Appendix A

Example Proof

Herein we show in detail a proof using Coq's deductive system. We do not intend to present in detail Coq's specification language Gallina, we simply intend to give a general idea of how a proof is conducted in Coq.

We show as an example the proof of `ac4i`

$(\forall i, s\ (i.actual(s) \Rightarrow (\exists a\ i.actual(do(a, s))))$

In the following all the statements preceded by < represent user input. The remaining lines represent Coq's output.

We start by introducing the theorem we intend to prove. This is done by using the command `Theorem`.

```
Coq < Theorem ac4i : (i:nat)(s:(sit act' i))
        (actual actual_line s)→
        (Ex [a:(act act' i)] (actual actual_line (do a s))).
1 subgoal

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  ============================
   (i:nat)
    (s:(sit act' i))
     (actual actual_line s)
      →(Ex [a:(act act' i)](actual actual_line (do a s)))
```

The system shows the hypothesis followed by the current goal. We have as hypothesis all the variables declared within the section. In our case we have a set of actions, a set of states, a set of fluents, an interpretation for *holds*, [_], *poss* and *actual*.

The first step is to unfold `actual` into it's definition.

```
ac4i < Unfold actual.
1 subgoal

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  ============================
   (i:nat)
    (s:(sit act' i))
     (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
      →(Ex [a:(act act' i)]
             (Ex [n:nat]
                  (nth (sit act' i) n (str_actual actual_line i))
                   =(do a s)))
```

The tactic `Intro` introduces as hypothesis the natural number `i`, the situation of object i `s` and the left side of the implication.

```
ac4i < Intros.
1 subgoal

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  ============================
   (Ex [a:(act act' i)]
        (Ex [n:nat]
             (nth (sit act' i) n (str_actual actual_line i))=
             (do a s)))
```

The tactic `Inversion` is used to obtain all the constraints necessary for a given inductive predicate to hold. If we apply `Inversion` to the hypothesis `H` we obtain as hypothesis a natural number `x` and the guarantee that the situation `s` is in the position `x` of the stream of actual situations.

```
ac4i < Inversion H.
1 subgoal

  act' :  nat→Set
  stt :  nat→Set
```

```
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  x :  nat
  H0 :  (nth (sit act' i) x (str_actual actual_line i))=s
  ============================
   (Ex [a:(act act' i)]
        (Ex [n:nat]
             (nth (sit act' i) n (str_actual actual_line i))=
             (do a s)))
```

To prove our goal we must provide the action `a` and the natural number `n` such that `(nth (sit act' i) n (str_actual actual_line i))=(do a s)))`. Our definition of *actual* guarantees that:

```
(i:nat)(n:nat)(Ex [a:(act i)]
((nth (sit i) (S n) (str_actual i))=
(do a (nth (sit i) n (str_actual i))))
∧ (poss i a (ss i (nth (sit i) n (str_actual i))))).
```

Thus we introduce this hypothesis, for the particular case of `i` and `x`. This is done with the tactic `Cut`. This tactic allows one to prove a goal `G` as a consequence of a statement `S`. It transforms the goal into `S→G` and yields a new subgoal `S`.

```
ac4i < Cut (Ex [a:(act act' i)]
           (nth (sit act' i) (S x) (actual_line i))
           =(do a (nth (sit act' i) x (actual_line i)))
           ∧(poss i a (ss i (nth (sit act' i) x (actual_line i))))).

2 subgoals

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  x :  nat
  H0 :  (nth (sit act' i) x (str_actual actual_line i))=s
  ============================
   (Ex [a:(act act' i)]
        (nth (sit act' i) (S x) (str_actual actual_line i))
         =(do a (nth (sit act' i) x (str_actual actual_line i)))
```

```
        ∧(f_poss poss a
            (f_ss ss
                (nth (sit act' i) x (str_actual actual_line i)))))
    →(Ex [a:(act act' i)]
           (Ex [n:nat]
                (nth (sit act' i) n (str_actual actual_line i))
                 =(do a s)))

subgoal 2 is:
 (Ex [a:(act act' i)]
      (nth (sit act' i) (S x) (str_actual actual_line i))
       =(do a (nth (sit act' i) x (str_actual actual_line i)))
      ∧(f_poss poss a
          (f_ss ss
                (nth (sit act' i) x (str_actual actual_line i)))))


ac4i < Intro.
2 subgoals

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  x :  nat
  H0 :  (nth (sit act' i) x (str_actual actual_line i))=s
  H1 :  (Ex [a:(act act' i)]
            (nth (sit act' i) (S x) (str_actual actual_line i))
             =(do a (nth (sit act' i) x (str_actual actual_line i)))
            ∧(f_poss poss a
                (f_ss ss
                  (nth (sit act' i) x (str_actual actual_line i)))))
  ============================
   (Ex [a:(act act' i)]
        (Ex [n:nat]
             (nth (sit act' i) n (str_actual actual_line i))
             =(do a s)))

subgoal 2 is:
 (Ex [a:(act act' i)]
      (nth (sit act' i) (S x) (str_actual actual_line i))
       =(do a (nth (sit act' i) x (str_actual actual_line i)))
      ∧(f_poss poss a
          (f_ss ss (nth (sit act' i) x (str_actual actual_line i)))))
```

By applying the tactic `Inversion` to the hypothesis H1 we obtain the wanted action.

```
ac4i < Inversion H1.
2 subgoals

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  x :  nat
  H0 :  (nth (sit act' i) x (str_actual actual_line i))=s
  H1 :  (Ex [a:(act act' i)]
            (nth (sit act' i) (S x) (str_actual actual_line i))
             =(do a (nth (sit act' i) x (str_actual actual_line i)))
            ∧(f_poss poss a
                (f_ss ss
                  (nth (sit act' i) x (str_actual actual_line i)))))
  x0 :  (act act' i)
  H2 :  (nth (sit act' i) (S x) (str_actual actual_line i))
        =(do x0 (nth (sit act' i) x (str_actual actual_line i)))
       ∧(f_poss poss x0
           (f_ss ss (nth (sit act' i) x (str_actual actual_line i))))
  ============================
   (Ex [a:(act act' i)]
        (Ex [n:nat]
             (nth (sit act' i) n (str_actual actual_line i))=
             (do a s)))

subgoal 2 is:
 (Ex [a:(act act' i)]
      (nth (sit act' i) (S x) (str_actual actual_line i))
       =(do a (nth (sit act' i) x (str_actual actual_line i)))
      ∧(f_poss poss a
          (f_ss ss (nth (sit act' i) x (str_actual actual_line i)))))
```

We are now ready to provide the action and the natural number required in the goal. The action is `x0`

```
ac4i < Exists x0.
2 subgoals

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
```

```
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  x :  nat
  H0 :  (nth (sit act' i) x (str_actual actual_line i))=s
  H1 :  (Ex [a:(act act' i)]
            (nth (sit act' i) (S x) (str_actual actual_line i))
             =(do a (nth (sit act' i) x (str_actual actual_line i)))
            ∧(f_poss poss a
                (f_ss ss
                  (nth (sit act' i) x (str_actual actual_line i)))))
  x0 :  (act act' i)
  H2 :  (nth (sit act' i) (S x) (str_actual actual_line i))
        =(do x0 (nth (sit act' i) x (str_actual actual_line i)))
       ∧(f_poss poss x0
           (f_ss ss (nth (sit act' i) x (str_actual actual_line i))))
  ============================
   (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=
             (do x0 s))

subgoal 2 is:
 (Ex [a:(act act' i)]
      (nth (sit act' i) (S x) (str_actual actual_line i))
       =(do a (nth (sit act' i) x (str_actual actual_line i)))
      ∧(f_poss poss a
          (f_ss ss (nth (sit act' i) x (str_actual actual_line i)))))
```

From `H0` we know that `s` is at position `x` of the stream. Thus from H2 `(do x0 s)` must be at position (S x).

```
ac4i < Exists (S x).
2 subgoals

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  x :  nat
  H0 :  (nth (sit act' i) x (str_actual actual_line i))=s
  H1 :  (Ex [a:(act act' i)]
            (nth (sit act' i) (S x) (str_actual actual_line i))
             =(do a (nth (sit act' i) x (str_actual actual_line i)))
            ∧(f_poss poss a
                (f_ss ss
```

```
                 (nth (sit act' i) x (str_actual actual_line i)))))
  x0 :  (act act' i)
  H2 :  (nth (sit act' i) (S x) (str_actual actual_line i))
         =(do x0 (nth (sit act' i) x (str_actual actual_line i)))
        ∧(f_poss poss x0
            (f_ss ss (nth (sit act' i) x (str_actual actual_line i))))
  ============================
   (nth (sit act' i) (S x) (str_actual actual_line i))=(do x0 s)

subgoal 2 is:
 (Ex [a:(act act' i)]
      (nth (sit act' i) (S x) (str_actual actual_line i))
       =(do a (nth (sit act' i) x (str_actual actual_line i)))
      ∧(f_poss poss a
          (f_ss ss (nth (sit act' i) x (str_actual actual_line i)))))
```

By inverting `H2` we eliminate the conjunction and deduce `H3` and `H4`

```
ac4i < Inversion H2.
2 subgoals

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  x :  nat
  H0 :  (nth (sit act' i) x (str_actual actual_line i))=s
  H1 :  (Ex [a:(act act' i)]
            (nth (sit act' i) (S x) (str_actual actual_line i))
             =(do a (nth (sit act' i) x (str_actual actual_line i)))
            ∧(f_poss poss a
                (f_ss ss
                  (nth (sit act' i) x (str_actual actual_line i)))))
  x0 :  (act act' i)
  H2 :  (nth (sit act' i) (S x) (str_actual actual_line i))
         =(do x0 (nth (sit act' i) x (str_actual actual_line i)))
        ∧(f_poss poss x0
            (f_ss ss (nth (sit act' i) x (str_actual actual_line i))))
  H3 :  (nth (sit act' i) (S x) (str_actual actual_line i))
         =(do x0 (nth (sit act' i) x (str_actual actual_line i)))
  H4 :  (f_poss poss x0
          (f_ss ss (nth (sit act' i) x (str_actual actual_line i))))
  ============================
   (nth (sit act' i) (S x) (str_actual actual_line i))=(do x0 s)

subgoal 2 is:
 (Ex [a:(act act' i)]
```

```
      (nth (sit act' i) (S x) (str_actual actual_line i))
       =(do a (nth (sit act' i) x (str_actual actual_line i)))
      ∧(f_poss poss a
          (f_ss ss (nth (sit act' i) x (str_actual actual_line i)))))
```

Now we use `H0` and replace `s` with (nth (sit act' i) x (str_actual actual_line i)) in `H3`. This way we obtain the goal.

```
ac4i < Rewrite H0 in H3.
2 subgoals

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  x :  nat
  H0 :  (nth (sit act' i) x (str_actual actual_line i))=s
  H1 :  (Ex [a:(act act' i)]
            (nth (sit act' i) (S x) (str_actual actual_line i))
             =(do a (nth (sit act' i) x (str_actual actual_line i)))
            ∧(f_poss poss a
                (f_ss ss
                  (nth (sit act' i) x (str_actual actual_line i)))))
  x0 :  (act act' i)
  H2 :  (nth (sit act' i) (S x) (str_actual actual_line i))
        =(do x0 (nth (sit act' i) x (str_actual actual_line i)))
       ∧(f_poss poss x0
           (f_ss ss (nth (sit act' i) x (str_actual actual_line i))))
  H3 :  (nth (sit act' i) (S x) (str_actual actual_line i))=(do x0 s)
  H4 :  (f_poss poss x0
         (f_ss ss (nth (sit act' i) x (str_actual actual_line i))))
  ============================
   (nth (sit act' i) (S x) (str_actual actual_line i))=(do x0 s)

subgoal 2 is:
 (Ex [a:(act act' i)]
      (nth (sit act' i) (S x) (str_actual actual_line i))
       =(do a (nth (sit act' i) x (str_actual actual_line i)))
      ∧(f_poss poss a
          (f_ss ss (nth (sit act' i) x (str_actual actual_line i)))))
```

The tactic `Auto` is able to recognize the goal as hypothesis `H3`. In alternative we could have used the command `Exact H3.`, `Apply H3.` or simply `Assumption.`

```
ac4i < Auto.
```

```
1 subgoal

  act' :  nat→Set
  stt :  nat→Set
  flt :  nat→Set
  holds :  (i:nat)(flt i)→(stt i)→Prop
  ss :  (ss_struc act' stt)
  poss :  (poss_struc ss)
  actual_line :  (actual_struc poss)
  i :  nat
  s :  (sit act' i)
  H : (Ex [n:nat](nth (sit act' i) n (str_actual actual_line i))=s)
  x :  nat
  H0 :  (nth (sit act' i) x (str_actual actual_line i))=s
  ============================
   (Ex [a:(act act' i)]
        (nth (sit act' i) (S x) (str_actual actual_line i))
         =(do a (nth (sit act' i) x (str_actual actual_line i)))
        ∧(f_poss poss a
            (f_ss ss (nth (sit act' i) x (str_actual actual_line i)))))
```

Finally we must prove the hypothesis that we introduced. Since it is the field **actual_do** of the structure actual_struc it may be proved with **Auto** (or alternatively with **Apply actual_do**).

```
ac4i < Auto.
Subtree proved!
```

With the command **Qed** we save the proof. A proof term is extracted and associated to the name declared.

```
ac4i < Qed.
Unfold actual.
Intros.
Inversion H.
Cut (Ex
      [a:(act act' i)]
       (nth (sit act' i) (S x) (actual_line i))
        =(do a (nth (sit act' i) x (actual_line i)))
       ∧(poss i a (ss i (nth (sit act' i) x (actual_line i))))).
Intros.
Inversion H1.
Exists x0.
Exists (S x).
Inversion H2.
Rewrite H0 in H3.
Auto.

Auto.

ac4i is defined
```