## Angelic Entanglement

### Motivation
Programmers learn and design algorithms by studying examples, before writing pseudocode. Can an oracle generate these examples for us?

We developed a language which uses an oracle to provide demonstrations of an algorithm’s execution before the algorithm is developed. The oracle provides values that the programmer does not know how to compute. The programmer then generalizes these executions into an algorithm.

The oracle is an angelically nondeterministic choice operator. Correctness of a demonstration is defined by assertions in the program.

```
sum = choose(1,2) + choose(1,2)
assert sum == 3
// traces generated: (1,2) (2,1)
```

### Case Study
The Deutsch-Schorr-Waite algorithm traverses a graph using constant storage. We decomposed the original algorithm to use a parasitic stack, which stores backtracking pointers using the pointers in the graph.

ParasiticStack
- `e = null`
- `n = choose(nodes)`
- `c = choose(0 until n.children.length)`
- `v = n.children[c]`
- `e, n.children[c] = choose(x, n, v, e)`

Using the ENT query, we see that oracles across function boundaries are coordinating. We use the SUP query to find subsets which remove this coordination.

Using the SUP query, we removed this coordination. We received 5 maximal subsets which did not display the coordination. When examining the 3rd subset we found it corresponded to the desired algorithm.

### Future Work
We are currently using this analysis to develop novel parallel scan algorithms. We implemented a scan as an angelic adder network, with each row representing one cycle. The oracles determine which values should be added.

Constraints can be added to the network, such as depth, number of adds, fanout and number of adds per row.

Independent subnetworks in the scan network can be shifted in time without affecting the reset of the computation. This property causes the chooses in each independent network to show up as entangled.

We want to be able to find recursive decompositions of scans using entanglement.

---

**Angelics Programming**

**Entanglement Query (ENT)**
- Input: Complete set of traces from an angelic program
- Output: Partitioning of all choose statements

The algorithm divides the chooses such that ones in different partitions do not coordinate with each other.

**Maximal Support Query (SUP)**
- Input: Complete set of traces T and a partitioning of choose statements
- Output: A subset of traces, S, such that ENT(S) ≤ P

The algorithm returns a maximal subset of traces in which coordination is removed. The subset is maximal because adding any subset of T to S would introduce coordination between the partitions of P.