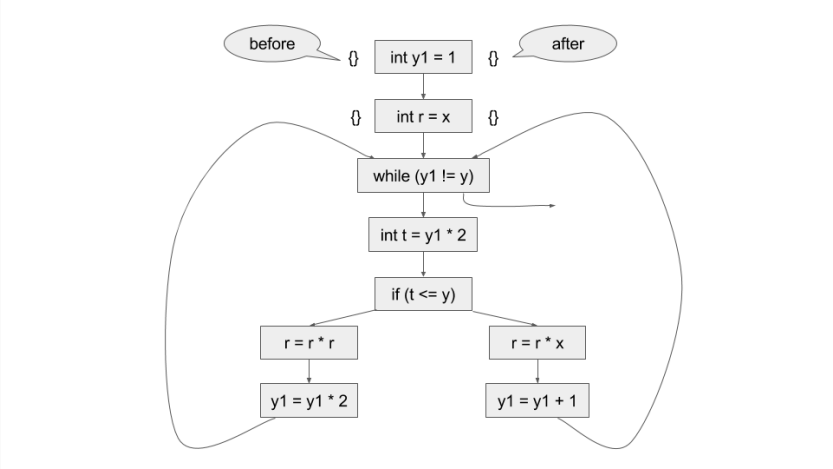
Available Expressions



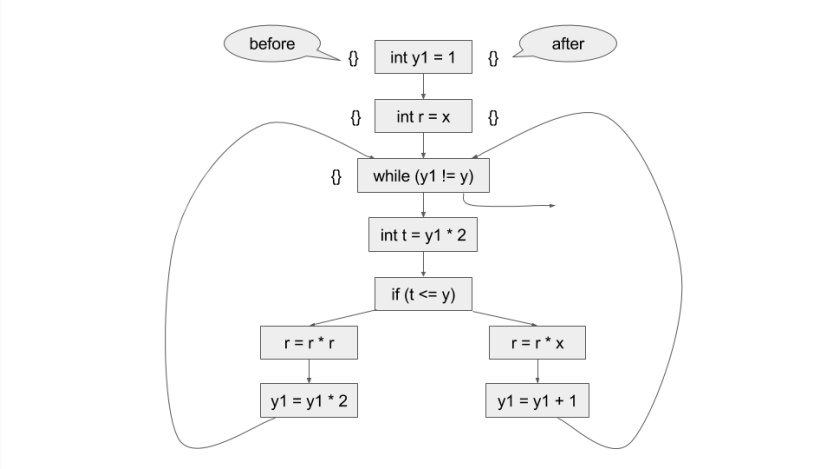
Available Expressions



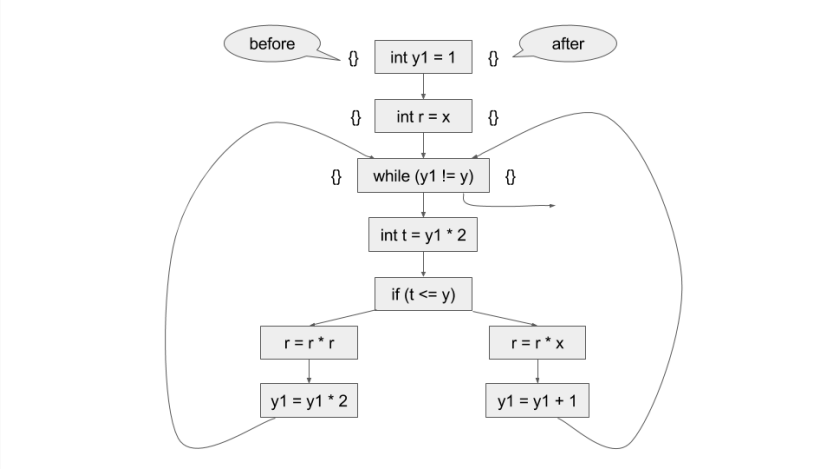
Available Expressions



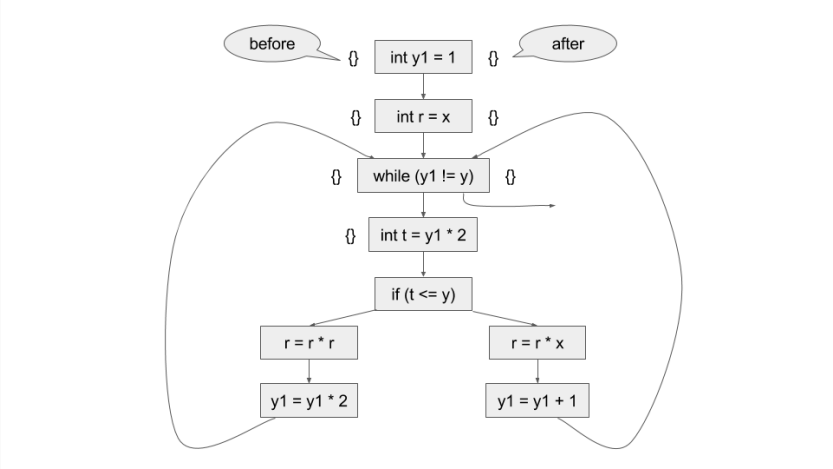
Available Expressions



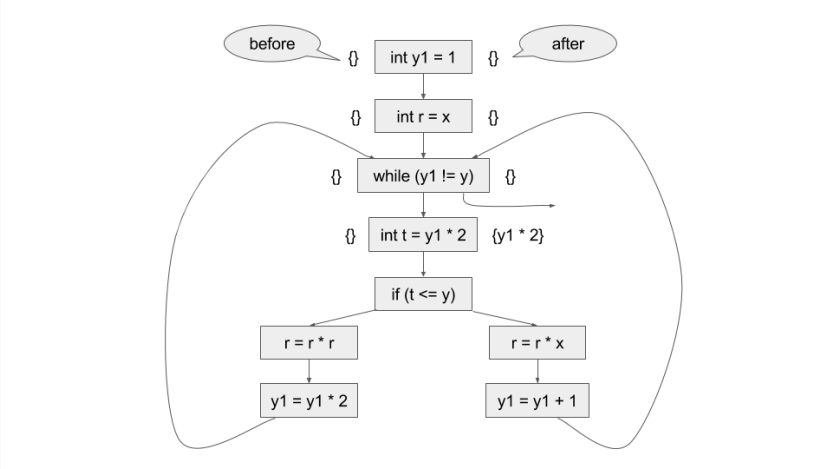
Available Expressions



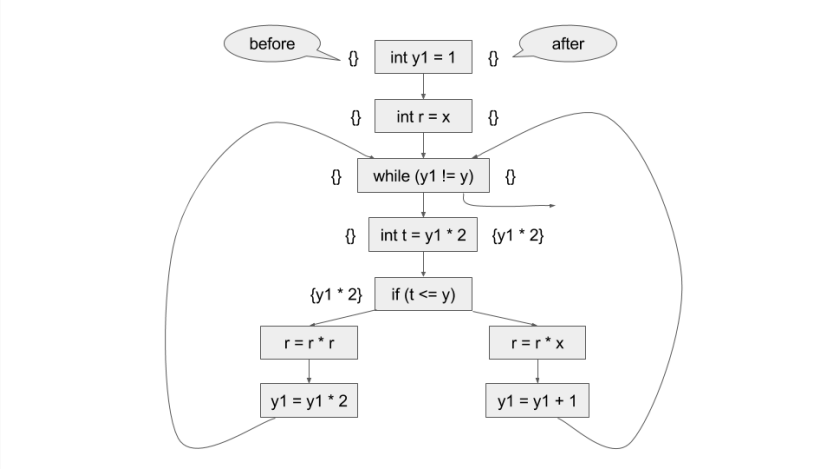
Available Expressions



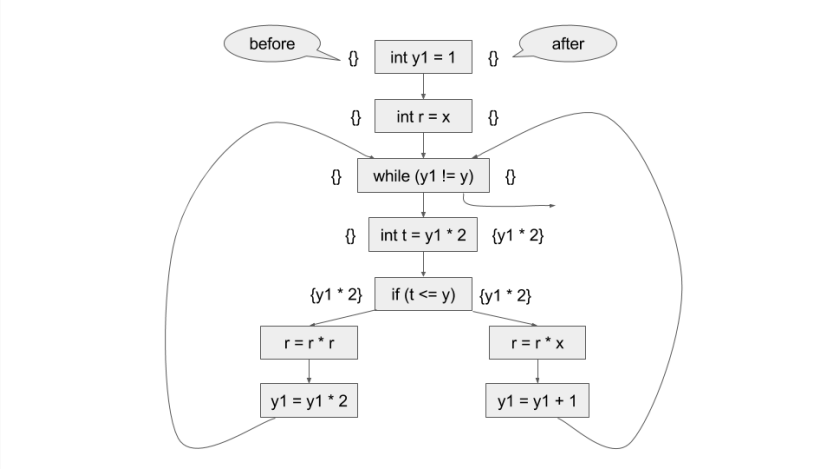
Available Expressions



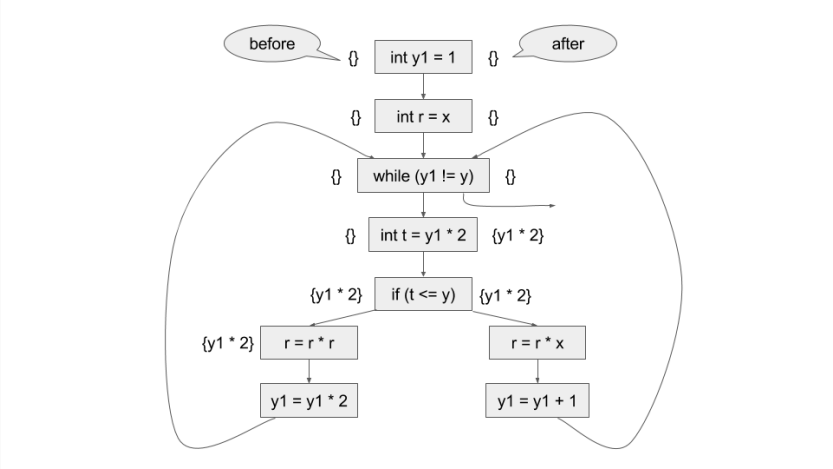
Available Expressions



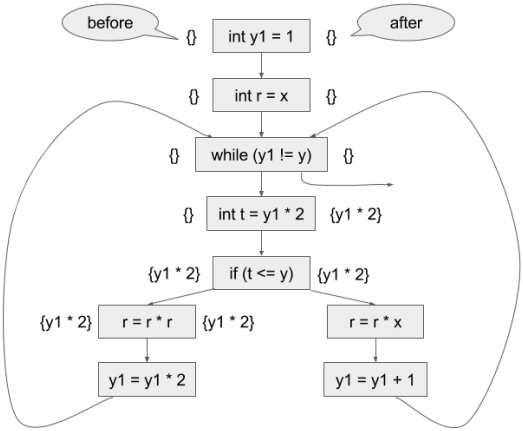
Available Expressions



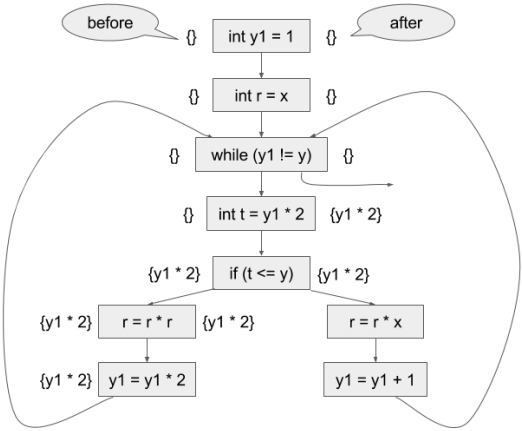
Available Expressions



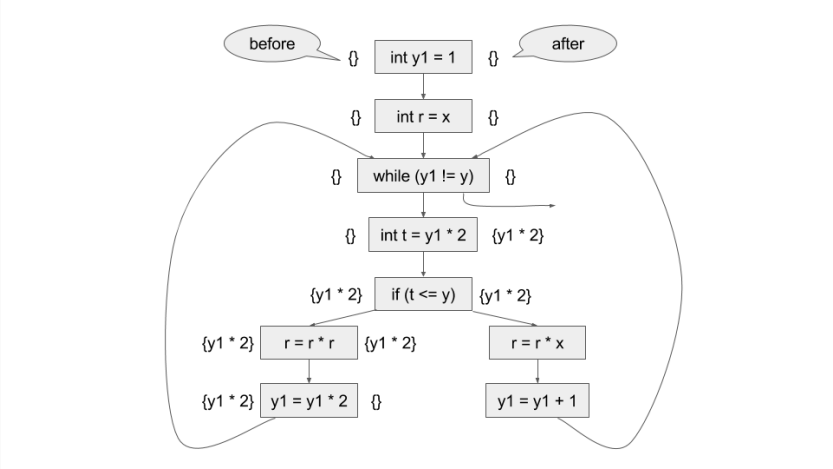
Available Expressions



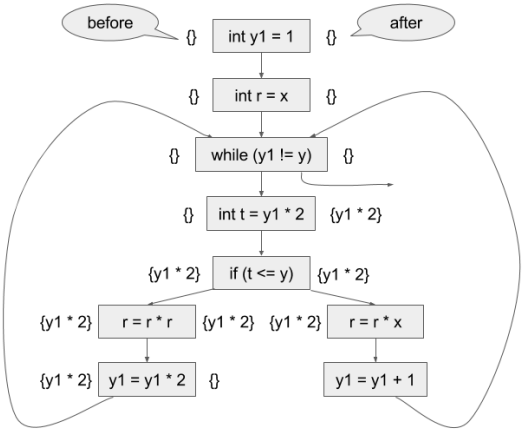
Available Expressions



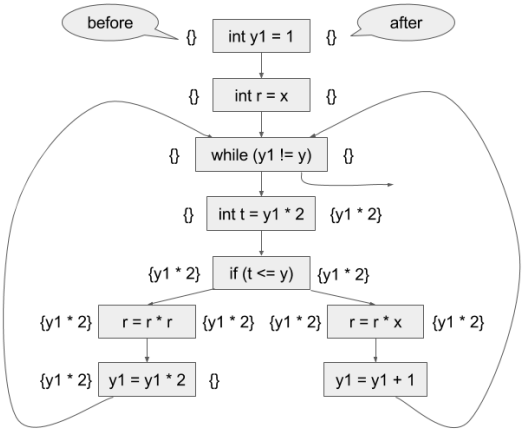
Available Expressions



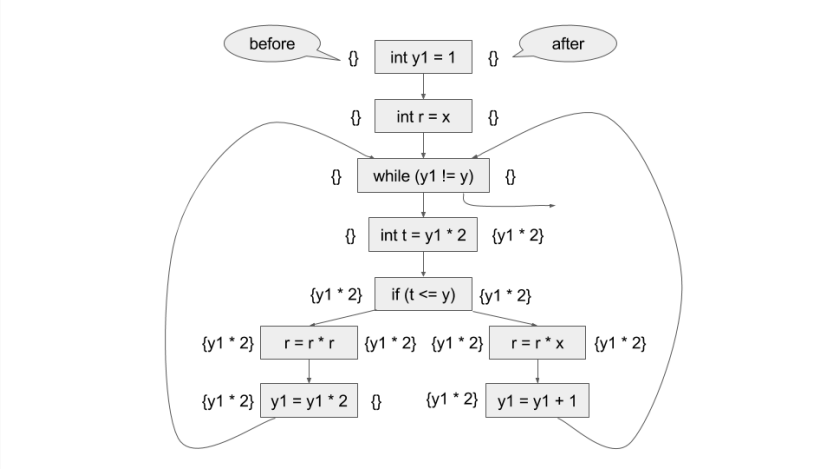
Available Expressions



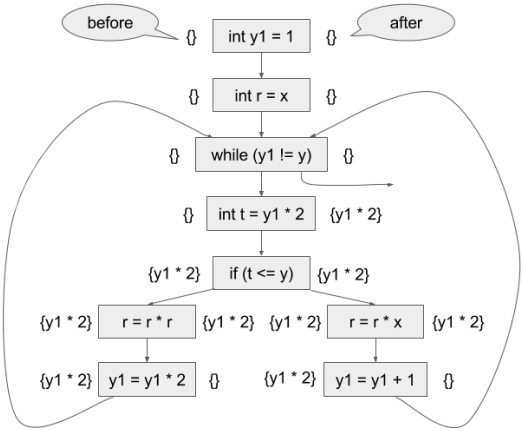
Available Expressions



Available Expressions



Available Expressions

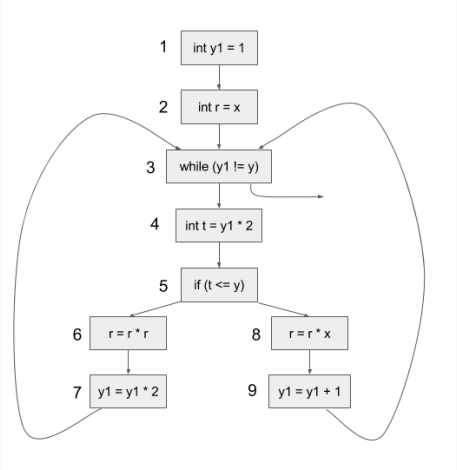


Formalizing The Analysis

How can these ideas be formalized?

1) introduce a variable i_n for the set of expressions available before node n, and a variable o_n for the set of expressions available after node n,
2) define equations between those variables,
3) solve those equations.

Equations



$$
\begin{aligned}
i_1 &= \{\} & o_1 &= i_1 \\
i_2 &= o_1 & o_2 &= i_2 \\
i_3 &= o_2 \cap o_7 \cap o_9 & o_3 &= i_3 \\
i_4 &= o_3 & o_4 &= \{\ \text{y1 * 2}\ \} \cup i_4 \\
i_5 &= o_4 & o_5 &= i_5 \\
i_6 &= o_5 & o_6 &= i_6 \downarrow \text{r} \\
i_7 &= o_6 & o_7 &= i_7 \downarrow \text{y1} \\
i_8 &= o_5 & o_8 &= i_8 \downarrow \text{r} \\
i_9 &= o_8 & o_9 &= i_9 \downarrow \text{y1}
\end{aligned}
$$

Notation: $S \downarrow x = S \setminus \{$ all expressions using x $\}$

Solving Equations

The equations can be solved by iteration:

- initialize all sets i_1, …, i_9, o_1, …, o_9 to the set of **all** nontrivial expressions in the program, here $\{$`y1*2, y1+1, r*r, r*x`$\}$,
- viewing the equations as assignments, compute the "new" value of those sets,
- iterate until a **fixed point** is reached.

Solving Equations

The equations can be solved by iteration:

- initialize all sets i_1, …, i_9, o_1, …, o_9 to the set of **all** nontrivial expressions in the program, here $\{$`y1*2, y1+1, r*r, r*x`$\}$,
- viewing the equations as assignments, compute the "new" value of those sets,
- iterate until a **fixed point** is reached.

Why initializing to all nontrivial expressions?

Solving Equations

The equations can be solved by iteration:

- initialize all sets i_1, …, i_9, o_1, …, o_9 to the set of **all** nontrivial expressions in the program, here $\{$`y1*2, y1+1, r*r, r*x`$\}$,
- viewing the equations as assignments, compute the "new" value of those sets,
- iterate until a **fixed point** is reached.

Why initializing to all nontrivial expressions?

- Because we are interested in finding the largest sets satisfying the equations: the larger a set is, the more information it conveys (for this Analysis).

Solving Equations

To simplify the equations, we can first replace all i_k variables by their values, to obtain a simpler system, and then solve that system.

For our example, we obtain the following equations about the "out'' facts:

$$
\begin{aligned}
\mathsf{o}_1 &= \{\} & \mathsf{o}_6 &= \mathsf{o}_5 \downarrow \mathsf{r} \\
\mathsf{o}_2 &= \mathsf{o}_1 & \mathsf{o}_7 &= \mathsf{o}_6 \downarrow \mathsf{y1} \\
\mathsf{o}_3 &= \mathsf{o}_2 \cap \mathsf{o}_7 \cap \mathsf{o}_9 & \mathsf{o}_8 &= \mathsf{o}_5 \downarrow \mathsf{r} \\
\mathsf{o}_4 &= \mathsf{o}_3 \cup \{\ \mathsf{y1} * 2\ \} & \mathsf{o}_9 &= \mathsf{o}_8 \downarrow \mathsf{y1} \\
\mathsf{o}_5 &= \mathsf{o}_4
\end{aligned}
$$

Solving Equations

The simpler system can be solved by iterating until a fixed point is reached, which happens after 7 iterations.

it.	1	2	3	4	5	6	7
o_1	$\top$	{}	{}	{}	{}	{}	{}
o_2	$\top$	$\top$	{}	{}	{}	{}	{}
o_3	$\top$	$\top$	R	{}	{}	{}	{}
o_4	$\top$	$\top$	$\top$	{y1*2, r*r, r*x}	{y1*2}	{y1*2}	{y1*2}
o_5	$\top$	$\top$	$\top$	$\top$	{y1*2, r*r, r*x}	{y1*2}	{y1*2}
o_6	$\top$	Y	Y	Y	Y	{y1*2}	{y1*2}
o_7	$\top$	R	{}	{}	{}	{}	{}
o_8	$\top$	Y	Y	Y	Y	{y1*2}	{y1*2}
o_9	$\top$	R	{}	{}	{}	{}	{}

Notation: $Y = \{\texttt{y1*2},\ \texttt{y1+1}\}$, $R = \{\texttt{r*r},\ \texttt{r*x}\}$, $\top = Y \cup R$