Row Polymorphism

John Skaller

August 1, 2016

1 Records

Felix has structurally typed records as illustrated below:

```
var r : (a:int,b:string) = (a=1,b="Hello");
```

The order of the fields fields with different names is irrelevant.

The names of the fields can be used as projections:

```
var i = a r;
var s = r.b;
```

Stand alone value projections can be specified like:

```
var p = a of (a:int,b:string);
var i = p r;
```

Pointer projections are also supported:

```
var pi : &int = &r.a;
var ps : &string = r&.b;
var i = *pi;
var s = *ps;
var p = a of &(a:int,b:string);
var i2 = *(p &r);
```

Record field names may be repeated:

```
var rr : (a:int,a:double,a:int,b:string)
  = (a=1,a=2.3,a=4L,b="Hello")
;
```

In this case projections refer to the leftmost field of the given name. The order of fields with the same name matters.

One can also use blank field names. The grammar allows the name `n""` to be used. Alternatively, the name can be omitted, or both the name and `=` sign can be omitted, except for the first field:

```
var x = (a=1,=2,n""=3);
println$ x._strr;
```

If all the field names are blank, you can also omit the = sign from the first entry, and, if precedence allows, the parentheses:

```
var x : int * int * string = 1,2,"hello";
println$ x._strr;
```

in which case the record called a tuple. Plain decimal integer literals can be used for tuple projections:

```
var x : int * int * string = 1,2,"hello";
println$ x.1;
```

Standalone tuples projections are denoted like:

```
var x : int * int * string = 1,2,"hello";
var prj1 = proj 1 of (int * int * string);
println$ prj1 x;
```

If the types of a tuple are all the same, it is called an array:

```
var x : int ^ 3 = 1,2,3;
println$ x._strr;
```

In this case the projections can be either an integer expression, or an expression of the type of the array index:

```
var x : int ^ 3 = 1,2,3;
var one = 0 x;
var two = x.1;
var three = x.(case 2 of 3);
```

If the projection is an integer it is bounds checked at run time. If it is a compact linear type which is the type of the array index, no bounds check is required.

2 Addition

Records and tuples can be concatenated with infix operator +. Duplicate fields retain their order.

3 Polyrecords

A list of fields may be added to any type with a polyrecord expression:

```
var x = (a=1,b=2);
var y = (a=3,d=2 | x);
```

The type of the result is a record if the RHS term is a record, including tuples or unit tuple.