DPJizer: A Tool for Automated Java to DPJ Transformation

To appear in Automated Software Engineering 2009

Mohsen Vakilian (mvakili2@illinois.edu),
Danny Dig, Robert Bocchino,
Jeffrey Overbey, Vikram Adve,
Ralph Johnson

UPCRC Illinois
Universal Parallel Computing Research Center

An Effect System

```
public FileWriter(String)
throws IOException
```


UPCRC Illinois
Universal Parallel Computing Research Center

Effect Systems

- Effect systems are important for reasoning about side effects.
- Effects are enforceable contracts.
- Effects are machine-checkable documentation.
- Effect system enable composability and modular reasoning.

UPCRC Illinois
Universal Parallel Computing Research Center

Effect Systems for Parallel Programming

- Effects systems can be used to reason about correctness of parallel programs.
- **DPJ** = Deterministic Parallel Java

DPJ Effect System

- A **region** is a name for a collection of locations.
- An **effect** is a **read** or **write** operation on a region.
- Compiler checks that parallel tasks are **noninterfering**.

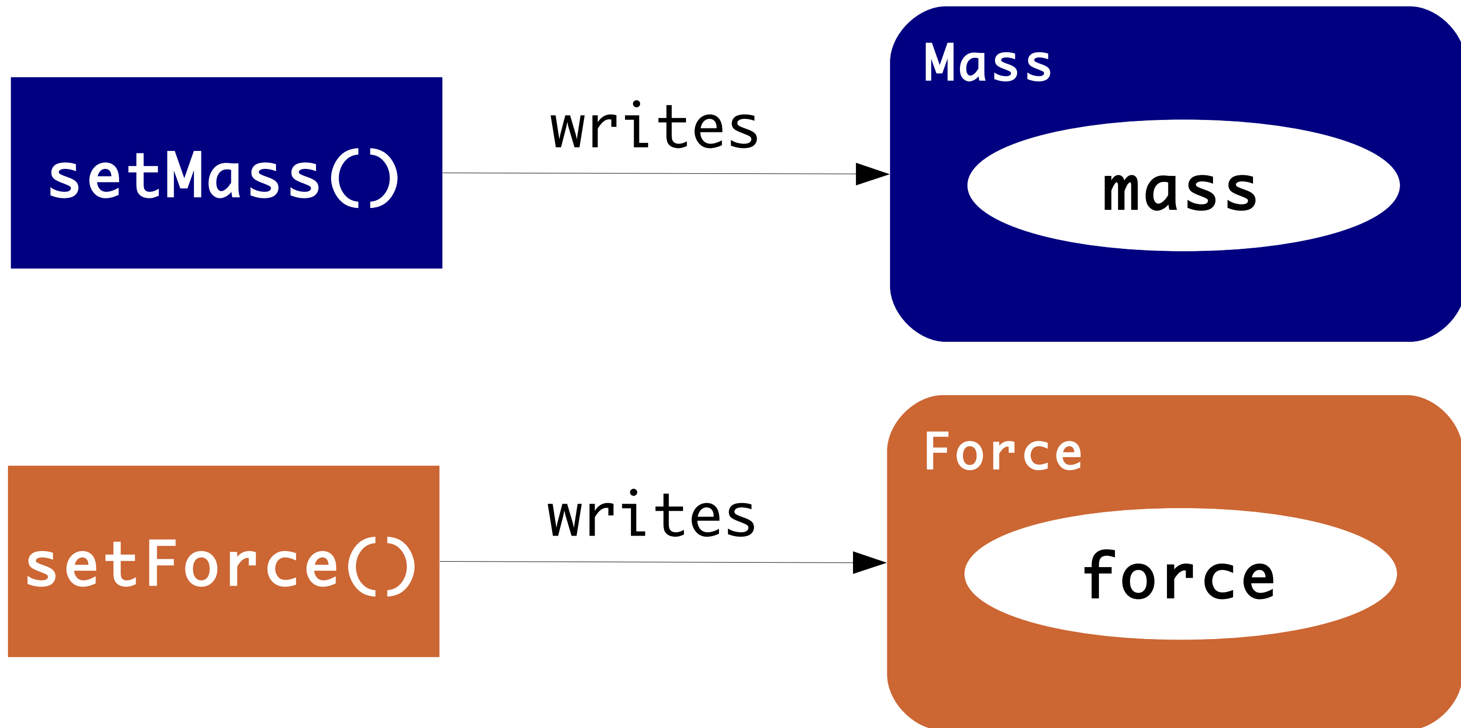
UPCRC Illinois
Universal Parallel Computing Research Center

Region and Effect

- Programmer assigns regions to fields.
- Programmer summarizes effects of method bodies.

UPCRC Illinois
Universal Parallel Computing Research Center

```
double mass in Mass;
double force in Force;
setMass(double) writes Mass;
setForce(double) writes Force;
```



Why have not effect systems taken off yet?

- Writing annotations manually is **error-prone** and **tedious**.
- Whenever the code changes annotations have to be updated.
- On average, programmers annotate **8%** of lines of DPJ source code.

UPCRC Illinois
Universal Parallel Computing Research Center

DPJizer

- DPJizer makes effect systems practical.
- DPJizer infers a summary of effects for each method.

UPCRC Illinois
Universal Parallel Computing Research Center

DPJizer Preview Dialog

Changes to be performed

Infer Effects

BinaryTree.java - TestProject/dpj-programs/01-flat-regions

BinaryTree.java

Original Source

```
1 public class BinaryTree {
2   region L, R, M;
3   BinaryTree left in L;
4   BinaryTree right in R;
5   double mass in M;
6
7   void setMass(double mass) {
8     this.mass = mass;
9     initTree();
10  }
11
12  void initTree() {
13    left = new BinaryTree();
14    right = new BinaryTree();
15  }
```

Refactored Source

```
1 public class BinaryTree {
2   region L, R, M;
3   BinaryTree left in L;
4   BinaryTree right in R;
5   double mass in M;
6
7   void setMass(double mass) writes Root : BinaryTree.L, Root : BinaryTree.M, Root : BinaryTree.R {
8     this.mass = mass;
9     initTree();
10  }
11
12  void initTree() writes Root : BinaryTree.L, Root : BinaryTree.R {
13    left = new BinaryTree();
14    right = new BinaryTree();
15  }
```

Preview > | OK | Cancel

UPCRC Illinois

Universal Parallel Computing Research Center

Highlight Source of Effects

```
void setMass(double mass)
                    writes Root : BinaryTree.M,
                    Root : BinaryTree.L,
                    Root : BinaryTree.R {
   this.mass = mass;
   initTree();
}
```


UPCRC Illinois
Universal Parallel Computing
Research Center

Region Names Ex. (1/2)

```
class Node {
    region Mass, Force;
    double mass in Mass;
    double force in Force;

    void setMass(double mass) writes Mass {
        this.mass = mass; // Effect: writes Mass
    }

    void setForce(double force) writes Force {
        this.force = force; // Effect: writes Force
    }
```


UPCRC Illinois
Universal Parallel Computing
Research Center

Region Names Ex. (2/2)

```
    void initialize(double mass, double force)
     writes Mass, Force {
        cobegin {
            this.setMass(mass); // Effect: writes Mass
            this.setForce(force); // Effect: writes Force
        }
    }
}
```

Input: DPJ Program Annotated with Region Information

DPJizer

Constraint Generator

Constraints: reads, writes, and invokes

Constraint Solver

Output: DPJ Program with Region and Effect Annotations

UPCRC Illinois
Universal Parallel Computing Research Center

Constraints in Region Names Ex. (1/2)

```
class Node {
    region Mass, Force;
    double mass in Mass;
    double force in Force;

    void setMass(double mass) {
        this.mass = mass; // Constraint: writes Mass
    }

    void setForce(double force) {
        this.force = force; // Constraint: writes Force
    }
```

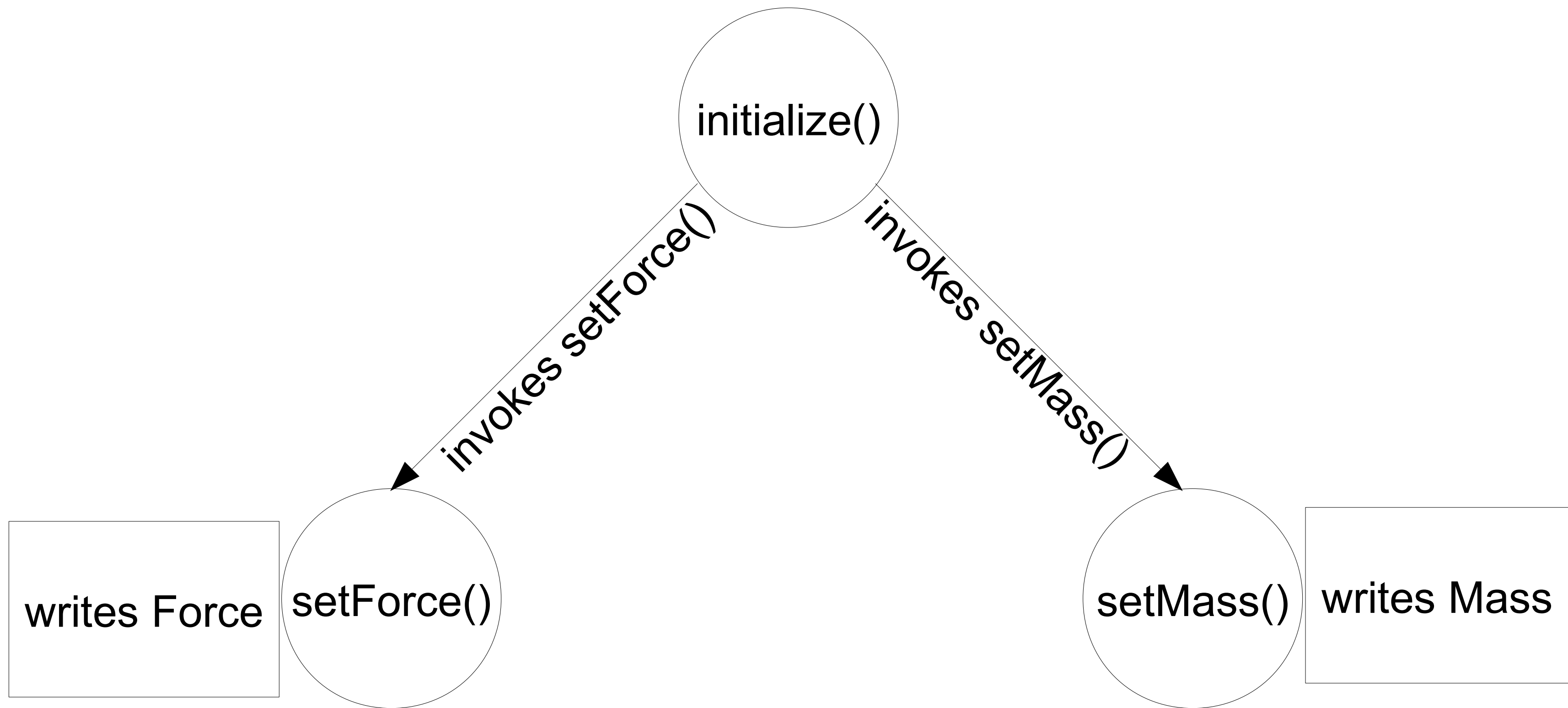
UPCRC Illinois
Universal Parallel Computing Research Center

Constraints in Region Names Ex. (2/2)

```
void initialize(double mass, double force) {
    cobegin {
        // Constraint: invokes setMass()
        this.setMass(mass);

        // Constraint: invokes setForce()
        this.setForce(force);
    }
  }
}
```

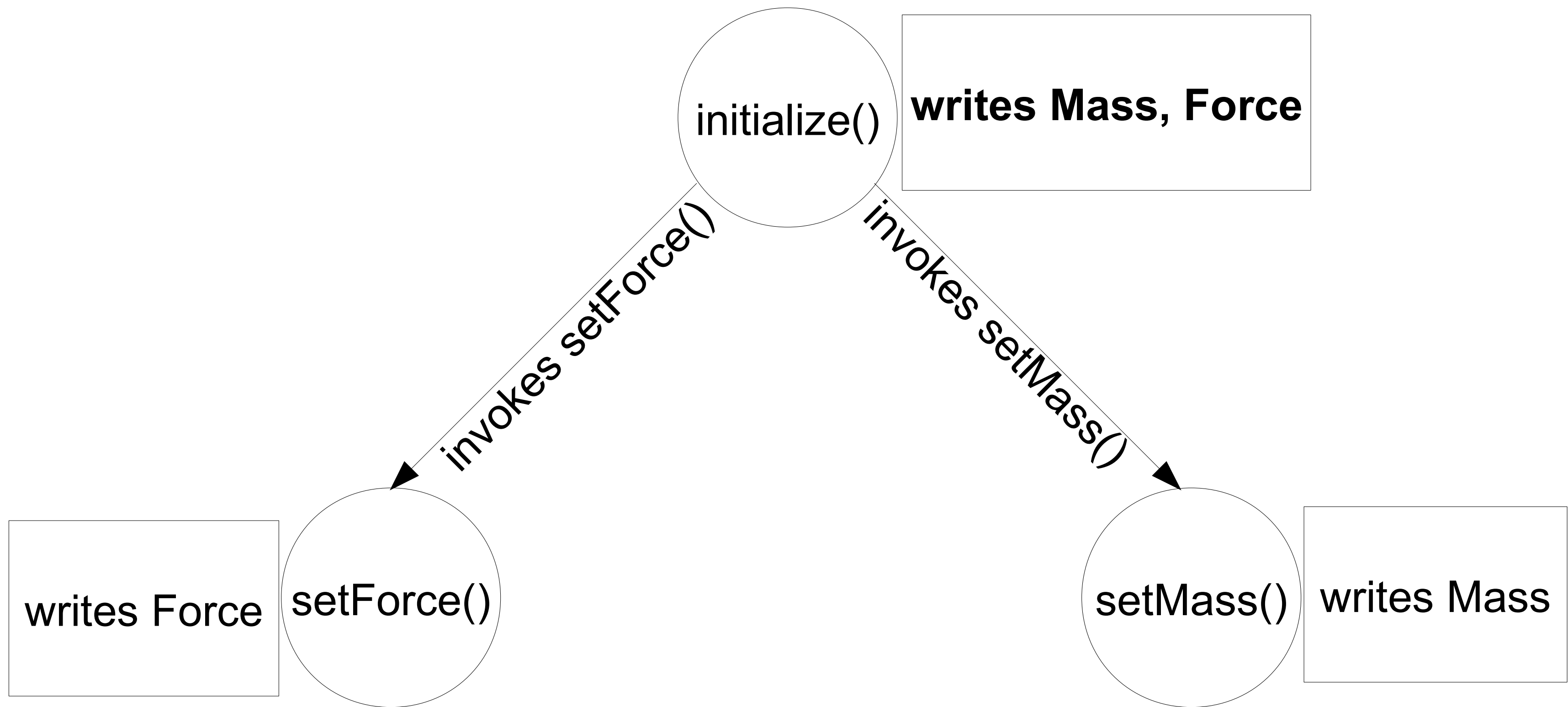
UPCRC Illinois
Universal Parallel Computing
Research Center

Constraint Generation Output



UPCRC Illinois
Universal Parallel Computing Research Center

Constraint Solving



Region Parameters

- Region names distinguish different fields of an object.
- Region parameters distinguish different **object instances**.

UPCRC Illinois
Universal Parallel Computing Research Center

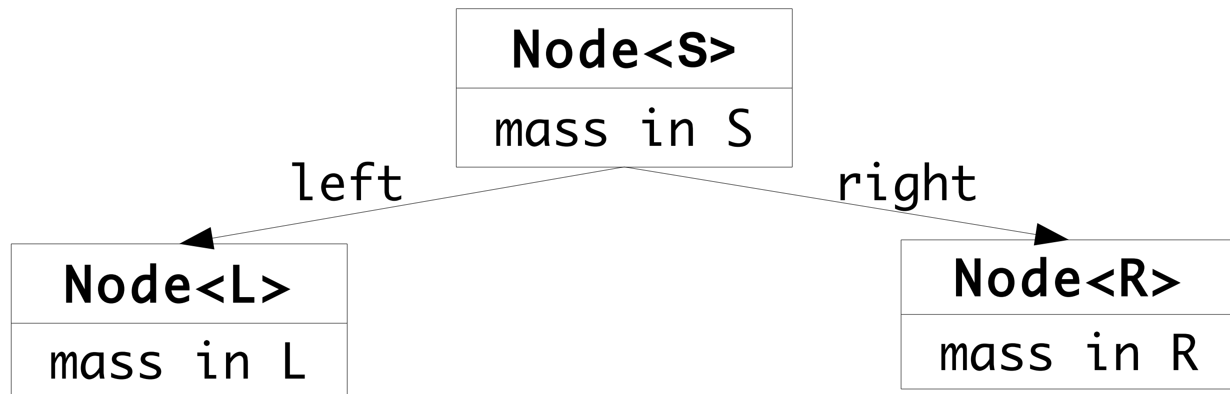
Region Parameters Ex. (1/6)

```
class Node<region P> {
    region L, R;
    double mass in P;
    Node<L> left in L;
    Node<R> right in R;
    //...
}
```

UPCRC Illinois
Universal Parallel Computing Research Center

Region Parameters Ex. (2/6)

```
class Node<region P> {
    region L, R;
    double mass in P;
    Node<L> left in L;
    Node<R> right in R;
    //...
}
```



UPCRC Illinois
Universal Parallel Computing
Research Center

Region Parameters Ex. (3/6)

```
class Node<region P> {
    region L, R;
    double mass in P;
    Node<L> left in L;
    Node<R> right in R;

    void setMass(double mass) {
        this.mass = mass;
    }
```

UPCRC Illinois
Universal Parallel Computing Research Center

Region Parameters Ex. (4/6)

```
class Node<region P> {
    region L, R;
    double mass in P;
    Node<L> left in L;
    Node<R> right in R;

    void setMass(double mass) writes P {
        this.mass = mass; // Effect: writes P
    }
```

UPCRC Illinois
Universal Parallel Computing Research Center

Region Parameters Ex. (5/6)

```
void setMassOfChildren(double mass) {
    cobegin {
        if (left != null) left.setMass(mass);

        if (right != null) right.setMass(mass);
    }
}
}
```

Region Parameters Ex. (6/6)

```
void setMassOfChildren(double mass) writes L, R {
    cobegin {
        // Effect: writes L
        if (left != null) left.setMass(mass);

        // Effect: writes R
        if (right != null) right.setMass(mass);
    }
  }
}
```


UPCRC Illinois
Universal Parallel Computing Research Center

Extending the Invokes Constraint

- The invokes constraint records region substitutions.
- `C<region P> { /* … */ }`
- `new C<R>().m()`
- `invokes m() where` **`{P → R}`**

UPCRC Illinois
Universal Parallel Computing Research Center

Constraints in Region Parameters Ex. (1/2)

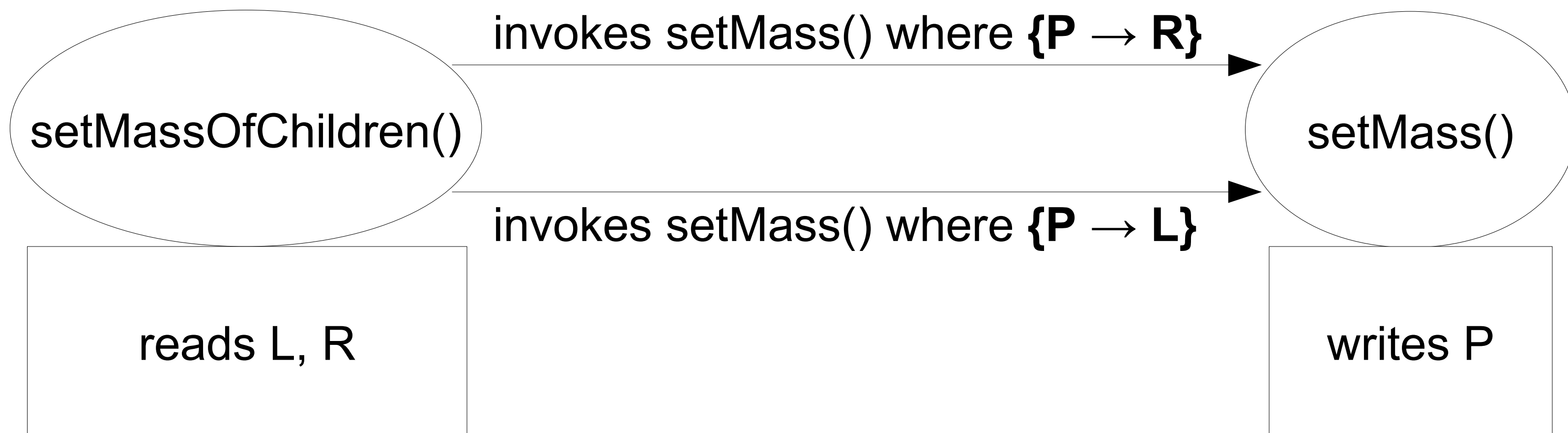
```
class Node<region P> {
    region L, R;
    double mass in P;
    Node<L> left in L;
    Node<R> right in R;

    void setMass(double mass) {
        this.mass = mass; // Constraint: writes P
    }
```

UPCRC Illinois
Universal Parallel Computing Research Center

Constraints in Region Parameters Ex. (2/2)

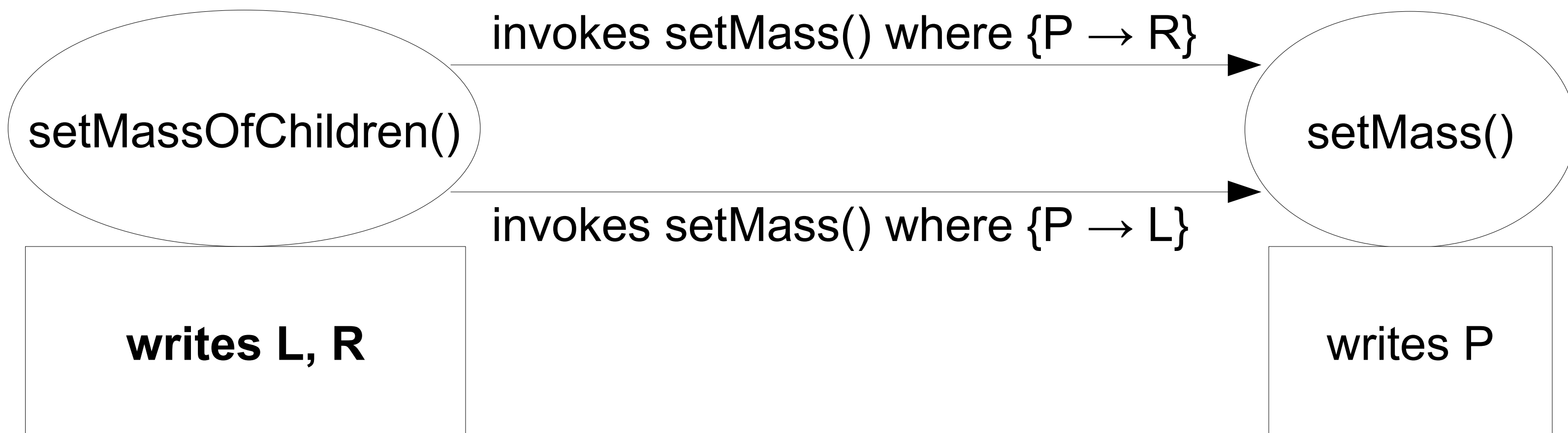
```
void setMassOfChildren(double mass) {
    cobegin {
        // Constraint: reads L,
        //     invokes setMass() where {P → L}
        if (left != null) left.setMass(mass);

        // Constraint: reads R,
        //     invokes setMass() where {P → R}
        if (right != null) right.setMass(mass);
    }
}
}
```

Constraint Generation Output



UPCRC Illinois
Universal Parallel Computing
Research Center

Constraint Solving



UPCRC Illinois
Universal Parallel Computing Research Center

Region Path Lists (RPLs)

- An RPL is a colon separated list of region names e.g. P : L : R : R
- $\{ P : R1 : R2, P : R3 \} \subseteq P : *$
- RPLs are useful to express tree-like recursive updates.

UPCRC Illinois
Universal Parallel Computing Research Center

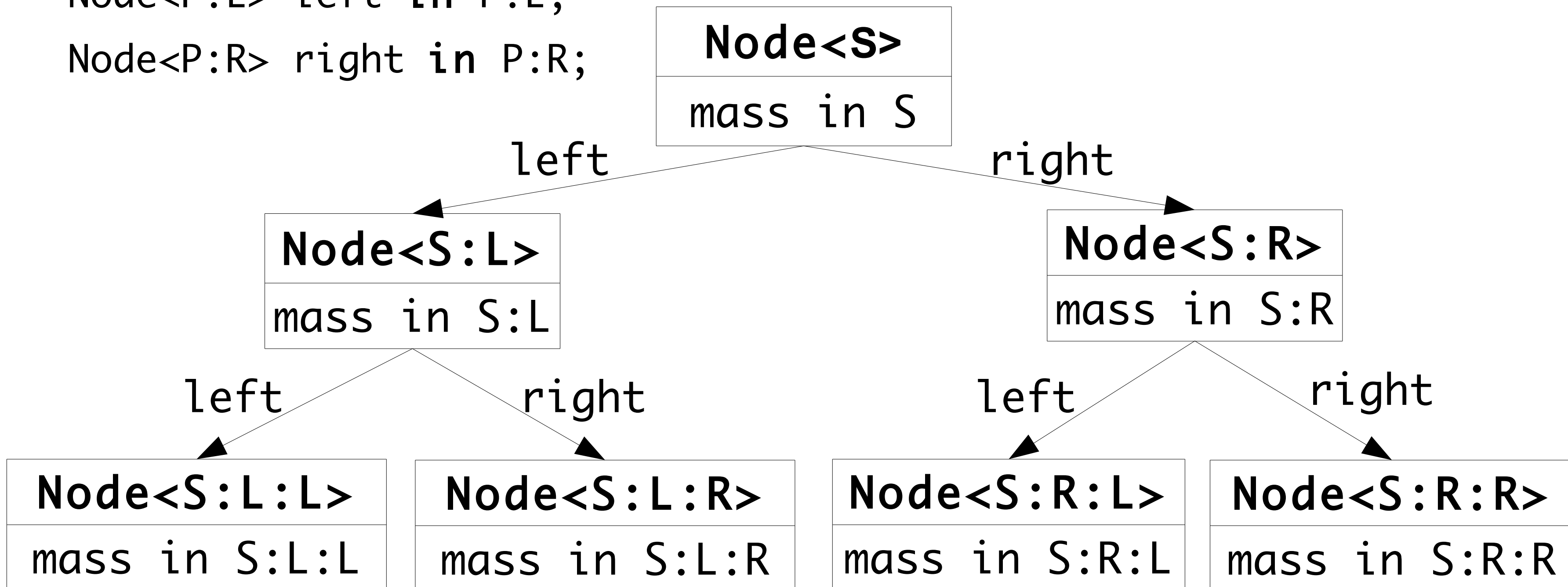
RPL Ex.

```
class Node<region P> {
    region L, R;
    double mass in P;
    Node<P:L> left in P:L;
    Node<P:R> right in P:R;
}
```

UPCRC Illinois
Universal Parallel Computing Research Center

RPL Ex.

```
class Node<region P> {
     region L, R;
     double mass in P;
     Node<P:L> left in P:L;
     Node<P:R> right in P:R;
}
```



RPL Ex. (1/2)

```
class Node<region P> {
    region L, R;
    double mass in P;
    Node<P:L> left in P:L;
    Node<P:R> right in P:R;
    void setMassForTree(double mass) {
        this.mass = mass;
        cobegin {

            if (left != null) left.setMassForTree(mass);

            if (right != null) right.setMassForTree(mass);
        }
    }
}
```


UPCRC Illinois
Universal Parallel Computing Research Center

RPL Ex. (2/2)

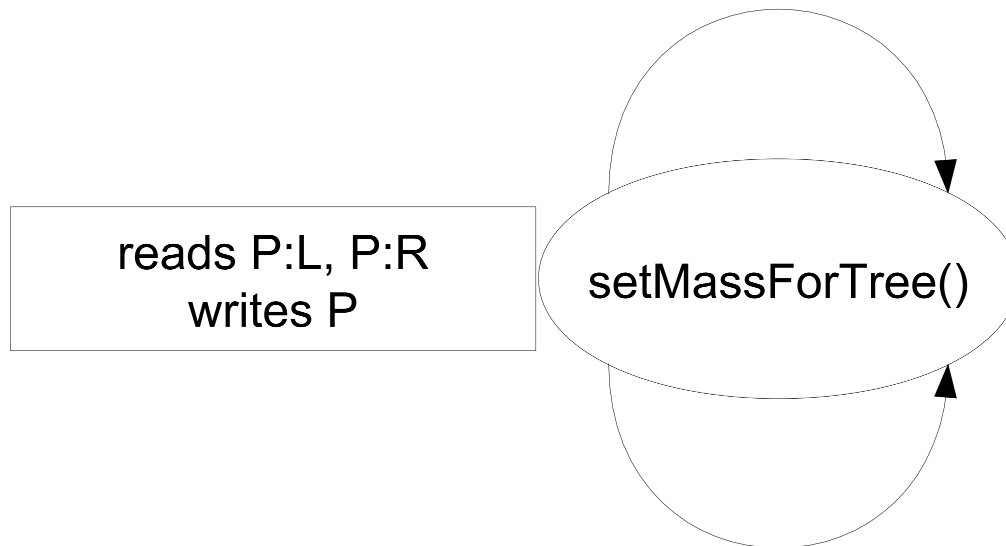
```
class Node<region P> {
    region L, R;
    double mass in P;
    Node<P:L> left in P:L;
    Node<P:R> right in P:R;
    void setMassForTree(double mass) writes P:* {
        this.mass = mass; // Effect: writes P
        cobegin {
            // Effect: writes P:L:*
            if (left != null)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

Constraints in RPL Ex.

```
class Node<region P> {
    region L, R;
    double mass in P;
    Node<P:L> left in P:L;
    Node<P:R> right in P:R;
    void setMassForTree(double mass) {
        this.mass = mass; // Constraint: writes P
        cobegin {
    //Constraint:reads P:L,invokes setMassForTree where {P→ P:L}
            if (left != null) left.setMassForTree(mass);
    //Constraint:reads P:R,invokes setMassForTree where {P→ P:R}
            if (right != null) right.setMassForTree(mass);
        }
    }
}
```

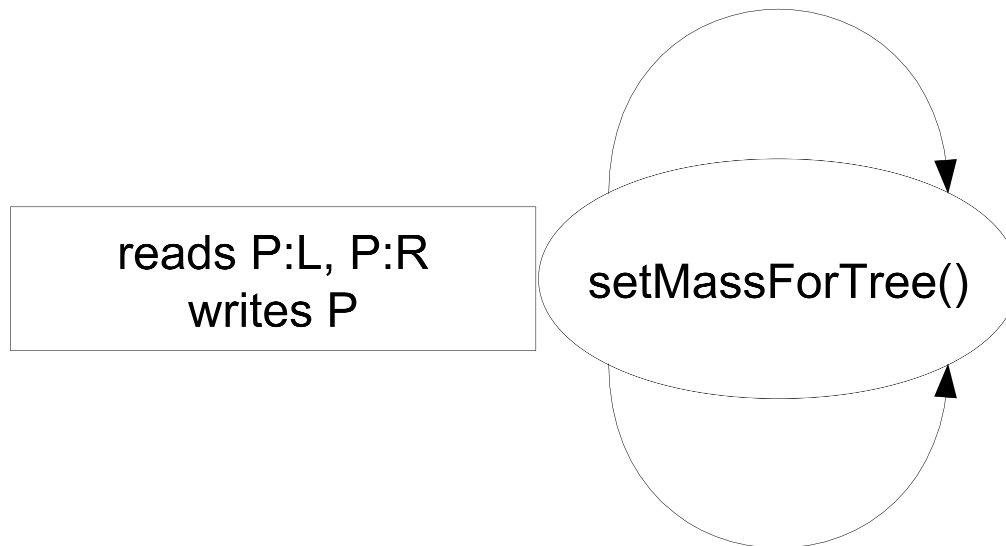

UPCRC Illinois
Universal Parallel Computing Research Center

Constraint Generation Output



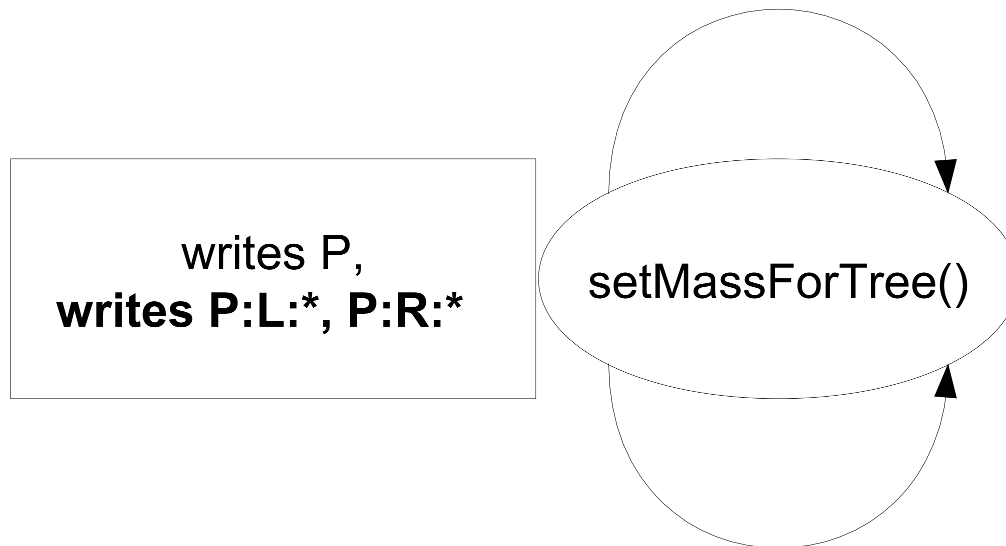
UPCRC Illinois
Universal Parallel Computing Research Center

Constraint Solving



UPCRC Illinois
Universal Parallel Computing Research Center

Constraint Solving



UPCRC Illinois
Universal Parallel Computing Research Center

Evaluation

- Research Questions
  - Is DPJizer useful?
    - Does it alleviate the burden of writing effect annotations?
  - Are the inferred effects precise?
    - Do the inferred effects enable the compiler to prove determinism of the program?

UPCRC Illinois
Universal Parallel Computing Research Center

Evaluation Methodology

- Quantitative
  - We compared the inferred effects to manual effects.
- Qualitative
  - We conducted a survey on DPJ programmers.

UPCRC Illinois
Universal Parallel Computing Research Center

Problems of Manual Effects

- It is **time consuming** to write the effects manually.
- Programmers might write **redundant** effects
  - e.g. reads P, writes P
- Programmer might write **coarse-grained** effects
  - e.g. writes P:* instead of writes P

UPCRC Illinois
Universal Parallel Computing Research Center

Comparing Manual to Inferred Effects

- We used DPJizer to infer method effect summaries of 11 programs.
  - Barnes-Hut, CollisionTree, IDEA, K-means, ListRanking, MergeSort, MonteCarlo, QuadTree, QuickSort, StringMatching, and SumReduce

Total SLOC	Total # of Manually Written Effects	Total # of Effects Too Coarse By Keyword	Total # of Effects Too Coarse By Region	Total # of Redundant Effects
5046	406	17	43	24

UPCRC Illinois
Universal Parallel Computing Research Center

Summary

- DPJizer is an interactive tool for inferring effect annotations given the region annotations.
- DPJizer infers effects that are sufficiently precise and fine-grained.
- We are working on analyses to infer most of the region annotations as well.

UPCRC Illinois
Universal Parallel Computing Research Center

Acknowledgement

This work is funded by Microsoft and Intel through the Universal Parallel Computing Research Center (UPCRC) at the University of Illinois and by NSF grants 07-02724, 07-20772, 08-33128 and 08-33188.

References

- http://dpj.cs.illinois.edu
- Mohsen Vakilian et al., Inferring Method Effect Summaries for Nested Heap Regions, to appear in ASE 2009
- Robert Bocchino et al., A Type and Effect System for Deterministic Parallel Java, OOPSLA 2009.