Multisort Species in Aldor

Ralf Hemmecke

Research Institute for Symbolic Computation
Johannes Kepler University Linz, Austria

25-Jul-2008

Aldor-Combinat

- Project initiated at Axiom Workshop 2006
- Authors: Hemmecke, Rubey
- Supporters: Aistleitner, Thiery
- `http://www.risc.uni-linz.ac.at/people/hemmecke/aldor/combinat/`
- Reference: F. Bergeron and G. Labelle and P. Leroux *"Combinatorial Species and Tree-Like Structures"*

Combinatorial Species

Definition

A species of structures is a rule F which

1. produces, for each finite set U, a finite set $F[U]$,
2. produces, for each bijection $\sigma : U \to V$, a function

$$F[\sigma] : F[U] \to F[V]. \tag{1}$$

The functions $F[\sigma]$ should further satisfy:

1. for all bijections $\sigma : U \to V$ and $\tau : V \to W$,

$$F[\tau \circ \sigma] = F[\tau] \circ F[\sigma], \tag{2}$$

2. for the identity map $\mathrm{Id}_U : U \to U$,

$$F[\mathrm{Id}_U] = \mathrm{Id}_{F[U]}. \tag{3}$$

Combinatorial Species

- A combinatorial structure is a covariant functor $F : \mathbb{B} \to \mathbb{B}$ from the category $\mathbb{B}$ of finite sets and bijections into itself.
- $s_1 \in F[U]$, $s_2 \in F[V]$, $\sigma : U \to V$ (bijection) s_1 is isomorphic to s_2 if $s_2 = \sigma \cdot s_1 = F[\sigma](s_1)$.
  s_1 and s_2 have the same isomorphism type.
- Action of the symmetric group

$$\begin{array}{ll} \sigma : U \to U & F[\sigma] : F[U] \to F[U] \\ S_U \times U \to U & S_U \times F[U] \to F[U] \\ \sigma \cdot u & \sigma \cdot s \end{array}$$

Combinatorial Species in Aldor

```
LabelType: Category == with {
        PrimitiveType;
        OutputType;
}
CombinatorialSpecies(L: LabelType): Category == with {
        <<exports: CombinatorialSpecies>>
}

Cycle(L: LabelType): CombinatorialSpecies L with {
        <<exports: Cycle>>
} == List L add {
        Rep == List L;
        import from I, Rep;
        <<implementation: Cycle>>
}
```

Product of Species

Definition

$$(F \cdot G)[U] = \sum_{\substack{U_1 \cap U_2 = \emptyset \\ U_1 \cup U_2 = U}} F[U_1] \times G[U_2]$$

```
macro SPECIES == (L: LabelType) -> CombinatorialSpecies L;
Times(
    F: SPECIES,
    G: SPECIES
)(L: LabelType): CombinatorialSpecies(L) == add {
        Rep == Record(tag: Subset L, left: F L, right: G L);
        import from Rep;
        <<implementation: Times>>
}
```

Composition of Species

Definition

$$(F \circ G)[U] = \sum_{\pi \text{ partition of } U} F[\pi] \times \prod_{p \in \pi} G[p]$$

for any finite set U where the (disjoint) sum is taken over the set of partitions π of U (i.e., $\pi \in \mathrm{Par}[U]$).

```
Compose(
    F: SPECIES,
    G: SPECIES
)(L: LabelType): CombinatorialSpecies(L) == add {
        Rep == Record(
            tag: Partition L,
            left: F SetSpecies L,
            right: SetSpecies G L
        );
        <<implementation: Compose>>
}
```

Multisort Species

- $F : \mathbb{B} \to \mathbb{B}$ (species)
- $F : \mathbb{B}^k \to \mathbb{B}$ (multisort species)

Multisets

Definition

A an object $U = (U_1, ..., U_k)$ of $\mathbb{B}^k$ is called a *k*-multiset.

Generic Multisort Species in Aldor

Similar to the species definition

```
F(L: LabelType): CombinatorialSpecies L == add { ... }
```

we want

```
G(M: MultiLabelType): MultiCombinatorialSpecies M == ...
```

How to define `MultiLabelType`?
We would like to say generically:

```
G(String);
G(Integer);
G(Integer, String);
G(String, String, Integer);
```

No problem if we define k-variate species for each k, but this is duplicating code.

How to define MultiLabelType?

We should have

```
(Integer, String): MultiLabelType;
(String, String, Integer): MultiLabelType;
```

It seems like the definition should be like this:

```
MultiLabelType == Tuple(LabelType)
```

However,

1. `(Integer, String)` is not a domain,
2. `MultiLabelType` is a domain but should be a category.

First point might be resolve by adding `Cross`.

Need another Tuple

```
define FiniteProductType(C: Category): Category == with {
        #: I;
        type: I -> C;
}
FiniteProduct(C: Category, F: Tuple C): FiniteProductType C
== add {
        #: I == length F;
        type(i: I): C == element(F, i);
}
```

Partitional Compostion

Definition

Let $F = F(Y_1, \ldots, Y_m)$ be a multisort species and $G_1, \ldots, G_m$ k-sort species. The partitional composition $F(G_1, \ldots, G_m)$ is a k-sort species defined by setting for $U = (U_1, \ldots, U_k)$

$$F(G_1, \ldots, G_m)[U] = \sum_{\substack{\pi \in \mathrm{Par}[U] \\ \chi : \pi \to [m]}} F[\chi^{-1}] \times \prod_{\substack{j \in [m] \\ C \in \chi^{-1}(j)}} G_j[C], \qquad (4)$$

where, for each function $\chi : \pi \to [m]$, χ^{-1} denotes the m-set $(\chi^{-1}(1), \ldots, \chi^{-1}(m))$ associated to χ.