Gale-Stewart Games

Sebastiaan J. C. Joosten

May 26, 2024

Abstract

This is a formalisation of the main result of Gale and Stewart from 1953, showing that closed finite games are determined. This property is now known as the Gale Stewart Theorem. While the original paper shows some additional theorems as well, we only formalize this main result, but do so in a somewhat general way. We formalize games of a fixed arbitrary length, including infinite length, using co-inductive lists, and show that defensive strategies exist unless the other player is winning. For closed games, defensive strategies are winning for the closed player, proving that such games are determined. For finite games, which are a special case in our formalisation, all games are closed.

Contents

1 Introduction

The original paper from Gale and Stewart [2] uses a function to point to a previous position. This encoding of sequences is not followed in this formalization, as it is not the way we think of games these days. Instead, we follow the approach taken in the formalization of Parity Games [1], where co-inductive lists are used to talk about possibly infinite plays. Although we rely on the Parity Games theory for some of the theorems about co-inductive lists, none of the notions about games are shared with that formalization.

We have proven some basic lemmas about prefixes, extended naturals (natural numbers plus infinity), and defined a function 'alternate' alternating lists. We have done this in separate Isabelle theory files, so that they can be reused independently without depending on the formalizations of infinite games presented here. In the same way this formalization is giving a nod to the parity games formalization. In this document, we only present the alternating lists, as this theory file contains new definitions, which are relevant preliminaries to know about. The additional lemmas about prefixes and extended natural numbers are less essential, they only contain 'obvious' properties, so we have left those theory files out of this document.

2 Alternating lists

In lists where even and odd elements play different roles, it helps to define functions to take out the even elements. We defined the function (l)alternate on (coinductive) lists to do exactly this, and define certain properties.

theory *AlternatingLists*
  imports *MoreCoinductiveList2*
begin

The functions "alternate" and "lalternate" are our main workhorses: they take every other item, so every item at even indices.

fun *alternate* **where**
  alternate Nil = *Nil* |
  alternate (*Cons x xs*) = *Cons x* (*alternate* (*tl xs*))

"lalternate" takes every other item from a co-inductive list.

primcorec *lalternate* :: *'a llist* ⇒ *'a llist*
  where
    lalternate xs = (*case xs of LNil* ⇒ *LNil* |
                                  (*LCons x xs*) ⇒ *LCons x* (*lalternate* (*ltl xs*)))

lemma *lalternate-ltake*:
  ltake (*enat n*) (*lalternate xs*) = *lalternate* (*ltake* (*2*∗*n*) *xs*)
⟨*proof*⟩

lemma *lalternate-llist-of*[*simp*]:
  lalternate (*llist-of xs*) = *llist-of* (*alternate xs*)
⟨*proof*⟩

lemma *lalternate-finite-helper*:
  assumes *lfinite* (*lalternate xs*)
  shows *lfinite xs*
⟨*proof*⟩

lemma *alternate-list-of*:

assumes *lfinite xs*
shows *alternate* (*list-of xs*) = *list-of* (*lalternate xs*)
⟨*proof*⟩

lemma *alternate-length*:
length (*alternate xs*) = (*1*+*length xs*) *div 2*
⟨*proof*⟩

lemma *lalternate-llength*:
llength (*lalternate xs*) ∗ *2* = (*1*+*llength xs*) ∨ *llength* (*lalternate xs*) ∗ *2* = *llength xs*
⟨*proof*⟩

lemma *lalternate-finite*[*simp*]:
shows *lfinite* (*lalternate xs*) = *lfinite xs*
⟨*proof*⟩

lemma *nth-alternate*:
assumes *2*∗*n* < *length xs*
shows *alternate xs* ! *n* = *xs* ! (*2* ∗ *n*)
⟨*proof*⟩

lemma *lnth-lalternate*:
assumes *2*∗*n* < *llength xs*
shows *lalternate xs* $ *n* = *xs* $ (*2* ∗ *n*)
⟨*proof*⟩

lemma *lnth-lalternate2*[*simp*]:
assumes *n* < *llength* (*lalternate xs*)
shows *lalternate xs* $ *n* = *xs* $ (*2* ∗ *n*)
⟨*proof*⟩

end

3 Gale Stewart Games

Gale Stewart Games are infinite two player games.

theory *GaleStewartGames*
imports *AlternatingLists MorePrefix MoreENat*
begin

3.1 Basic definitions and their properties.

A GSgame G(A) is defined by a set of sequences that denote the winning games for the first player. Our notion of GSgames generalizes both finite and infinite games by setting a game length. Note that the type of n is 'enat' (extended nat): either a nonnegative integer or infinity. Our only requirement on GSgames is that the winning games must have the length as specified as the length of the game. This helps certain theorems about winning look a bit more natural.

locale *GSgame* =
  fixes *A N*
  assumes *length*:$\forall$ *e*$\in$*A*. *llength e* = *2*∗*N*
begin

A position is a finite sequence of valid moves.

definition *position* **where**
  position (*e*::′*a list*) ≡ *length e* ≤ *2*∗*N*

lemma *position-maxlength-cannotbe-augmented*:
assumes *length p* = *2*∗*N*
shows ¬ *position* (*p* @ [*m*])
⟨*proof*⟩

A play is a sequence of valid moves of the right length.

definition *play* **where**
  play (*e*::′*a llist*) ≡ *llength e* = *2*∗*N*

lemma *plays-are-positions-conv*:
  shows *play* (*llist-of p*) ⟷ *position p* ∧ *length p* = *2*∗*N*
⟨*proof*⟩

lemma *finite-plays-are-positions*:
  assumes *play p lfinite p*
  shows *position* (*list-of p*)
⟨*proof*⟩

end

We call our players Even and Odd, where Even makes the first move. This means that Even is to make moves on plays of even length, and Odd on the others. This corresponds nicely to Even making all the moves in an even position, as the 'nth' and 'lnth' functions as predefined in Isabelle's library count from 0. In literature the players are sometimes called I and II.

A strategy for Even/Odd is simply a function that takes a position of even/odd length and returns a move. We use total functions for strategies. This means that their Isabelle-type determines that it is a strategy. Consequently, we do not have a definition of 'strategy'. Nevertheless, we will use σ as a letter to indicate when something is a strategy. We can combine two strategies into one function, which gives a collective strategy that we will refer to as the joint strategy.

definition *joint-strategy* :: (′*b list* ⇒ ′*a*) ⇒ (′*b list* ⇒ ′*a*) ⇒ (′*b list* ⇒ ′*a*) **where**
  joint-strategy σ_e σ_o *p* = (*if even* (*length p*) *then* σ_e *p else* σ_o *p*)

Following a strategy leads to an infinite sequence of moves. Note that we are not in the context of 'GSGame' where 'N' determines the length of our plays: we just let sequences go on ad infinitum here. Rather than reasoning about our own recursive definitions, we build this infinite sequence by reusing definitions that are already in place. We do this by first defining all prefixes of the infinite sequence we are interested in. This gives an infinite list such that the nth element is of length n. Note that this definition allows us to talk about how a strategy would continue if it were played from an arbitrary position (not necessarily one that is reached via that strategy).

definition *strategy-progression* **where**
strategy-progression σ *p* = *lappend* (*llist-of* (*prefixes p*)) (*ltl* (*iterates* (*augment-list* σ) *p*))

lemma *induced-play-infinite*:
¬ *lfinite* (*strategy-progression* σ *p*)
⟨*proof*⟩

lemma *plays-from-strategy-lengths*[*simp*]:
length (*strategy-progression* σ *p* \$ *i*) = *i*
⟨*proof*⟩

lemma *length-plays-from-strategy*[*simp*]:
llength (*strategy-progression* σ *p*) = ∞
⟨*proof*⟩
lemma *length-ltl-plays-from-strategy*[*simp*]:
llength (*ltl* (*strategy-progression* σ *p*)) = ∞
⟨*proof*⟩

lemma *plays-from-strategy-chain-Suc*:
shows *prefix* (*strategy-progression* σ *p* \$ *n*) (*strategy-progression* σ *p* \$ *Suc n*)
⟨*proof*⟩

lemma *plays-from-strategy-chain*:
shows *n* ≤ *m* ⟹ *prefix* (*strategy-progression* σ *p* \$ *n*) (*strategy-progression* σ *p* \$ *m*)
⟨*proof*⟩

lemma *plays-from-strategy-remains-const*:
assumes *n* ≤ *i*
shows *take n* (*strategy-progression* σ *p* \$ *i*) = *strategy-progression* σ *p* \$ *n*
⟨*proof*⟩

lemma *infplays-augment-one*[*simp*]:
strategy-progression σ (*p* @ [σ *p*]) = *strategy-progression* σ *p*
⟨*proof*⟩

lemma *infplays-augment-many*[*simp*]:
strategy-progression σ ((*augment-list* σ ^^ *n*) *p*) = *strategy-progression* σ *p*

⟨*proof*⟩

lemma *infplays-augment-one-joint*[*simp*]:
even (*length p*) $\Longrightarrow$ *strategy-progression* (*joint-strategy* σ_e σ_o) (*augment-list* σ_e *p*)
= *strategy-progression* (*joint-strategy* σ_e σ_o) *p*
odd (*length p*) $\Longrightarrow$ *strategy-progression* (*joint-strategy* σ_e σ_o) (*augment-list* σ_o *p*)
= *strategy-progression* (*joint-strategy* σ_e σ_o) *p*
⟨*proof*⟩

Following two different strategies from a single position will lead to the same plays if the strategies agree on moves played after that position. This lemma allows us to ignore the behavior of strategies for moves that are already played.

lemma *infplays-eq*:
assumes $\bigwedge$ *p'*. *prefix p p'* $\Longrightarrow$ *augment-list s1 p'* = *augment-list s2 p'*
shows *strategy-progression s1 p* = *strategy-progression s2 p*
⟨*proof*⟩

context *GSgame*
begin

By looking at the last elements of the infinite progression, we can get a single sequence, which we trim down to the right length. Since it has the right length, this always forms a play. We therefore name this the 'induced play'.

definition *induced-play* **where**
induced-play $\sigma \equiv$ *ltake* (*2***N*) *o lmap last o ltl o strategy-progression* σ

lemma *induced-play-infinite-le*[*simp*]:
enat x < *llength* (*strategy-progression* σ *p*)
enat x < *llength* (*lmap f* (*strategy-progression* σ *p*))
enat x < *llength* (*ltake* (*2***N*) (*lmap f* (*strategy-progression* σ *p*))) $\longleftrightarrow$ *x* < *2***N*
⟨*proof*⟩

lemma *induced-play-is-lprefix*:
assumes *position p*
shows *lprefix* (*llist-of p*) (*induced-play* σ *p*)
⟨*proof*⟩

lemma *length-induced-play*[*simp*]:
llength (*induced-play s p*) = *2* * *N*
⟨*proof*⟩

lemma *induced-play-lprefix-non-positions*:
assumes *length* (*p*::*'a list*) $\geq$ *2* * *N*
shows *induced-play* σ *p* = *ltake* (*2* * *N*) (*llist-of p*)
⟨*proof*⟩

lemma *infplays-augment-many-lprefix*[*simp*]:
shows *lprefix* (*llist-of* ((*augment-list* σ ^^ *n*) *p*)) (*induced-play* σ *p*)
= *position* ((*augment-list* σ ^^ *n*) *p*) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

3.2 Winning strategies

A strategy is winning (in position p) if, no matter the moves by the other player, it leads to a sequence in the winning set.

definition *strategy-winning-by-Even* **where**
strategy-winning-by-Even σ_e *p* ≡ (∀ σ_o. *induced-play* (*joint-strategy* σ_e σ_o) *p* ∈ *A*)
definition *strategy-winning-by-Odd* **where**
strategy-winning-by-Odd σ_o *p* ≡ (∀ σ_e. *induced-play* (*joint-strategy* σ_e σ_o) *p* ∉ *A*)

It immediately follows that not both players can have a winning strategy.

lemma *at-most-one-player-winning*:
shows ¬ (∃ σ_e. *strategy-winning-by-Even* σ_e *p*) ∨ ¬ (∃ σ_o. *strategy-winning-by-Odd* σ_o *p*)
⟨*proof*⟩

If a player whose turn it is not makes any move, winning strategies remain winning. All of the following proofs are duplicated for Even and Odd, as the game is entirely symmetrical. These 'dual' theorems can be obtained by considering a game in which an additional first and final move are played yet ignored, but it is quite convenient to have both theorems at hand regardless, and the proofs are quite small, so we accept the code duplication.

lemma *any-moves-remain-winning-Even*:
assumes *odd* (*length p*) *strategy-winning-by-Even* σ *p*
shows *strategy-winning-by-Even* σ (*p* @ [*m*])
⟨*proof*⟩

lemma *any-moves-remain-winning-Odd*:
assumes *even* (*length p*) *strategy-winning-by-Odd* σ *p*
shows *strategy-winning-by-Odd* σ (*p* @ [*m*])
⟨*proof*⟩

If a player does not have a winning strategy, a move by that player will not give it one.

lemma *non-winning-moves-remains-non-winning-Even*:
assumes *even* (*length p*) ∀ σ. ¬ *strategy-winning-by-Even* σ *p*
shows ¬ *strategy-winning-by-Even* σ (*p* @ [*m*])
⟨*proof*⟩

lemma *non-winning-moves-remains-non-winning-Odd*:

assumes *odd* (*length p*) ∀ *σ*. ¬ *strategy-winning-by-Odd σ p*
shows ¬ *strategy-winning-by-Odd σ* (*p* @ [*m*])
⟨*proof*⟩

If a player whose turn it is makes a move according to its stragey, the new position will remain winning.

lemma *winning-moves-remain-winning-Even*:
assumes *even* (*length p*) *strategy-winning-by-Even σ p*
shows *strategy-winning-by-Even σ* (*p* @ [*σ p*])
⟨*proof*⟩

lemma *winning-moves-remain-winning-Odd*:
assumes *odd* (*length p*) *strategy-winning-by-Odd σ p*
shows *strategy-winning-by-Odd σ* (*p* @ [*σ p*])
⟨*proof*⟩

We speak of winning positions as those positions in which the player has a winning strategy. This is mainly for presentation purposes.

abbreviation *winning-position-Even* **where**
winning-position-Even p ≡ *position p* ∧ (∃ *σ*. *strategy-winning-by-Even σ p*)
abbreviation *winning-position-Odd* **where**
winning-position-Odd p ≡ *position p* ∧ (∃ *σ*. *strategy-winning-by-Odd σ p*)

lemma *winning-position-can-remain-winning-Even*:
assumes *even* (*length p*) ∀ *m*. *position* (*p* @ [*m*]) *winning-position-Even p*
shows ∃ *m*. *winning-position-Even* (*p* @ [*m*])
⟨*proof*⟩

lemma *winning-position-can-remain-winning-Odd*:
assumes *odd* (*length p*) ∀ *m*. *position* (*p* @ [*m*]) *winning-position-Odd p*
shows ∃ *m*. *winning-position-Odd* (*p* @ [*m*])
⟨*proof*⟩

lemma *winning-position-will-remain-winning-Even*:
assumes *odd* (*length p*) *position* (*p* @ [*m*]) *winning-position-Even p*
shows *winning-position-Even* (*p* @ [*m*])
⟨*proof*⟩

lemma *winning-position-will-remain-winning-Odd*:
assumes *even* (*length p*) *position* (*p* @ [*m*]) *winning-position-Odd p*
shows *winning-position-Odd* (*p* @ [*m*])
⟨*proof*⟩

lemma *induced-play-eq*:
assumes ∀ *p'*. *prefix p p'* ⟶ (*augment-list s1*) *p'* = (*augment-list s2*) *p'*
shows *induced-play s1 p* = *induced-play s2 p*
⟨*proof*⟩

end

end

3.3 Defensive strategies

A strategy is defensive if a player can avoid reaching winning positions. If the opponent is not already in a winning position, such defensive strategies exist. In closed games, a defensive strategy is winning for the closed player, so these strategies are a crucial step towards proving that such games are determined.

theory *GaleStewartDefensiveStrategies*
 imports *GaleStewartGames*
begin

context *GSgame*
begin

definition *move-defensive-by-Even* **where**
 move-defensive-by-Even m p ≡ *even* (*length p*) ⟶ ¬ *winning-position-Odd* (*p* @ [*m*])
definition *move-defensive-by-Odd* **where**
 move-defensive-by-Odd m p ≡ *odd* (*length p*) ⟶ ¬ *winning-position-Even* (*p* @ [*m*])

lemma *defensive-move-exists-for-Even*:
assumes [*intro*]:*position p*
shows *winning-position-Odd p* ∨ (∃ *m. move-defensive-by-Even m p*) (**is** *?w* ∨ *?d*)
⟨*proof*⟩

lemma *defensive-move-exists-for-Odd*:
assumes [*intro*]:*position p*
shows *winning-position-Even p* ∨ (∃ *m. move-defensive-by-Odd m p*) (**is** *?w* ∨ *?d*)
⟨*proof*⟩

definition *defensive-strategy-Even* **where**
defensive-strategy-Even p ≡ *SOME m. move-defensive-by-Even m p*
definition *defensive-strategy-Odd* **where**
defensive-strategy-Odd p ≡ *SOME m. move-defensive-by-Odd m p*

lemma *position-augment*:
 assumes *position* ((*augment-list f* ^^ *n*) *p*)
 shows *position p*
 ⟨*proof*⟩

lemma *defensive-strategy-Odd*:
 assumes ¬ *winning-position-Even p*

shows ¬ *winning-position-Even* $(((augment\text{-}list\ (joint\text{-}strategy\ \sigma_e\ defensive\text{-}strategy\text{-}Odd))$ $\frown\frown$ $n)\ p)$
⟨*proof*⟩

lemma *defensive-strategy-Even*:
assumes ¬ *winning-position-Odd p*
shows ¬ *winning-position-Odd* $(((augment\text{-}list\ (joint\text{-}strategy\ defensive\text{-}strategy\text{-}Even\ \sigma_o))$ $\frown\frown$ $n)\ p)$
⟨*proof*⟩

end

locale *closed-GSgame* = *GSgame* +
assumes *closed*:$e \in A \Longrightarrow \exists\ p.\ lprefix\ (llist\text{-}of\ p)\ e \wedge (\forall\ e'.\ lprefix\ (llist\text{-}of\ p)\ e' \longrightarrow llength\ e' = 2{*}N \longrightarrow e' \in A)$

locale *finite-GSgame* = *GSgame* +
assumes *fin*:$N \neq \infty$
begin

Finite games are closed games. As a corollary to the GS theorem, this lets us conclude that finite games are determined.

sublocale *closed-GSgame*
⟨*proof*⟩
end

context *closed-GSgame* **begin**
lemma *never-winning-is-losing-even*:
assumes *position p* $\forall\ n.$ ¬ *winning-position-Even* $(((augment\text{-}list\ \sigma)$ $\frown\frown$ $n)\ p)$
shows *induced-play* $\sigma\ p \notin A$
⟨*proof*⟩

lemma *every-position-is-determined*:
assumes *position p*
shows *winning-position-Even p* ∨ *winning-position-Odd p* (**is** *?we* ∨ *?wo*)
⟨*proof*⟩

end

end

3.4 Determined games

theory *GaleStewartDeterminedGames*
imports *GaleStewartDefensiveStrategies*
begin

locale *closed-GSgame* = *GSgame* +
  assumes *closed*:$e \in A \Longrightarrow \exists\ p.\ lprefix\ (llist\text{-}of\ p)\ e \wedge (\forall\ e'.\ lprefix\ (llist\text{-}of\ p)\ e' \longrightarrow llength\ e' = 2{*}N \longrightarrow e' \in A)$

locale *finite-GSgame* = *GSgame* +
  assumes *fin*:$N \neq \infty$
begin

Finite games are closed games. As a corollary to the GS theorem, this lets us conclude that finite games are determined.

sublocale *closed-GSgame*
⟨*proof*⟩
end

context *closed-GSgame* **begin**
lemma *never-winning-is-losing-even*:
  assumes *position p* $\forall\ n.\ \neg$ *winning-position-Even* (((*augment-list* σ) ^^ *n*) *p*)
  shows *induced-play* σ *p* $\notin A$
⟨*proof*⟩

By proving that every position is determined, this proves that every game is determined (since a game is determined if its initial position [] is)

lemma *every-position-is-determined*:
  assumes *position p*
  shows *winning-position-Even p* $\vee$ *winning-position-Odd p* (**is** *?we* $\vee$ *?wo*)
⟨*proof*⟩
lemma *empty-position*: *position* [] ⟨*proof*⟩
lemmas *every-game-is-determined* = *every-position-is-determined*[*OF empty-position*]

We expect that this theorem can be easier to apply without the 'position p' requirement, so we present that theorem as well.

lemma *every-position-has-winning-strategy*:
  shows ($\exists\ \sigma$. *strategy-winning-by-Even* σ *p*) $\vee$ ($\exists\ \sigma$. *strategy-winning-by-Odd* σ *p*) (**is** *?we* $\vee$ *?wo*)
⟨*proof*⟩

end

end

References

[1] C. Dittmann. Positional determinacy of parity games. *Archive of Formal Proofs*, Nov. 2015. https://isa-afp.org/entries/Parity_Game.html, Formal proof development.

[2] D. Gale and F. M. Stewart. Infinite games with perfect information. *Contributions to the Theory of Games*, 2(245-266):2–16, 1953.