The Hypergame Paradox in Coq

Robbert Krebbers

Århus University

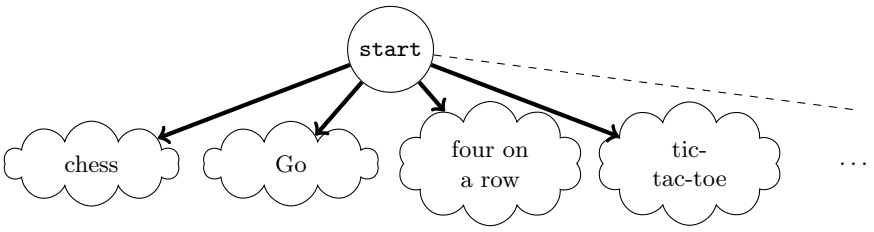
Abstract. Let's play a game called *hypergame*. You pick any game that is guaranteed to end after a finite number of moves (chess, Go, four on a row, tic-tac-toe, *etc*), and then we play it.
It is easy to see that hypergame ends after a finite number of moves: the number of moves is just one more than it would take to play the chosen game. But now that we have established finiteness of hypergame, we may just choose hypergame while playing hypergame, and keep on doing this. This shows that hypergame is infinite, which is a contradiction.
I will present a simple Coq formalization of the hypergame paradox. In contrast to the Girard or Hurkens paradox, my formalization uses Coq's inductive types instead of Church encodings. Albeit less general (it no longer provides a paradox in plain $\lambda*$), this approach gives rise to a much better understandable formalization.

1 Formalization of Hypergame

We represent finite games as strongly normalizing transition relations (called `sn` below) $R \subseteq A \times A$ over a set of states A. Games do not have an initial state, any state may be used to initiate the game. Hypergame is formalized as:

```
Inductive sn {A} (R : A → A → Prop) (x : A) : Prop :=
  | sn_intro : (∀ y : A, R x y → sn R y) → sn R x.
Inductive state :=
  | start : state
  | game {A} (R : A → A → Prop) (Rsn : ∀ x, sn R x) : A → state.
Inductive step : state → state → Prop :=
  | step_start {A} (R : A → A → Prop) Rsn x :
     step start (game R Rsn x)
  | step_game {A} (R : A → A → Prop) Rsn x y :
     R x y → step (game R Rsn x) (game R Rsn y).
```

Starting in state `start`, one can choose any game R by transitioning to the state `game` R R_{sn} x, where R_{sn} is a proof that R is strongly normalizing and x some state of the chosen game. This can be visualized as follows:



2 Finiteness and Infiniteness of Hypergame

We prove that hypergame is finite by case analysis on the initial step, and then by induction on the proof of strong normalization of the chosen game.

```
Lemma help_wf A (R : A → A → Prop) Rsn x : sn step (game R Rsn x).
Proof.
  induction (Rsn x) as [x _ IH]; constructor; intros S H; revert H IH.
  change (step (game R Rsn x) S → match game R Rsn x with
  | start => True
  | game R Rsn x => (∀ y, R x y → sn step (game R Rsn y)) → sn step S
  end); destruct 1; auto. (* manual inversion using match *)
Qed.
Lemma hypergame_wf S : sn step S.
Proof. constructor; destruct 1; auto using help_wf. Qed.
```

In order to prove that hypergame is not strongly normalizing, we exhibit the infinite game: `start`, hypergame, hypergame, hypergame, ...

```
Fixpoint infinite (n : nat) : state :=
  match n with
  | 0 => start | S n => game step hypergame_wf (infinite n)
  end.
Lemma infinite_step n : step (infinite n) (infinite (S n)).
Proof. induction n; simpl; constructor; auto. Qed.
Lemma hypergame_not_wf : ¬sn step (infinite 0).
Proof.
  intros Hwf; cut (∀ S, sn step S → ∀ n, S ≠ infinite n).
  { intros help; apply (help _ Hwf 0); auto. } (* strengthen the IH *)
  induction 1 as [S _ IH]; intros n ->.
  apply (IH _ (infinite_step n) (S n)); auto.
Qed.
Definition paradox : False := hypergame_not_wf (hypergame_wf _).
```

It is important to note that `infinite` will not be accepted by ordinary versions of Coq as it contains the subterm `@game state` (implicit argument displayed using `@`), where the constructor `game` of the inductive type `state` is applied to `state` itself. This results in a "universe inconsistency". However, Coq 8.5 has introduced the (unsafe) `-type-in-type` flag that disables universe checking.

3 Sources

The sources are available at `http://robbertkrebbers.nl/misc/hypergame.v` and compile with Coq 8.5 beta 2 with the `-type-in-type` flag enabled. The file is self-contained, and consists of 38 lines of code (including white space).

References

1. William S. Zwicker. Playing Games with Games: The Hypergame Paradox. *The American Mathematical Monthly*, 94(6):507–514, 1987.