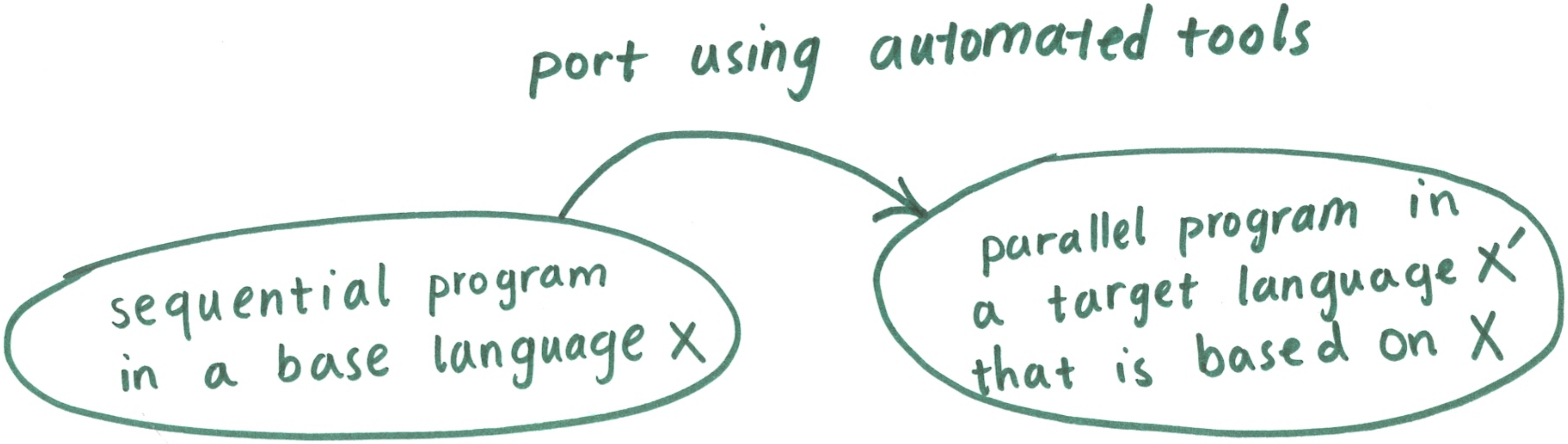
Tools for porting programs to a new parallel programming language

(Feb. 3, 2011)

Mohsen Vakilian (mvakili2@illinois.edu)
(Joint work with Rob Bocchino, Danny Dig, Jeff Overbey, Vikram Adve, Ralph Johnson)



Some parallel programming environments are based on sequential ones.

DPJ

Intel Array Building Block

Microsoft PLINQ

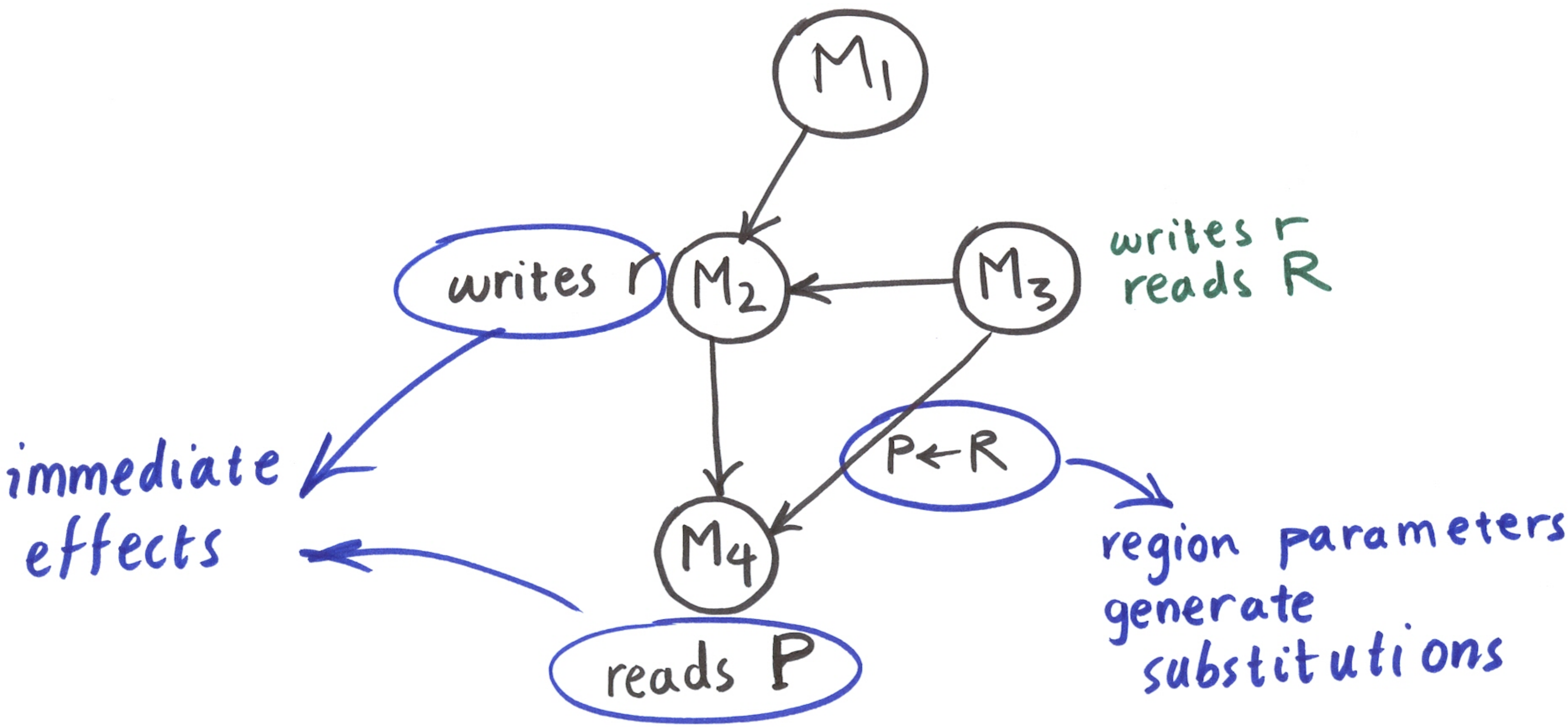
PLINQ extends LINQ

```
IEnumerable<T> data = ...;

var q = data.AsParallel().Where(x => p(x)).
                 Orderby(x => k(x)).
                 Select(x => f(x));

foreach (var e in q) a(e);
```

DPJ has region, RPL, region parameter and effects.

```
                                   region parameter
class Node<region P>{
    region L, R;                           RPL
    double mass in P;
    Node<P:L> left in P:L;
    Node<P:R> right in P:R;
    void setMassForTree(double mass)
effects →  writes P, P:L:*, P:R:* {
        this.mass = mass;
        if (left != null) left.setMassForTree(mass);
        if (right != null) right.setMassForTree(mass);
}
```

It is difficult for programmers or the compiler to infer the effects.



constraint graph

DPJizer infers effects better than programmers.

writes P, P:L:*, P:R:*	instead of	writes P:*
writes P	instead of	reads P, writes P

```
class Tree {
    Body<[i]/π1>[]#i/i bodies;

    void computegrav() {
        foreach (int j in 0, bodies.length) {
            HGStruct<[j]/π2> hg = new HGStruct<[j]/π3>();
            bodies[j].hackgrav(hg);
        }
    }
}

class Body<region BR> {
    Vector<BR/π4> vel in BR = new Vector<BR/π5>();

    void hackgrav(HGStruct<BR/π6> hg) {
        vel.<π̄11>SETV(hg.acc0);
    }
}
```

```
class Vector<region VR> {
    double[]<VR/π7>#k/k elts in VR = new double[3]<VR/π8>#l/l;

    <region SR> void SETV(Vector<SR/π9> u) {
        for (int m = 0; m < 3; m++) {
            elts[m] = u.elts[m];
        }
    }
}
class HGStruct<region HR> {
    Vector<HR/π10> acc0 in HR;
}
```

Region inclusion constraints are derived from subtyping rules.

double[]$\langle \pi_7 \rangle$#k elts in VR = new double[3]$\langle \pi_8 \rangle$#l;	$\pi_8 \subseteq \pi_7$
vel.$\langle \pi_{11} \rangle$SETV(hg.acc∅);	$VR[VR \leftarrow \pi_{10}][HR \leftarrow \pi_6] \subseteq VR[VR \leftarrow \pi_9][VR \leftarrow \pi_4][SR \leftarrow \pi_{11}]$
bodies[j].hackgrav(hg);	$HR[HR \leftarrow \pi_2] \subseteq HR[HR \leftarrow \pi_6][BR \leftarrow \pi_1][i \leftarrow j]$
HGStruct$\langle \pi_2 \rangle$hg = new HGStruct$\langle \pi_3 \rangle$();	$\pi_3 \subseteq \pi_2$
Vector$\langle \pi_4 \rangle$ vel in BR = new Vector$\langle \pi_5 \rangle$();	$\pi_5 \subseteq \pi_4$

Parallel statements generate disjointness constraints.

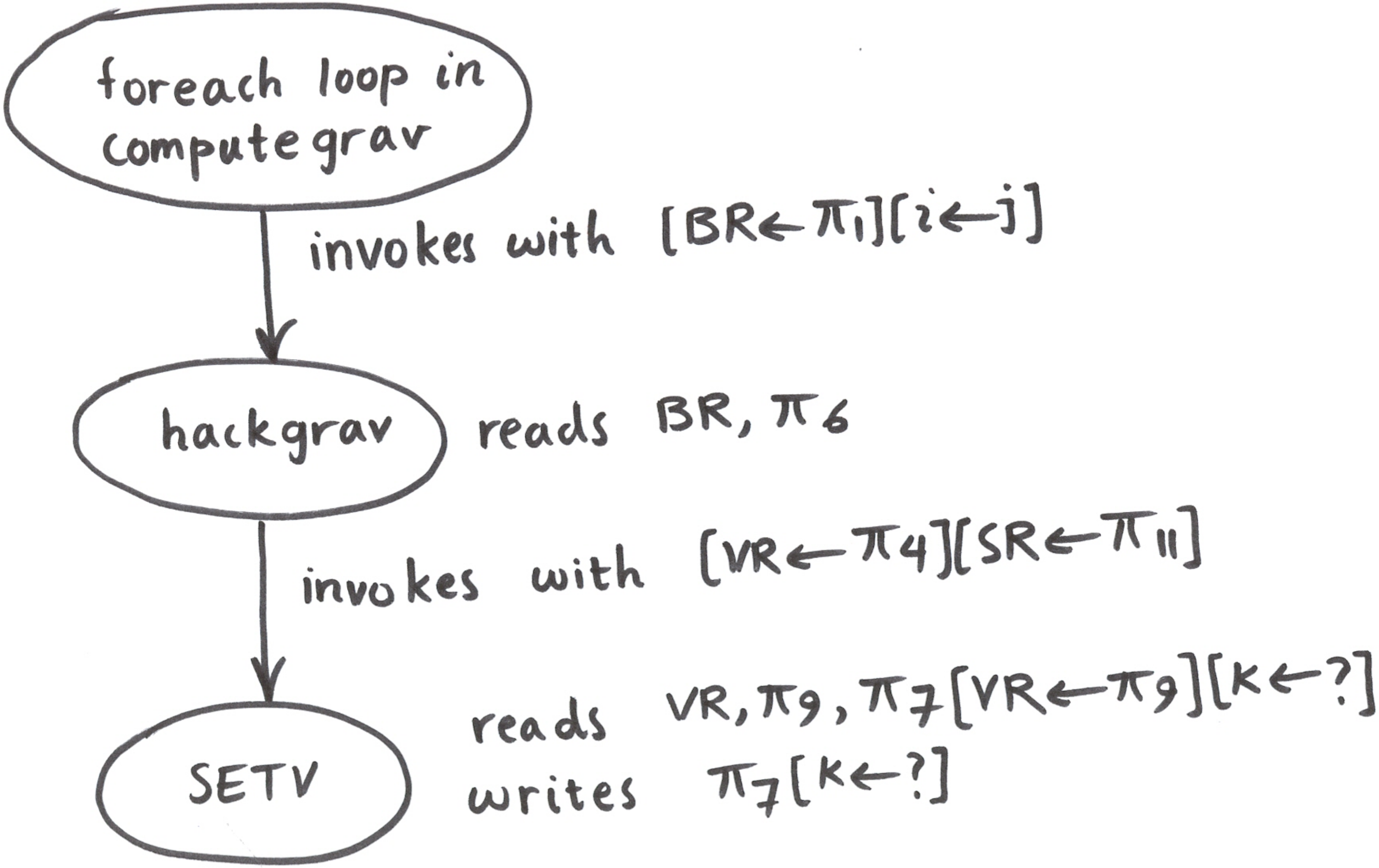
```
foreach ( int j in 0, bodies.length) {
    HGStruct<π2> hg = new HGStruct<π3>();
    bodies[j].hackgrav(hg);
}
```

$$\mathcal{E}(hackgrav)[BR \leftarrow \pi_1][i \leftarrow j][j \leftarrow a]$$

$$\#$$

$$\mathcal{E}(hackgrav)[BR \leftarrow \pi_1][i \leftarrow j][j \leftarrow b]$$

Write effects should be disjoint from each other, and read effects should be disjoint from write effects.



The index variable of the foreach loop should differentiate the write effects.

writes $\pi_7[K \leftarrow ?][VR \leftarrow \pi_4][SR \leftarrow \pi_{11}][BR \leftarrow \pi_1][i \leftarrow j]$

$\pi_7 = VR$ $\pi_4 = BR$ $\pi_1 = [i]$

Read effects should be disjoint from write effects.

reads $\pi_9[VR \leftarrow \pi_4][SR \leftarrow \pi_{11}][BR \leftarrow \pi_1][i \leftarrow j]$

$\pi_9 = VR$ OR $\pi_9 = SR,\ \pi_{11} = BR$

Region inclusion constraints determine the rest of variables.

$\pi_1 = [i]$, $\pi_4 = BR$, $\pi_7 = VR$, $\pi_9 = SR$, $\pi_{11} = BR$

$\pi_8 \subseteq \pi_7$	$\pi_8 = VR$
$VR[VR \leftarrow \pi_{10}][HR \leftarrow \pi_6] \subseteq VR[VR \leftarrow \pi_9][VR \leftarrow \pi_4][SR \leftarrow \pi_{11}]$	$\pi_6 = BR$, $\pi_{10} = HR$
$HR[HR \leftarrow \pi_2] \subseteq HR[HR \leftarrow \pi_6][BR \leftarrow \pi_1][i \leftarrow j]$	$\pi_2 = [j]$
$\pi_3 \subseteq \pi_2$	$\pi_3 = [j]$
$\pi_5 \subseteq \pi_4$	$\pi_5 = BR$

This work is funded by Microsoft and Intel through UPCRC and by NSF grants.

The DPJ homepage is http://dpj.cs.illinois.edu