Software metrics for parallelism

Motivating example

(*Two alternative implementations of vector multiplication – different data structures, representing vectors in memory, lead to different code parallelisability)*

Good Score

```
int a[1000];
int b[1000];
int c;

// initialize arrays

c = 0;
for (int i = 0; i < 1000; i++) {
    c += a[i] * b[i];
}
```

Bad Score

```
Node* a;
Node* b;
int c;

// initialize linked lists

c = 0;
while (a && b) {
    c += a->val * b->val;
    a = a->next;
    b = b->next;
}
```

While array implementation is parallelisible, the while loop on the right has cross iteration dependency - we cannot proceed to the next iteration until we compute address of the next element in the list. This example clearly demonstrates the problem. Suboptimal data structure choice made by the programmer hid all the inherent parallelism of the task and made it further impossible for compiler to automatically parallelise the program. If only a programmer had some parallelisation feedback tools, which could mark the code on the left as “good” and the code on the right as “bad”, then he would have paid more attention to this piece of code and gave it a second thought.

EPSRC
Engineering and Physical Sciences Research Council

THE UNIVERSITY of EDINBURGH
informatics

EPSRC Centre for Doctoral Training in
Pervasive Parallelism