

./main.cpp

```

1: // declaration
2: // (D,F,K,sigma,tau)->Black(D,F,K,sigma,tau)
3:
4: template <typename D, typename F, typename K, typename V, typename T>
5: struct BlackTraits;
6:
7: template <typename D, typename F, typename K, typename V, typename T> inline
8: typename BlackTraits<D, F, K, V, T>::type black(const FuncExpr<D>& d,
9:         const FuncExpr<F>& f, const FuncExpr<K>& k, const FuncExpr<V>& v,
10:        const FuncExpr<T>& t)
11: {
12:     return BlackTraits<D, F, K, V, T>::apply(d(), f(), k(), v(), t());
13: }
14:
15: // (D,F,K,sigma,tau) -> D((Ff,(d_1)-Kf,(d_2))
16: template <typename D, typename F, typename K, typename V, typename T>
17: struct BlackTraits {
18:     typedef ForwardBlackTraits<F, K, V, T> FwdBl;
19:     typedef FuncBinatyTraits<D, typename FwdBl::type, scalar_mult> Disc;
20:     typedef typename Disc::type type;
21:
22:     static type apply(const D& d, const F& f, const K& k, const V& v, const T& t)
23:     {
24:         return Disc::apply(d, FwdBl::apply(f,k,v,t));
25:     }
26:
27: };
28:
29:
30: // (F, K, sigma, tau)->Ff,(d_1)-Kf,(d_2)
31: template <typename D, typename F, typename K, typename V, typename T>
32: struct ForwardBlackTraits {
33:     typedef AoNTraits<F, K, V, T> AoN;
34:     typedef CoNTraits<F, K, V, T> CoN;
35:     typedef FuncBinaryTraits<typename AoN::type, typename CoN::type,
36:         scalar_minus> Minus;
37:     typedef typename Minus::type type;
38:
39:     static type apply(const D& d, const F& f, const K& k, const V& v, const T& t)
40:     {
41:         return Minus::apply(AoN::apply(f, k, v, t), CoN::apply(f, k, v, t));
42:     }
43: };
44:
45: // (F, K, sigma, tau)->Ff,(d_1)
46: template <typename D, typename F, typename K, typename V, typename T>
47: struct AoNTraits {
48:     typedef DlTraits<F, K, V, T> Dl;
49:     typedef FuncUnaryTraits<typename Dl::type, scalar_normdist> Nl;
50:     typedef FuncBinaryTraits<F, typename Nl::type, scalar_mult> Mult;
51:     typedef typename Mult::type type;
52:
53:     static type apply(const F& f, const K& k, const V& v, const T& t)
54:     {
55:         return Mult::apply(f, Nl::apply(Dl::apply(f, k, v, t)));
56:     }
57: };
58:
59: // (F, K, sigma, tau)->Kf,(d_2)
60: template <typename F, typename K, typename V, typename T>
61: struct CoNTraits {
62:     typedef D2Traits<F, K, V, T> D2;
63:     typedef FuncUnaryTraits<typename D2::type, scalar_normdist> N2; // 26 \205æ\224

```

```

c@\227traits
64:     typedef FuncBinaryTraits<F, typename N2::type, scalar_mult> Mult; // 36 \205æ\
\224c@\227traits
65:     typedef typename Mult::type type;
66:
67:     static type apply(const F& f, const K& k, const V& v, const T& t)
68:     {
69:         return Mult::apply(f, N2::apply(D2::apply(f, k, v, t)));
70:     }
71:
72: };
73:
74: // (F, K, sigma, tau)->d_2 + sigma * sqrt(tau)
75: template <typename F, typename K, typename V, typename T>
76: struct DlTraits {
77:     typedef D2Traits<F, K, V, T> D2;
78:     typedef StdevTraits<V, T> Sd;
79:     typedef FuncBinaryTraits<typename D2::type, typename Sd::type, scalar_plus> Plus
;
80:     typedef typename Plus::type type;
81:
82:     static type apply(const F& f, const K& k, const V& v, const T& t)
83:     {
84:         return Plus::apply(D2::apply(f, k, v, t), Sd::apply(v, t));
85:     }
86: };
87:
88: // (F, K, sigma, tau)->log(F/K)/(sigma * sqrt(tau) + 0.5 * sigma * sqrt(tau)
89: template <typename F, typename K, typename V, typename T>
90: struct D2Traits {
91:     typedef LogMoneynessTraits<F, K> LnFK;
92:     typedef StdefTraits<V, T> Sd;
93:     typedef Rational<double, 1, 2> Half;
94:     typedef FuncBinaryTraits<typename LnFK::type, typename Sd::type, scalar_div> X1;
95:     typedef FuncBinaryTraits<typename X1::type, typename X2::type, scalar_mult> X2;
96:     typedef FuncBinaryTraits<typename X1::type, typename X2::type, scalar_plus> Plus
;
97:     typedef typename Plus::type type;
98:
99:     static type apply(const F& f, const K& k, const V& v, const T& t)
100:     {
101:         return Plus::apply(X1::apply(LnFK::aply(f, k), Sd::apply(v, t)),
102:             X2::apply(Half(), Sd::apply(v, t)));
103:     }
104: };
105:
106: // (F,K)->log(F/K)
107: template <typename F, typename K>
108: struct LogMoneynessTraits {
109:     typedef FuncBinaryTraits<F, K, scalar_div> FK;
110:     typedef FuncUnaryTraits<typename FK::type scalar_log> Ln;
111:     typedef typename Ln::type type;
112:
113:     static type apply(const F& f, const K& k)
114:     {
115:         return Ln::apply(FK::apply(f, k));
116:     }
117: };
118:
119: // (sigma, tau) -> sigma * sqrt(tau)
120: template <typename V, typename T>
121: struct StdevTraits {
122:     typedef FuncUnaryTraits<T, scalar_sqrt> Sqrt;

```

```
123:     typedef FuncBinaryTraits<V, typename Sqrt::type, scalar_mult> Mult;
124:     typedef typename Mult::type type;
125:
126:     static type apply(const V& v, const T& t)
127:     {
128:         return Mult::apply(v, Sqrt::apply(t));
129:     }
130: };
131:
132: int main()
133: {
134:
135:     return 0;
136: }
```