http://parasol.tamu.edu/mach7/

Parasol
Smarter computing.
Texas A&M University

Open Pattern Matching for C++

Yuriy Solodkyy · Gabriel Dos Reis · Bjarne Stroustrup

λ-calculus in C++

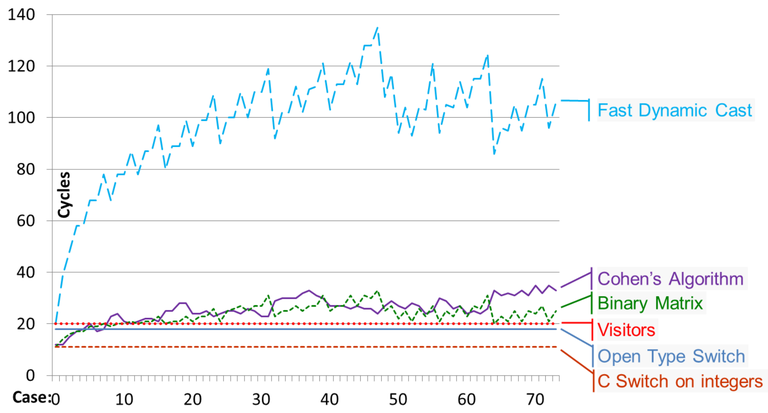
```
struct Term       { virtual ~Term() {} };
struct Var : Term { std::string name;  };
struct Abs : Term { Var&  var;  Term& body; };
struct App : Term { Term& func; Term& arg;  };

Term* eval(Term* t) {
  var<const Var&> v; var<const Term&> a,b;
  Match(t) {
    Case(C<Var>())                return &match0;
    Case(C<Abs>())                return &match0;
    Case(C<App>(C<Abs>(&v,&b),&a)) return eval(subs(b,v,a));
    Otherwise() std::cerr << "Invalid term"; return nullptr;
  } EndMatch
}

bool operator==(const Term& left, const Term& right) {
  var<std::string> s; var<const Term&> v,t,f;
  Match( left        , right        ) {
    Case(C<Var>(s)   , C<Var>(+s)   ) return true;
    Case(C<Abs>(v,t) , C<Abs>(+v,+t)) return true;
    Case(C<App>(f,t) , C<App>(+f,+t)) return true;
    Otherwise()                       return false;
  } EndMatch
}
```

Generalized n+k Patterns

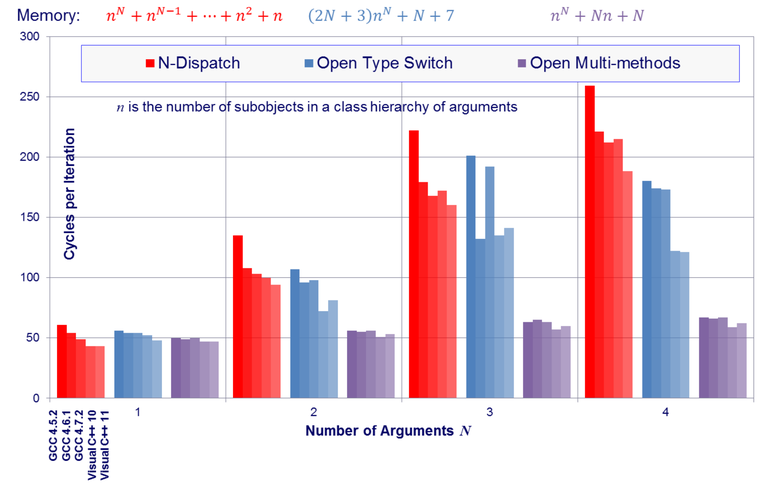
```
double power(double x, int n) {
    var<int> m;
    Match(n) {
      Case(0)      return 1.0;
      Case(1)      return x;
      Case(2*m)    return   sqr(power(x,m));
      Case(2*m+1) return x*sqr(power(x,m));
    } EndMatch
}
```

$$x^n = \begin{cases} 1 & n = 0 \\ x & n = 1 \\ (x^m)^2 & n = 2m \\ x(x^m)^2 & n = 2m+1 \end{cases}$$

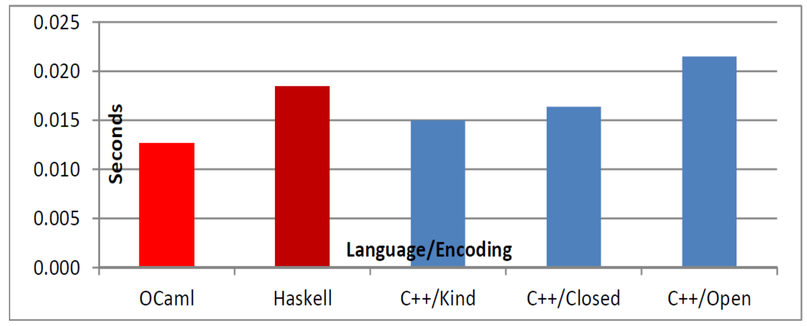
Comparison to Alternatives



Comparison to Multiple Dispatch

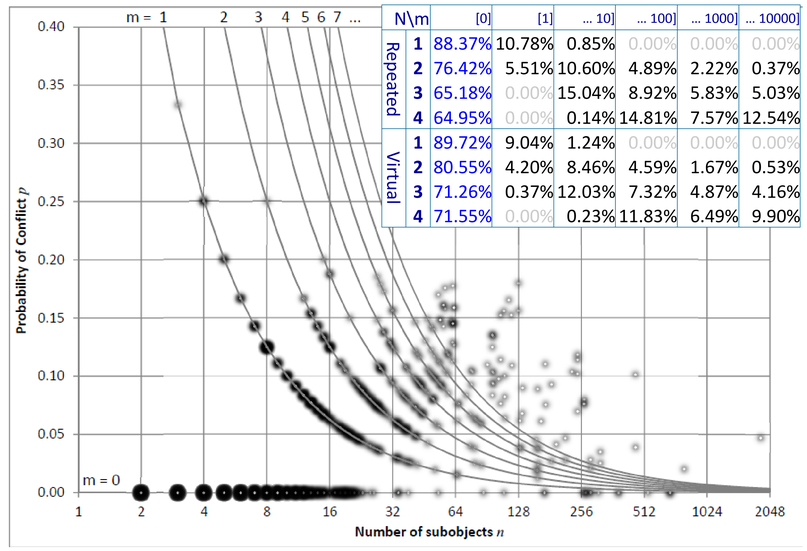


Comparison with OCaml & Haskell



Efficiency of Hashing

88.37% percentage of real-world type switches with no collisions in the hash
N—number of arguments, m—number of collisions due to hashing



N\m		[0]	[1]	... 10]	... 100]	... 1000]	... 10000]
Repeated	1	88.37%	10.78%	0.85%	0.00%	0.00%	0.00%
	2	76.42%	5.51%	10.60%	4.89%	2.22%	0.37%
	3	65.18%	0.00%	15.04%	8.92%	5.83%	5.03%
	4	64.95%	0.00%	0.14%	14.81%	7.57%	12.54%
Virtual	1	89.72%	9.04%	1.24%	0.00%	0.00%	0.00%
	2	80.55%	4.20%	8.46%	4.59%	1.67%	0.53%
	3	71.26%	0.37%	12.03%	7.32%	4.87%	4.16%
	4	71.55%	0.00%	0.23%	11.83%	6.49%	9.90%

Performance Evaluation

42% pattern matching is faster than visitors by
42% visitors are faster than pattern matching

		Open				Closed			
		G++		Visual C++		G++		Visual C++	
	x86-32	Linux	Windows	w/ PGO	w/o PGO	Linux	Windows	w/ PGO	w/o PGO
	REP	16%	14%	1%	2%	124%	122%	100%	76%
	SEQ	56%	12%	48%	2%	640%	467%	29%	30%
	RND	56%	0%	9%	5%	603%	470%	35%	32%
Forwarding	REP	33%	22%	8%	24%	53%	49%	24%	20%
Forwarding	SEQ	55%	233%	135%	193%	86%	290%	48%	12%
Forwarding	RND	78%	25%	3%	13%	88%	33%	8%	18%

Y.Solodkyy, G.Dos Reis, B.Stroustrup. "Open and Efficient Type Switch for C++" Proceedings OOPSLA'12, October 19-26, 2012, Tucson, AZ, USA
Y.Solodkyy, G.Dos Reis, B.Stroustrup. "Open Pattern Matching for C++" Proceedings GPCE'13, October 27-28, 2013, Indianapolis, IN, USA
Y.Solodkyy "Simplifying the Analysis of C++ Programs" Ph.D. Thesis, August 2013, Texas A&M University, USA

Pattern Matching Overhead

42% faster than handcrafted version
42% slower than handcrafted version

	Open Patterns					Patterns as Objects				
	G++			Visual C++		G++			Visual C++	
	4.5.2	4.6.1	4.7.2	10	11	4.5.2	4.6.1	4.7.2	10	11
$factorial_0$	15%	13%	17%	85%	35%	347%	408%	419%	2121%	1788%
$factorial_1$	0%	6%	0%	83%	21%	410%	519%	504%	2380%	1812%
fibonacci	17%	2%	2%	62%	15%	340%	431%	395%	2730%	2597%
gcd_1	21%	25%	25%	309%	179%	1503%	1333%	1208%	8876%	7810%
gcd_2	1%	0%	1%	38%	15%	119%	102%	108%	1575%	1319%
lambda	58%	54%	56%	29%	34%	837%	780%	875%	259%	289%
power	10%	8%	13%	50%	6%	291%	337%	338%	1950%	1648%

Partially supported by NSF grants CCF-0702765, CCF-1043084, and CCF-1150055