

## 13 Fold\_ZF.thy

```
theory Fold_ZF imports InductiveSeq_ZF
```

```
begin
```

Suppose we have a binary operation  $P : X \times X \rightarrow X$  written multiplicatively as  $P\langle x, y \rangle = x \cdot y$ . In informal mathematics we can take a sequence  $\{x_k\}_{k \in 0..n}$  of elements of  $X$  and consider the product  $x_0 \cdot x_1 \cdot \dots \cdot x_n$ . To do the same thing in formalized mathematics we have to define precisely what is meant by that "...". The definition we want to use is based on the notion of sequence defined by induction discussed in `InductiveSeq_ZF`. We don't really want to derive the terminology for this from the word "product" as that would tie it conceptually to the multiplicative notation. This would be awkward when we want to reuse the same notions to talk about sums like  $x_0 + x_1 + \dots + x_n$ . In functional programming there is something called "fold". Namely for a function  $f$ , initial point  $a$  and list  $[b, c, d]$  the expression `fold(f, a, [b, c, d])` is defined to be  $f(f(f(a, b), c), d)$  (in Haskell something like this is called `foldl`). If we write  $f$  in multiplicative notation we get  $a \cdot b \cdot c \cdot d$ , so this is exactly what we need. The notion of folds in functional programming is actually much more general than what we need here (not that I know anything about that). In this theory file we just make a slight generalization and talk about folding a list with a binary operation  $f : X \times Y \rightarrow X$  with  $X$  not necessarily the same as  $Y$ .

### 13.1 Folding in ZF

Suppose we have a binary operation  $f : X \times Y \rightarrow X$ . Then every  $y \in Y$  defines a transformation of  $X$  defined by  $T_y(x) = f\langle x, y \rangle$ . In `IsarMathLib` such transformation is called as `Fix2ndVar(f, y)`. Using this notion, given a function  $f : X \times Y \rightarrow X$  and a sequence  $y = \{y_k\}_{k \in N}$  of elements of  $Y$  we can get a sequence of transformations of  $X$ . This is defined in `Seq2TransSeq` below. Then we use that sequence of transformations to define the sequence of partial folds (called `FoldSeq`) by means of `InductiveSeqVarFN` (defined in `InductiveSeq_ZF` theory) which implements the inductive sequence determined by a starting point and a sequence of transformations. Finally, we define the fold of a sequence as the last element of the sequence of the partial folds.

Definition that specifies how to convert a sequence  $a$  of elements of  $Y$  into a sequence of transformations of  $X$ , given a binary operation  $f : X \times Y \rightarrow X$ .

**definition**

$$\text{Seq2TrSeq}(f, a) \equiv \{\langle k, \text{Fix2ndVar}(f, a(k)) \rangle. k \in \text{domain}(a)\}$$

Definition of a sequence of partial folds.

**definition**

```
FoldSeq(f,x,a) ≡  
InductiveSeqVarFN(x,fstodom(f),Seq2TrSeq(f,a),domain(a))
```

Definition of a fold.

**definition**

```
Fold(f,x,a) ≡ Last(FoldSeq(f,x,a))
```

If  $X$  is a set with a binary operation  $f : X \times Y \rightarrow X$  then  $\text{Seq2TransSeqN}(f, a)$  converts a sequence  $a$  of elements of  $Y$  into the sequence of corresponding transformations of  $X$ .

**lemma seq2trans\_seq\_props:**

```
assumes A1: n ∈ nat and A2: f : X×Y → X and A3: a:n→Y and  
A4: T = Seq2TrSeq(f,a)
```

**shows**

```
T : n → (X→X) and  
∀k∈n. ∀x∈X. (T(k))(x) = f(x,a(k))
```

**proof -**

```
from 'a:n→Y' have D: domain(a) = n using func1_1_L1 by simp  
with A2 A3 A4 show T : n → (X→X)  
  using apply_funtype fix_2nd_var_fun ZF_fun_from_total Seq2TrSeq_def  
  by simp  
with A4 D have I: ∀k ∈ n. T(k) = Fix2ndVar(f,a(k))  
  using Seq2TrSeq_def ZF_fun_from_tot_val0 by simp  
{ fix k fix x assume A5: k∈n x∈X  
  with A1 A3 have a(k) ∈ Y using apply_funtype  
  by auto  
  with A2 A5 I have (T(k))(x) = f(x,a(k))  
  using fix_var_val by simp  
} thus ∀k∈n. ∀x∈X. (T(k))(x) = f(x,a(k))  
  by simp
```

**qed**

Basic properties of the sequence of partial folds of a sequence  $a = \{y_k\}_{k \in \{0, \dots, n\}}$ .

**theorem fold\_seq\_props:**

```
assumes A1: n ∈ nat and A2: f : X×Y → X and  
A3: y:n→Y and A4: x∈X and A5: Y≠0 and  
A6: F = FoldSeq(f,x,y)
```

**shows**

```
F: succ(n) → X  
F(0) = x and  
∀k∈n. F(succ(k)) = f(F(k), y(k))
```

**proof -**

```
let T = Seq2TrSeq(f,y)  
from A1 A3 have D: domain(y) = n  
  using func1_1_L1 by simp  
from 'f : X×Y → X' 'Y≠0' have I: fstodom(f) = X  
  using fstodomdef by simp
```

```

with A1 A2 A3 A4 A6 D show
  II: F: succ(n) → X and F(0) = x
  using seq2trans_seq_props FoldSeq_def fin_indseq_var_f_props
  by auto
from A1 A2 A3 A4 A6 I D have ∀k∈n. F(succ(k)) = T(k)(F(k))
  using seq2trans_seq_props FoldSeq_def fin_indseq_var_f_props
  by simp
moreover
{ fix k assume A5: k∈n hence k ∈ succ(n) by auto
  with A1 A2 A3 II A5 have (T(k))(F(k)) = f⟨F(k),y(k)⟩
    using apply_funtype seq2trans_seq_props by simp }
ultimately show ∀k∈n. F(succ(k)) = f⟨F(k), y(k)⟩
  by simp
qed

```

A consistency condition: if we make the list shorter, then we get a shorter sequence of partial folds with the same values as in the original sequence. This can be proven as a special case of `fin_indseq_var_f_restrict` but a proof using `fold_seq_props` and induction turns out to be shorter.

```

lemma foldseq_restrict: assumes
  n ∈ nat    k ∈ succ(n) and
  i ∈ nat    f : X×Y → X    a : n → Y    b : i → Y and
  n ⊆ i      ∀j ∈ n. b(j) = a(j)    x ∈ X    Y ≠ 0
shows FoldSeq(f,x,b)(k) = FoldSeq(f,x,a)(k)
proof -
  let P = FoldSeq(f,x,a)
  let Q = FoldSeq(f,x,b)
  from assms have
    n ∈ nat    k ∈ succ(n)
    Q(0) = P(0) and
    ∀j ∈ n. Q(j) = P(j) → Q(succ(j)) = P(succ(j))
    using fold_seq_props by auto
  then show Q(k) = P(k) by (rule fin_nat_ind)
qed

```

A special case of `foldseq_restrict` when the longer sequence is created from the shorter one by appending one element.

```

corollary fold_seq_append:
  assumes n ∈ nat    f : X×Y → X    a:n → Y and
  x∈X    k ∈ succ(n)    y∈Y
  shows FoldSeq(f,x,Append(a,y))(k) = FoldSeq(f,x,a)(k)
proof -
  let b = Append(a,y)
  from assms have b : succ(n) → Y    ∀j ∈ n. b(j) = a(j)
    using append_props by auto
  with assms show thesis using foldseq_restrict by blast
qed

```

What we really will be using is the notion of the fold of a sequence, which we

define as the last element of (inductively defined) sequence of partial folds. The next theorem lists some properties of the product of the fold operation.

**theorem** fold\_props:

assumes A1:  $n \in \text{nat}$  and  
 A2:  $f : X \times Y \rightarrow X$   $a : n \rightarrow Y$   $x \in X$   $Y \neq 0$   
 shows  
 $\text{Fold}(f, x, a) = \text{FoldSeq}(f, x, a)(n)$  and  
 $\text{Fold}(f, x, a) \in X$

**proof** -

from assms have  $\text{FoldSeq}(f, x, a) : \text{succ}(n) \rightarrow X$   
 using fold\_seq\_props by simp  
 with A1 show  
 $\text{Fold}(f, x, a) = \text{FoldSeq}(f, x, a)(n)$  and  $\text{Fold}(f, x, a) \in X$   
 using last\_seq\_elem apply\_funtype Fold\_def by auto

qed

The next theorem tells us what happens to the fold of a sequence when we add one more element to it.

**theorem** fold\_append:

assumes A1:  $n \in \text{nat}$  and A2:  $f : X \times Y \rightarrow X$  and  
 A3:  $a : n \rightarrow Y$  and A4:  $x \in X$  and A5:  $y \in Y$   
 shows  
 $\text{FoldSeq}(f, x, \text{Append}(a, y))(n) = \text{Fold}(f, x, a)$  and  
 $\text{Fold}(f, x, \text{Append}(a, y)) = f(\text{Fold}(f, x, a), y)$

**proof** -

let  $b = \text{Append}(a, y)$   
 let  $P = \text{FoldSeq}(f, x, b)$   
 from A5 have I:  $Y \neq 0$  by auto  
 with assms show thesis1:  $P(n) = \text{Fold}(f, x, a)$   
 using fold\_seq\_append fold\_props by simp  
 from assms I have II:  
 $\text{succ}(n) \in \text{nat}$   $f : X \times Y \rightarrow X$   
 $b : \text{succ}(n) \rightarrow Y$   $x \in X$   $Y \neq 0$   
 $P = \text{FoldSeq}(f, x, b)$   
 using append\_props by auto  
 then have  
 $\forall k \in \text{succ}(n). P(\text{succ}(k)) = f(P(k), b(k))$   
 by (rule fold\_seq\_props)  
 with A3 A5 thesis1 have  $P(\text{succ}(n)) = f(\text{Fold}(f, x, a), y)$   
 using append\_props by auto  
 moreover  
 from II have  $P : \text{succ}(\text{succ}(n)) \rightarrow X$   
 by (rule fold\_seq\_props)  
 then have  $\text{Fold}(f, x, b) = P(\text{succ}(n))$   
 using last\_seq\_elem Fold\_def by simp  
 ultimately show  $\text{Fold}(f, x, \text{Append}(a, y)) = f(\text{Fold}(f, x, a), y)$   
 by simp

qed

end