1 Well–Founded Relations

Let $\prec$ be a binary relation on a set A. The type $Fin(A, \prec)$ is the set of elements $a \in A$ such that there is no infinite descending sequence $\{a_n\}_{n \in N}$ verifying

$$\ldots a_{n+1} \prec a_n \prec \ldots a_2 \prec a_1 \prec a. \tag{1}$$

The relation $\prec \subseteq A \times A$ is called noetherian if $A = Fin(A, \prec)$.

Furthermore, given $A : Set$ and $\prec \subseteq A \times A$, the concept of *accessibility* can be defined as an inductive predicate: an element $x \in A$ is accessible if every $y \in A$ such that $y \prec x$ is accessible:

$$\forall x : A \bullet (\forall y : A \bullet x \prec y \Rightarrow (Acc \ \prec \ y)) \Rightarrow (Acc \ \prec \ x) \tag{2}$$

and the relation $\prec \subseteq A \times A$ is *well–founded* if $A = Acc(A, \prec)$.

Section *generalidades.*

Require *Arith.*

Variables (*A*:*Set*) (*R*:*A*→*A*→*Prop*).

Print *Acc.*
Check Acc_intro.
Check Acc_inv.

Definition *wfis*:=∀ (*B*: *A*→*Set*),(∀(*x*:*A*),(∀ (*y*:*A*),(*R y x*)->(*B y*))->(*B x*))->
∀(*a*:*A*),(*B a*).

Definition *wfip*:=∀ (*P*:*A*→*Prop*),(∀ (*x*:*A*),(∀ (*y*:*A*),(*R y x*)->(*P y*))->(*P x*))->
∀(*a*:*A*),(*P a*).

Theorem *wf_wfis* : (*well_founded R*) →*wfis.*
Proof.
unfold wfis.
intros wf family wfp element.
elim (wf element).
intros; auto.
Qed.

Theorem *wf_wfip*:(*well_founded R*) → *wfip.*
Proof.
intro H.
red in H.

```
red.
intros.
elim (H a).
intros; auto.
Qed.

Theorem wfip_wf:wfip → (well_founded R).
Proof.
intro H.
red in H.
red; split; auto.
intros.
apply (H (fun (u:A)=>(Acc R u))).
intros x cla.
split; intros y0 H00.
apply cla.
auto.
Qed.

Section proj.

Variable B:A→Prop.
Definition pr1:=fun (H:{z:A|B z})=>let (a,_):= H in a.

Definition pr2:=fun (H:{z:A|B z})=>let (a,p) return (B (pr1 H)) := H in p.

Check pr1.
End proj.

Definition p1:=fun y:A=>(pr1 (fun _:A ⇒ (Acc R y))).

Check pr2.
Definition p2:=fun y:A=>(pr2 (fun _:A ⇒ (Acc R y))).
Check p2.

Lemma nec : ∀(x:A),(∀(y:A),(R y x)
->{a:A|(Acc R y)})->{a:A|(Acc R x)}.
Proof.
intros.
split with x.
split.
intros.
apply (p2 y (H y H0)).
Qed.

Theorem wfis_wf :wfis → (well_founded R).
Proof.
intro H.
```

```
red in H.
red.
intros.
apply (p2 a (H (fun(a:A)=>{z:A | (Acc R a)}) nec a)).
Qed.
```

```
Lemma norefl_acc:∀(x:A),(Acc R x)->~(R x x).
Proof.
red.
intros x H.
elim H.
intros x0 H0 H1 H2.
apply (H1 x0); auto.
Qed.
```

```
Theorem norefl_Wf:(well_founded R)->∀(a:A),~(R a a).
Proof.
intros wf a.
red in wf.
exact (norefl_acc a (wf a)).
Qed.
```

```
Definition shift:=fun(s:nat→A)=>fun(n:nat)=>(s (S n)).
```

```
Lemma shift_triv:∀(s:nat→A),(shift s O)=(s (1)).
Proof.
unfold shift.
auto.
Qed.
```

```
Definition Desc_chain := fun(s:nat→A)=>∀ (i:nat),(R (s (S i)) (s i)).
```

```
Definition no_Desc_chain:=fun(a:A)=>∀ (s:nat→A),(s O)=a->~(Desc_chain s).
```

```
Lemma wfip_no_Desc_chain:wfip→∀(a:A),(no_Desc_chain a).
Proof.
unfold wfip.
intros H x.
apply (H no_Desc_chain).
intros.
red.
red in H0.
intros.
red; intros.
red in H2.
rewrite ← H1 in H0.
```

```
red in H0.
apply (H0 (s (1)) (H2 O) (shift s) ).
simpl.
unfold shift.
auto.

red.
unfold shift.
intro i.
apply H2.
Qed.

Lemma Desc_chain_nowf:(∃ s:nat→A,(Desc_chain s))->~wfip.
intros.
case H;intros s p.
clear H.
intro.
assert (∀(a:A),(no_Desc_chain a)).
apply wfip_no_Desc_chain;auto.
Check (H0 (s 0)).
unfold no_Desc_chain in H0.
Print eq.
case (H0 (s 0) s (refl_equal (s 0))).
auto.
Qed.
```