Multisorted Algebras

Friend or Foe?

Complex(R : Ring) : Algebra(R)

F(T : S(A, B), A : E(B), B : Ring)

ρ : List(T)

Type expression size is $O(h^2)$
where h is height of type expr.

Want $O(h)$

Why don't Java + C++ worlds
see this as a pb?

A : C++ : Type parameters not qualified

A : Java – not much experience –
– "object" lock down. // no fundamental effect.

A : Algebra(R), R : Commutative Ring, Z : Integer

$* : (R, A) \to A$

$\wedge : (A, Z) \to A$

Alg : Σ

f(Alg : Σ)

a : Alg.A

My P = (. . . . .)

X(P : DMPCat(A ~~B, C, D, E~~))

A : S₁(B, C)

~~D~~ : S₂(E, A)

~~B~~ :

X(MyP, Z)

p : P

a : A

- Parametric polymorphism (SML, Haskell..)

length :: [a] → Int

length [1, 2, 3, 3]

length :: forall a. [a] → Int

Hindley-Milner

- C++0x — Concepts

Haskell type class

```
class Ord a where
  (<) :: a -> a -> Bool
  (>) :: a -> a -> Bool
```

Ord a => a -> a -> Bool

1 < 2

```
template <class InputIterator, class OutputIterator>
OutputIterator
copy(InputIter first, InputIterator last, OutputIter out)
{
    for( ; first != last; ++first)
        *out++ = *first;
} return out;
```

where

```
vector<int> v;
list<int> l;
copy(l.begin(), l.end(), v.begin);
```

```
concept InputIterator<typename T> {
    typename value_type;
    bool operator==(T,T);
    bool operator!=(T,T);
    value_type operator*(T);
};

concept_map InputIterator<int*> {
    ...

}
```

<InputIterator iter, OutputIterator outer>

```
template<class InIter, class OutIter>
    where InputIterator<Inter> && OutputIterator<OutIter>

copy(InIter first, Inter last, OutIter out)
{
  ___
  ___
}

    copy(l.begin(), l.end(), v.begin()).
```

SMW

$$F(X : C_1(A, B), A : C_2(Z))$$

$$\exists B : C_3(A)$$

$$\exists ? : C_5(A, B, X)$$