If this parallelism is not yet sufficient, we could try to speculatively execute possibly useful tasks ...

For that, we require:

- an idea which alternative is executed more frequently;
- the wrong execution may not end in a catastrophy, i.e., run-time errors such as, e.g., division by 0;
- the wrong execution must allow roll-back (e.g., by delaying a commit) or may not have any observational effects ...

... in the Example:

	$z = x - 1$	$y = M[A]$	if $(x > 0)$ goto B
	$y = M[A+1]$		
$B:$			
	$y = y + 1$		

In the case $x \leq 0$ we have $y = M[A]$ executed in advance.

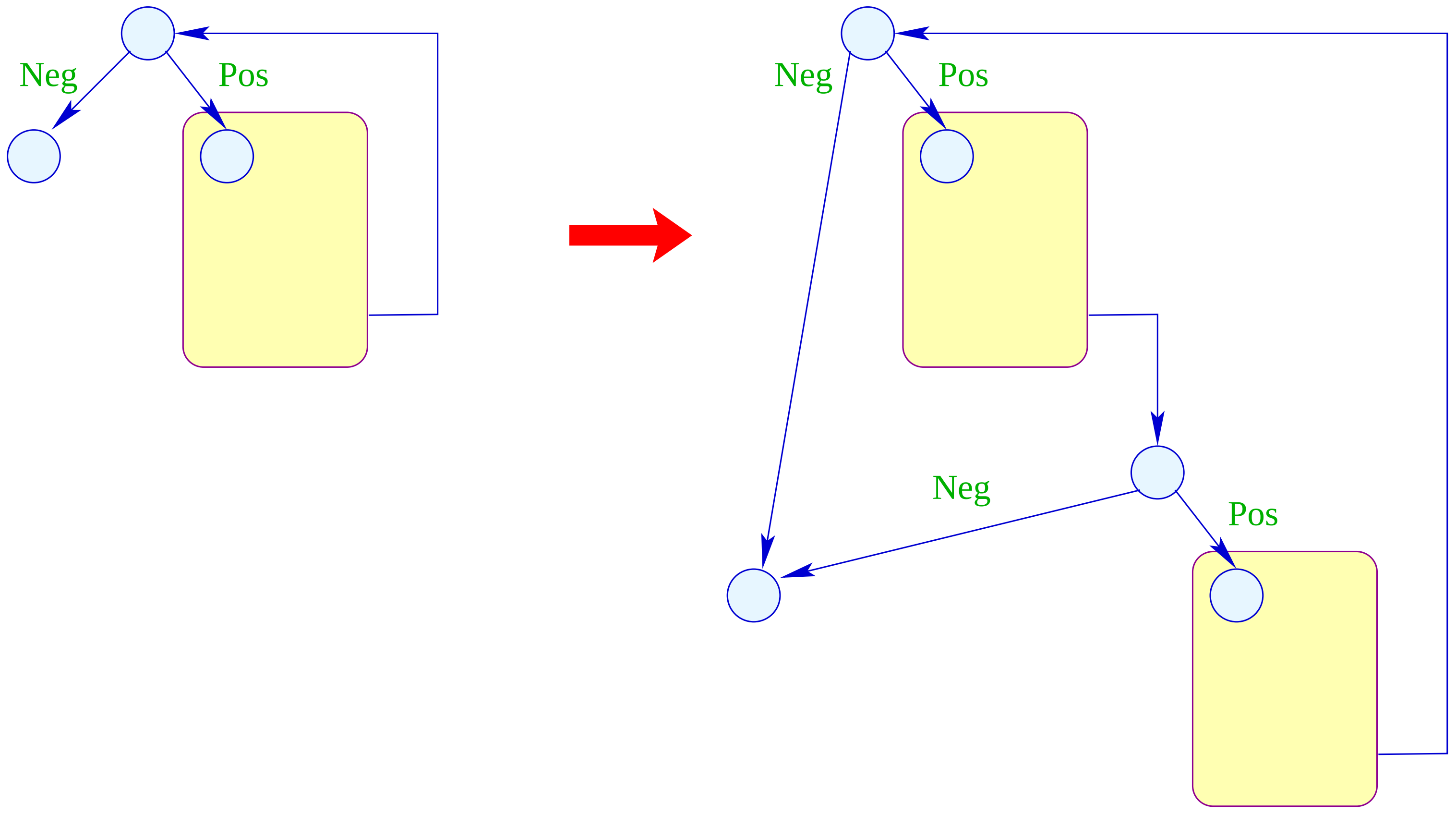
This value, however, is overwritten in the next step :-)

In general:

$x = e;$ has no observable effect in a branch if x is dead in this branch :-)

Extension 2: Unrolling of Loops

We may unrole important, i.e., inner loops several times:



Now it is clear which side of tests to prefer:

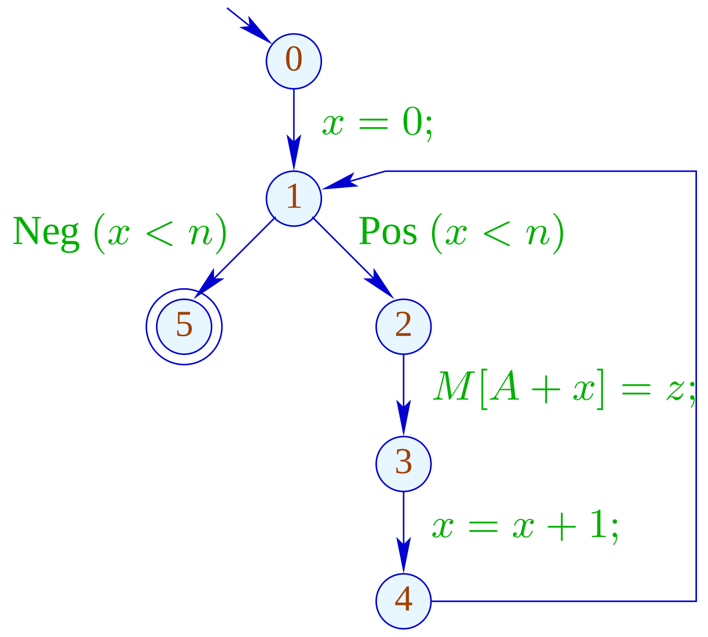
the side which stays within the unroled body of the loop :-)

Warning:

- The different instances of the body are translated relative to possibly different initial states :-)
- The code behind the loop must be correct relative to the exit state corresponding to every jump out of the loop!

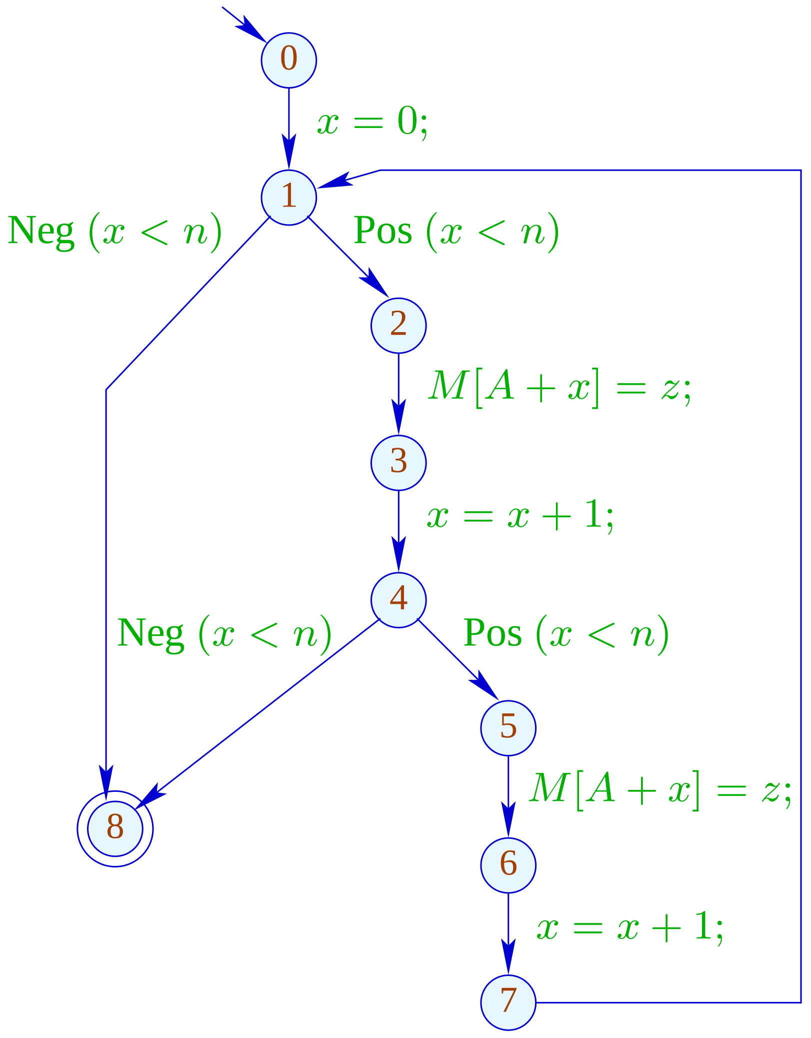
Example:

$$\textsf{for}\ \ (x = 0; x < n; x{+}{+})$$
$$\qquad M[A + x] = z;$$



Duplication of the body yields:

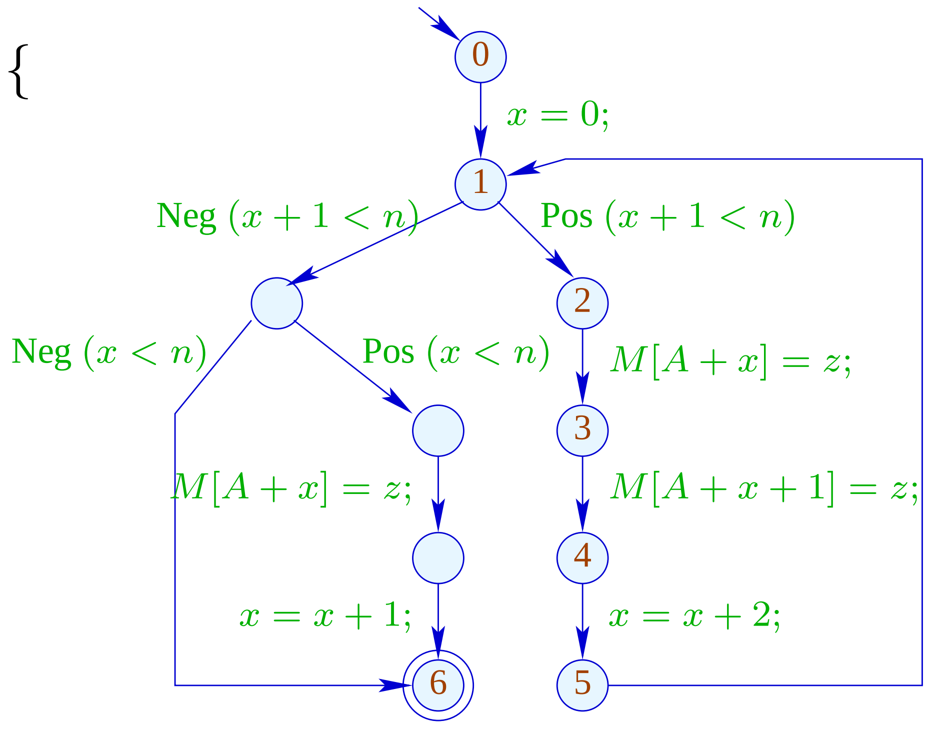
```
for $(x = 0; x < n; x{+}{+})$ {
        $M[A + x] = z;$
        $x = x + 1;$
        if $(!(x < n))$ break;
        $M[A + x] = z;$
        }
```



It would be better if we could remove the assignment $x = x + 1;$ together with the test in the middle — since these serialize the execution of the copies !!

This is possible if we substitute $x + 1$ for x in the second copy, transform the condition and add a compensation code:

```
for  $(x = 0; x + 1 < n; x = x + 2)$  {
        $M[A + x] = z;$
        $M[A + x + 1] = z;$
        }
if  $(x < n)$  {
        $M[A + x] = z;$
        $x = x + 1;$
        }
```



Discussion:

- Elimination of the intermediate test together with the the fusion of all increments at the end reveals that the different loop iterations are in fact independent :-)
- Nonetheless, we do not gain much since we only allow one store per word :-(
- If right-hand sides, however, are more complex, we can interleave their evaluation with the stores :-)

Extension 3:

Sometimes, one loop alone does not provide enough opportunities for parallelization :-(

... but perhaps two successively in a row :-)

Example:

```
for (x = 0; x < n; x++) {            for (x = 0; x < n; x++) {
        R = B[x];                            R = B[x];
        S = C[x];                            S = C[x];
        T_1 = R + S;                         T_2 = R - S;
        A[x] = T_1;                          C[x] = T_2;
        }                                    }
```

In order to fuse two loops into one, we require that:

- the iteration schemes coincide;
- the two loops access different data.

In case of individual variables, this can easily be verified.

This is more difficult in presence of arrays.

Taking the source program into account, accesses to distinct statically allocated arrays can be identified.

An analysis of accesses to the same array is significantly more difficult ...

Assume that the blocks A, B, C are distinct.

Then we can combine the two loops into:

```
for $(x = 0; x < n; x{+}{+})$ {
        $R = B[x];$                              $R = B[x];$
        $S = C[x];$                              $S = C[x];$
        $T_1 = R + S;$                           $T_2 = R - S;$
        $A[x] = T_1;$                            $C[x] = T_2;$
                                                 }
```

The first loop may in iteration x not read data which the second loop writes to in iterations $< x$.

The second loop may in iteration x not read data which the first loop writes to in iterations $> x$.

If the index expressions of jointly accessed arrays are linear, the given constraints can be verified through integer linear programming ...

$$\begin{array}{rcl} i & \geq & 0 \\ i & \leq & x - 1 \end{array} \qquad \begin{array}{rcl} x_{\text{write}} & = & i \\ x_{\text{read}} & = & x \\ x_{\text{read}} & = & x_{\text{write}} \end{array}$$

// x_{read} read access to C by 1st loop
// x_{write} write access to C by 2nd loop

... obviously has no solution :-)

General Form:

$$\begin{aligned} s &\geq t_1 \\ t_2 &\geq s \\ y_1 &= s_1 \\ y_2 &= s_2 \\ y_1 &= y_2 \end{aligned}$$

for linear expressions s, t_1, t_2, s_1, s_2 over i and the iteration variables.

This can be simplified to:

$$0 \leq s - t_1 \qquad 0 \leq t_2 - s \qquad 0 = s_1 - s_2$$

What should we do with it ???

Simple Case:

The two inequations have no solution over $\mathbb{Q}$.

Then they also have no solution over $\mathbb{Z}$:-)

... in Our Example:

$$\begin{array}{rcl} x & = & i \\ 0 & \leq & i \qquad\qquad = \quad x \\ 0 & \leq & x - 1 - i \quad = \quad -1 \end{array}$$

The second inequation has no solution :-)

Equal Signs:

If a variable x occurs in all inequations with the same sign, then there is always a solution :-(

Example:

$$\begin{array}{lll} 0 & \leq & 13 + 7 \cdot x \\ 0 & \leq & -1 + 5 \cdot x \end{array}$$

The variable x may, e.g., be chosen as:

$$x \geq \mathsf{max}(-\frac{13}{7}, \frac{1}{5}) = \frac{1}{5}$$

Unequal Signs:

A variable x occurs in one inequation negative, in all others positive (if at all). Then a system can be constructed without x ...

Example:

$$\begin{array}{rcl} 0 & \leq & 13 - 7 \cdot x \\ 0 & \leq & -1 + 5 \cdot x \end{array} \quad \Longleftrightarrow \quad \begin{array}{rcl} x & \leq & \frac{13}{7} \\ 0 & \leq & -1 + 5 \cdot x \end{array}$$

Since $0 \leq -1 + 5 \cdot \frac{13}{7}$ the system has at least a rational solution ...

One Variable:

The inequations where x occurs positive, provide lower bounds.

The inequations where x occurs negative, provide upper bounds.

If G, L are the greatest lower and the least upper bound, respectively, then all (integer) solution are in the interval $[G, L]$:-)

Example:

$$\begin{array}{lll} 0 & \leq & 13 - 7 \cdot x \\ 0 & \leq & -1 + 5 \cdot x \end{array} \qquad \Longleftrightarrow \qquad \begin{array}{lll} x & \leq & \frac{13}{7} \\ x & \geq & \frac{1}{5} \end{array}$$

The only integer solution of the system is $x = 1$:-)

Discussion:

- Solutions only matter within the bounds to the iteration variables.
- Every integer solution there provides a conflict.
- Fusion of loops is possible if no conflicts occur :-)
- The given secial cases suffice to solve the case of two variables over $\mathbb{Q}$ and of one variable over $\mathbb{Z}$:-)
- The number of variables in the inequations corresponds to the nesting-depth of for-loops $\Longrightarrow$ in general, is quite small :-)

Discussion:

- Integer Linear Programming (ILP) can decide satisfiability of a finite set of equations/inequations over $\mathbb{Z}$ of the form:

$$\sum_{i=1}^{n} a_i \cdot x_i = b \quad \text{bzw.} \quad \sum_{i=1}^{n} a_i \cdot x_i \geq b \; , \qquad a_i \in \mathbb{Z}$$

- Moreover, a (linear) cost function can be optimized :-)
- Warning: The decision problem is in general, already NP-hard !!!
- Notwithstanding that, surprisingly efficient implementations exist.
- Not just loop fusion, but also other re-organizations of loops yield ILP problems ...

Background 5: Presburger Arithmetic

Many problems in computer science can be formulated without multiplication :-)

Let us first consider two simple special cases ...

1. Linear Equations

$$\begin{array}{rcrcrcr} 2x & + & 3y & & & = & 24 \\ x & - & y & + & 5z & = & 3 \end{array}$$

Question:

- Is there a solution over $\mathbb{Q}$?
- Is there a solution over $\mathbb{Z}$?
- Is there a solution over $\mathbb{N}$?

Let us reconsider the equations:

$$\begin{array}{rcrcrcr} 2x & + & 3y & & & = & 24 \\ x & - & y & + & 5z & = & 3 \end{array}$$

Answers:

- Is there a solution over $\mathbb{Q}$? Yes
- Is there a solution over $\mathbb{Z}$? No
- Is there a solution over $\mathbb{N}$? No

Complexity:

- Is there a solution over $\mathbb{Q}$? Polynomial
- Is there a solution over $\mathbb{Z}$? Polynomial
- Is there a solution over $\mathbb{N}$? NP-hard

Solution Method for Integers:

Observation 1:

$$a_1 x_1 + \ldots + a_k x_k = b \qquad\qquad (\forall\, i: \;\; a_i \neq 0)$$

has a solution iff

$$\text{gcd}\{a_1, \ldots, a_k\} \quad \mid \quad b$$

Example:

$$5y - 10z = 18$$

has no solution over $\mathbb{Z}$:-)