Stefan **Bold**, Benedikt **Löwe**,
Thoralf **Räsch**, Johan **van Benthem** (*eds.*)
Foundations of the Formal Sciences V
Infinite Games

Nonmonotone Game Labellings

Tikitu de Jager and Benedikt Löwe*

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Nieuwe Doelenstraat 15
1012 CP Amsterdam, The Netherlands
S.T.deJager@uva.nl

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam, The Netherlands
bloewe@science.uva.nl

ABSTRACT. We extend the traditional Gale-Stewart algorithm (backward induction) for the combinatorial graph game to an asymmetric variant; the extension makes the algorithm non-monotonic, but a linear-time formulation is still possible.

1 Introduction

If **G** is a graph, the (symmetric) combinatorial game on **G** is played by two players pushing a token on the graph. Whoever moves the token into a terminal node, wins. An example of a game of this type is the game of *Nim* (removing matches from a number of rows of matches until the game board is empty). If **G** was cyclic, then it is not guaranteed that one of the players will push the token into a terminal node. An infinite walk through **G** is considered a draw in the combinatorial game.[1]

Combinatorial games are perfect information games with simple payoff sets, and thus by the Gale-Stewart Theorem determined [GalSte53]. You can determine the winner of the game by unfolding the game into a game

*The authors would like to thank Philipp Rohde (Aachen) and Nick Bezhanishvili (Amsterdam) for fruitful discussions of topics connected to different aspects of this paper.

[1]More details can be found in the four-volume second edition of *Winning Ways* by Berlekamp, Conway and Guy.

tree $\mathbf{T}_{\mathbf{G}}$, then labelling the tree via the Gale-Stewart labelling and read off the winner from the label of the node s.

However, analyzing combinatorial games via their game trees might not be optimal for several reasons:

Firstly, if $\mathbf{G}$ was cyclic, the game tree will be infinite and the labelling of the game tree will be an infinitary, possibly transfinite procedure.

Secondly, the Gale-Stewart procedure is not metamathematically parsimonious. There are computable trees with no computable winning strategy, and the Gale-Stewart theorem on determinacy of open games is equivalent to a nontrivial metamathematical statement of second-order arithmetic by a theorem of Steel.[2]

Thus, a labelling procedure directly on the graph that doesn't need unfolding into the game tree is a *desideratum*. The existence of such a procedure is well-known in automata theory (*cf.*, *e.g.*, [Maz02, Exercise 2.6]); in this paper we consider a variant of the standard combinatorial game, which gives rise to an interesting complication for the corresponding labelling procedure.

The asymmetric combinatorial game on $\mathbf{G}$ (starting at s) is a variant of the combinatorial games where one player has to play into terminal nodes and the other has to keep the game alive for an infinite number of steps. Again, this game is a perfect information game with simple payoff, and thus could be analyzed via the Gale-Stewart technique with similar drawbacks.

It is the goal of this paper to give a finitary algorithm for asymmetric combinatorial games directly on $\mathbf{G}$. The algorithm we give is strongly influenced by the non-monotonic Gale-Stewart technique developed in [Löw03].

In Section 2, we define some basic graph-theoretical notions used in Section 3 where we define our games and discuss labellings and their connections to games abstractly, understanding labellings on graphs as quotients of the labellings on their associated game trees.

Finally, following the ideas of quotient labellings, in Section 4 we develop two variant algorithms for the asymmetric combinatorial games (shown in Figures 3 and 4) and discuss their running times.

2 Graphs

2.1 Graphs & Bisimulations

Our graphs $\mathbf{G} = \langle V_{\mathbf{G}}, E_{\mathbf{G}} \rangle$ are directed graphs (digraphs), *i.e.*, $V_{\mathbf{G}}$ is a set of vertices and $E \subseteq V_{\mathbf{G}} \times V_{\mathbf{G}}$ is an arbitrary binary relation. If $\equiv$ is an

[2] The statement is ATR_0, a set-theoretic existence statement for sets defined by transfinite recursion along an arithmetically defined wellorder. *Cf.* [Ste76, Tan90]; for a detailed overview in the context of Reverse Mathematics, *cf.* [Sim99, § V.8].

equivalence relation on $V_{\mathbf{G}}$, we can define a graph structure on the set of $\equiv$-equivalence classes $V_{\mathbf{G}/\equiv} := \{[v]_\equiv\,;\, v \in V_{\mathbf{G}}\}$ as follows:

$$\langle [v]_\equiv, [w]_\equiv\rangle \in E_{\mathbf{G}/\equiv} :\Longleftrightarrow \exists v', w'(v \equiv v' \;\&\; w \equiv w' \;\&\; \langle v, w\rangle \in E_{\mathbf{G}}).$$

We write $\mathbf{G}/\equiv := \langle V_{\mathbf{G}/\equiv}, E_{\mathbf{G}/\equiv}\rangle$ for the quotient graph.

If $s \in V_{\mathbf{G}}$, we call the pair $\langle \mathbf{G}, s\rangle$ a **pointed graph**. As usual, the natural numbers are identified with the sets of their predecessors, *i.e.*, $0 = \varnothing$ and $n+1 = \{0, \ldots, n\}$. If $N \in \mathbb{N} \cup \{\mathbb{N}\}$, we call a function $W\colon N \to V$ a **walk through** $\langle \mathbf{G}, s\rangle$ **of length** N if

1. for each $n+1 \in N$, we have $\langle W(n), W(n+1)\rangle \in E_{\mathbf{G}}$, and
2. $W(0) = s$.

A walk is called **finite** if $N \in \mathbb{N}$. It is called **maximal** if it is either infinite or finite of length $n+1$ where $W(n)$ is a terminal node of $\mathbf{G}$. We define the **connected component of** v **in** $\mathbf{G}$ (in symbols: $\mathbf{C}^v_{\mathbf{G}}$) to be the set of vertices w such that there is a walk W of length $n+1 \in \mathbb{N}$ through $\langle \mathbf{G}, v\rangle$ with $W(n) = w$. A pointed graph $\langle \mathbf{G}, v\rangle$ is called **connected** if $V_{\mathbf{G}} = \mathbf{C}^v_{\mathbf{G}}$.

If $\langle \mathbf{G}, s\rangle$ and $\langle \mathbf{H}, t\rangle$ are pointed graphs, then a function $Z\colon V_{\mathbf{G}} \to V_{\mathbf{H}}$ is called a **bounded epimorphism** if the following conditions hold:

1. $Z(s) = t$;
2. Z is surjective;
3. if $v_0 \in V_{\mathbf{G}}$ and $\langle v_0, v_1\rangle \in E_{\mathbf{G}}$, then $\langle Z(v_0), Z(v_1)\rangle \in E_{\mathbf{H}}$; and
4. if $w_0 \in V_{\mathbf{H}}$ and $\langle w_0, w_1\rangle \in E_{\mathbf{H}}$ and $Z(v_0) = w_0$, then there is some $v_1 \in V_{\mathbf{G}}$ such that $Z(v_1) = w_1$ and $\langle v_0, v_1\rangle \in E_{\mathbf{G}}$.

If Z is a bounded epimorphism between $\mathbf{G}$ and $\mathbf{H}$, and we can define an equivalence relation $\equiv_Z$ on $V_{\mathbf{G}}$ by

$$v \equiv_Z w :\Longleftrightarrow Z(v) = Z(w).$$

Proposition 1. Let $\langle \mathbf{G}, s\rangle$ and $\langle \mathbf{H}, t\rangle$ be pointed graphs and Z a bounded epimorphism between them. Let $\equiv_Z$ be the equivalence relation on $V_{\mathbf{G}}$ defined via Z. Then $\langle \mathbf{G}/\equiv_Z, [s]_{\equiv_Z}\rangle \cong \langle \mathbf{H}, t\rangle$.

Proof. Define $\widehat{Z}\colon V_{\mathbf{G}}/\equiv_Z \to V_{\mathbf{H}}$ by

$$\widehat{Z}([v]_{\equiv_Z}) := Z(v).$$

This function is clearly well-defined and a bijection. Using the fact that Z is a bounded epimorphism, it is easy to see that $\widehat{Z}$ is structure preserving.

q.e.d.

2.2 The unravelled tree and the alternating graph

Let $\langle \mathbf{G}, s\rangle$ be a pointed graph. Define the set $V_{\mathbf{T}^s_{\mathbf{G}}}$ to be the set of finite walks through $\langle \mathbf{G}, s\rangle$ (*i.e.*, finite sequences of nodes in $\mathbf{C}^s_{\mathbf{G}}$ connected by $E_{\mathbf{G}}$ and starting with s). For walks W_0 of length n and W_1 of length $n+1$, we let $\langle W_0, W_1\rangle \in E_{\mathbf{T}^s_{\mathbf{G}}}$ if and only if $W_0 = W_1{\restriction}n$. Furthermore, let $\mathrm{root}_{\mathbf{G},s} := \{\langle 0, s\rangle\}$ be the unique walk of length 1. Then

$$\mathbf{T}^s_{\mathbf{G}} := \langle V_{\mathbf{T}^s_{\mathbf{G}}}, E_{\mathbf{T}^s_{\mathbf{G}}}\rangle.$$

We call $\langle \mathbf{T}^s_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},s}\rangle$ the **unravelled tree of** $\langle \mathbf{G}, s\rangle$.

In the following, we will use the parity function $\mathrm{par}\colon \mathbb{N} \to 2$ assigning to each natural number its parity. Let $V_{\mathbf{A}_{\mathbf{G}}} := 2 \times V_{\mathbf{G}}$,

$$\langle\langle e, v\rangle, \langle 1-e, w\rangle\rangle \in E_{\mathbf{A}_{\mathbf{G}}} :\Longleftrightarrow \langle v, w\rangle \in E_{\mathbf{G}},$$

and call $\mathbf{A}_{\mathbf{G}} := \langle V_{\mathbf{A}_{\mathbf{G}}}, E_{\mathbf{A}_{\mathbf{G}}}\rangle$ the **alternating graph of G**. If $s \in V_{\mathbf{G}}$, then we let $\mathbf{A}^s_{\mathbf{G}}$ be the connected component of $\langle 0, s\rangle$ in $\mathbf{A}_{\mathbf{G}}$.

Proposition 2. If $\langle \mathbf{G}, s\rangle$ is a pointed graph, there are bounded epimorphisms from $\langle \mathbf{T}^s_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},s}\rangle$ to $\langle \mathbf{G}, s\rangle$, from $\langle \mathbf{A}^s_{\mathbf{G}}, \langle 0, s\rangle\rangle$ to $\langle \mathbf{G}, s\rangle$ and from $\langle \mathbf{T}^s_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},s}\rangle$ to $\langle \mathbf{A}^s_{\mathbf{G}}, \langle 0, s\rangle\rangle$.

Proof. Let $e \in 2$, $v \in V_{\mathbf{G}}$, and $\mathrm{dom}(W) = n+1$ with $W(n) = v$. Then define

$$\begin{aligned} Z_{\mathbf{T}}(W) &:= v, \\ Z_{\mathbf{A}}(\langle e, v\rangle) &:= v, \text{ and} \\ Z_{\mathbf{T},\mathbf{A}}(W) &:= \langle \mathrm{par}(n), v\rangle. \end{aligned}$$

The functions $Z_{\mathbf{T}}$, $Z_{\mathbf{A}}$ and $Z_{\mathbf{T},\mathbf{A}}$ are bounded epimorphisms. q.e.d.

If $\langle \mathbf{G}, s\rangle$ is a pointed graph, $v \in V_{\mathbf{G}}$ and W is a walk through $\langle \mathbf{G}, s\rangle$ of length $n+1$ such that $W(n) = v$, then the connected component of W in $\mathbf{T}^s_{\mathbf{G}}$ and the graph $\mathbf{T}^v_{\mathbf{G}}$ are isomorphic as graphs. As a consequence, we get a slightly more general version of Proposition 2:

Proposition 3. Let $\langle \mathbf{G}, s\rangle$ be a pointed graph, and W be a walk through $\langle \mathbf{G}, s\rangle$ of length $n+1$ such that $W(n) = v$. Then there are bounded epimorphisms from $\langle \mathbf{T}^s_{\mathbf{G}}, W\rangle$ to $\langle \mathbf{G}, v\rangle$ and from $\langle \mathbf{T}^s_{\mathbf{G}}, W\rangle$ to $\langle \mathbf{A}^v_{\mathbf{G}}, \langle 0, v\rangle\rangle$.

Proof. Compose the bounded epimorphisms $Z_{\mathbf{T}}$ between $\mathbf{T}^v_{\mathbf{G}}$ and $\mathbf{G}$ and $Z_{\mathbf{T},\mathbf{A}}$ between $\mathbf{T}^v_{\mathbf{G}}$ and $\mathbf{A}^v_{\mathbf{G}}$ with the mentioned graph isomorphism. q.e.d.

3 Games

3.1 Games and game equivalences

Given a graph $\mathbf{G} = \langle V_{\mathbf{G}}, E_{\mathbf{G}} \rangle$ and $s \in V_{\mathbf{G}}$, we define the **(symmetric) combinatorial game on** $\langle \mathbf{G}, s \rangle$ (in symbols: $\mathsf{S}(\mathbf{G}, s)$): at the beginning of the game, a token is positioned in the vertex s; players 0 and 1 move in turn with player 0 starting by pushing the token along the edges of $\mathbf{G}$; the player making the last move wins the game. If the game goes on for infinitely many steps, the outcome of the game is a draw. We define the **inverted symmetric combinatorial game** $\overline{\mathsf{S}}(\mathbf{G}, s)$ to be the game played like the symmetric combinatorial game, just with the rôles of the two players interchanged, *i.e.*, player 1 starts.

In the asymmetric version, the rôles of player 1 and the draw are interchanged: Given a graph $\mathbf{G} = \langle V_{\mathbf{G}}, E_{\mathbf{G}} \rangle$ and $s \in V_{\mathbf{G}}$, we define the **asymmetric combinatorial game on** $\langle \mathbf{G}, s \rangle$ (in symbols: $\mathsf{A}(\mathbf{G}, s)$): at the beginning of the game, a token is positioned in the vertex s; players 0 and 1 move in turn with player 0 starting by pushing the token along the edges of $\mathbf{G}$; if player 0 pushes the token into a terminal node, he wins; if player 1 pushes the token into a terminal node, the game is a draw. If the game goes on for infinitely many steps, player 1 wins. Again, we define the **inverted asymmetric game** $\overline{\mathsf{A}}(\mathbf{G}, s)$ to be the game with the players interchanged.

Strategies in these combinatorial games are simply functions that tell the players which edge $\langle v_0, v_1 \rangle$ to use if they are presented with the token in vertex v_0 (it is obvious that such memory-free strategies suffice for these games). A strategy is **winning** if the player following the strategy wins the game regardless of how the other player plays, and a strategy is called **nonlosing** if the game in which one player follows the strategy results in either a win for that player or a draw.

By the determinacy theorem of Gale and Stewart (for details, *cf.* Section 3.2) each of the games $\mathsf{X} \in \{\mathsf{A}, \overline{\mathsf{A}}, \mathsf{S}, \overline{\mathsf{S}}\}$ defined above will have one of the following three **values**, denoted by $\mathrm{val}(\mathsf{X})$:

- W. Player 0 has a winning strategy,
- D. both players have a nonlosing strategy,
- L. Player 1 has a winning strategy.

On the set $L = \{\mathsf{L}, \mathsf{D}, \mathsf{W}\}$ of these values, we define a lattice structure by

$\mathsf{L} \leq \mathsf{D} \leq \mathsf{W}$ and an inversion function $\mathrm{inv}\colon L \to L$ defined by

$$\begin{array}{rcl} \mathsf{W} & \mapsto & \mathsf{L} \\ \mathsf{D} & \mapsto & \mathsf{D} \\ \mathsf{L} & \mapsto & \mathsf{W}. \end{array}$$

Let $\mathsf{X}, \mathsf{Y} \in \{\mathsf{A}, \overline{\mathsf{A}}, \mathsf{S}, \overline{\mathsf{S}}\}$. We say that X and Y are **equivalent** if they have the same value. We say that they are **anti-equivalent** if $\mathrm{val}(\mathsf{X}) = \mathrm{inv}(\mathrm{val}(\mathsf{Y}))$.

We define the notion of a X-Y-(anti)-equivalence: Let $\mathbf{G}$ and $\mathbf{H}$ be graphs and let $f\colon V_{\mathbf{G}} \to V_{\mathbf{H}}$ be a function. Then f is called a **X-Y-(anti)-equivalence** if for all $v \in V_{\mathbf{G}}$, we have that $\mathsf{X}(\mathbf{G}, v)$ and $\mathsf{Y}(\mathbf{H}, f(v))$ are (anti)-equivalent.

There are some obvious facts about equivalence of combinatorial games:

Proposition 4. For every pointed graph $\langle \mathbf{G}, v\rangle$, the games $\mathsf{S}(\mathbf{G}, v)$ and $\overline{\mathsf{S}}(\mathbf{G}, v)$ are anti-equivalent. In other words, $\mathrm{id}\colon V_{\mathbf{G}} \to V_{\mathbf{G}}$ is an S-$\overline{\mathsf{S}}$-anti-equivalence.

Proof. Obvious. q.e.d.

Proposition 5. Let $\mathbf{G}$ and $\mathbf{H}$ be graphs, let $\mathsf{X} \in \{\mathsf{S}, \overline{\mathsf{S}}, \mathsf{A}, \overline{\mathsf{A}}\}$, and let $F\colon V_{\mathbf{G}} \to V_{\mathbf{H}}$ be a function. If F is a bounded epimorphism, then it is a X-X-equivalence.

Proof. Obvious. q.e.d.

An immediate consequence of Propositions 2, 3 and 5 is that in order to analyze arbitrary combinatorial games, it is enough to analyze games on trees:

Corollary 6. Let $\mathsf{X} \in \{\mathsf{S}, \overline{\mathsf{S}}, \mathsf{A}, \overline{\mathsf{A}}\}$ and let $\langle \mathbf{G}, s\rangle$ be a pointed graph. Then the games $\mathsf{X}(\mathbf{G}, s)$ and $\mathsf{X}(\mathbf{T}^s_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},s})$ are equivalent. Also, for any walk W through $\langle \mathbf{G}, s\rangle$ with length $n+1$ and $W(n) = v$, the games $\mathsf{X}(\mathbf{G}, v)$ and $\mathsf{X}(\mathbf{T}^s_{\mathbf{G}}, W)$ are equivalent.

3.2 A translation into the usual Gale-Stewart theory of infinite games

Corollary 6 is the underlying methodology of the Gale-Stewart theory [GalSte53]. Instead of looking at the (possibly cyclic) graph, we look at the unravelled tree and analyze the game on the tree via backwards induction with a (possibly transfinite) labelling construction.

We shall translate our tree games into the usual topological notation of Gale-Stewart theory: Look at the space ${V_{\mathbf{G}}}^{\mathbb{N}}$ of functions from $\mathbb{N}$ into $V_{\mathbf{G}}$, endowed with the product topology of the discrete topology on $V_{\mathbf{G}}$.

We define three infinite games $\mathsf{G}_0(\mathbf{G}, v)$, $\mathsf{G}_1(\mathbf{G}, v)$, and $\mathsf{H}_1(\mathbf{G}, v)$. In all of the games, players 0 and 1 play elements of $V_{\mathbf{G}}$ in turn and produce an element of ${V_{\mathbf{G}}}^{\mathbb{N}}$, let's call it X. We assume that $X(0) = v$ and that player 0 plays the odd digits and player 1 plays the even digits. The payoff sets of the games are defined as follows:

- In $\mathsf{G}_0(\mathbf{G}, v)$, player 0 wins if the least $n+1$ such that $X{\restriction}n+1$ is not a walk through $\langle \mathbf{G}, v\rangle$ exists and is odd. Otherwise, player 1 wins.
- In $\mathsf{G}_1(\mathbf{G}, v)$, player 0 wins if either X is an infinite walk through $\langle \mathbf{G}, v\rangle$, or the least $n+1$ such that $X{\restriction}n+1$ is not a walk through $\langle \mathbf{G}, v\rangle$ is odd. Otherwise, player 1 wins.
- In $\mathsf{H}_1(\mathbf{G}, v)$, player 0 wins if there is a least $n+1$ such that $X{\restriction}n+1$ is not a walk through $\langle \mathbf{G}, v\rangle$ and either $n+1$ is odd or $X(n)$ is a terminal node of $\mathbf{G}$. Otherwise, player 1 wins.

The payoff sets for player 0 in the defined three games are either open (G_0 and H_1) or closed (G_1) in the topology defined on ${V_{\mathbf{G}}}^{\mathbb{N}}$, and by the usual Gale-Stewart theorem for open and closed games without draw, one of the two players has a winning strategy, *i.e.*, the values are either W or L.

It is easy to see that these infinite Gale-Stewart games correspond to the combinatorial games as follows:

Proposition 7. For every pointed graph $\langle \mathbf{G}, v\rangle$, the following equivalences hold:

$$
\begin{array}{rcl}
\mathrm{val}(\mathsf{G}_0(\mathbf{G}, v)) = \mathsf{W} & \iff & \mathrm{val}(\mathsf{S}(\mathbf{T}^v_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},v})) = \mathsf{W} \\
 & \iff & \mathrm{val}(\mathsf{A}(\mathbf{T}^v_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},v})) = \mathsf{W} \\
\mathrm{val}(\mathsf{G}_0(\mathbf{G}, v)) = \mathsf{L} & \iff & \text{player 1 has a nonlosing strategy for } \mathsf{S}(\mathbf{T}^v_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},v}) \\
 & \iff & \text{player 1 has a nonlosing strategy for } \mathsf{A}(\mathbf{T}^v_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},v}) \\
\mathrm{val}(\mathsf{G}_1(\mathbf{G}, v)) = \mathsf{W} & \iff & \text{player 0 has a nonlosing strategy for } \mathsf{S}(\mathbf{T}^v_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},v}) \\
\mathrm{val}(\mathsf{G}_1(\mathbf{G}, v)) = \mathsf{L} & \iff & \mathrm{val}(\mathsf{S}(\mathbf{T}^v_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},v})) = \mathsf{L} \\
\mathrm{val}(\mathsf{H}_1(\mathbf{G}, v)) = \mathsf{W} & \iff & \text{player 0 has a nonlosing strategy for } \mathsf{A}(\mathbf{T}^v_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},v}) \\
\mathrm{val}(\mathsf{H}_1(\mathbf{G}, v)) = \mathsf{L} & \iff & \mathrm{val}(\mathsf{A}(\mathbf{T}^v_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},v})) = \mathsf{L}
\end{array}
$$

As a consequence, we get a proof of the claim (see above) that the values W, L and D are the only possible values for our games.

3.3 Sound labellings

Let $L := \{\mathsf{L}, \mathsf{D}, \mathsf{W}\}$, $\mathbf{G}$ be a graph and $\mathsf{X} \in \{\mathsf{S}, \overline{\mathsf{S}}, \mathsf{A}, \overline{\mathsf{A}}\}$. An L-labelling $\ell\colon V_{\mathbf{G}} \to L$ is called **X-sound** if it is a total function and if for each vertex $v \in V_{\mathbf{G}}$, we have

$$\ell(v) = \mathrm{val}(\mathsf{X}(\mathbf{G}, v)).$$

Because of Proposition 4, the notions of S-soundness and $\overline{\mathsf{S}}$-soundness are closely connected:

Corollary 8. Let $\mathbf{G}$ be a graph and $\ell\colon V_{\mathbf{G}} \to L$ be S-sound. Then $\overline{\ell}$ defined by $\overline{\ell}(v) := \mathrm{inv}(\ell(v))$ is $\overline{\mathsf{S}}$-sound.

The Gale-Stewart analysis for games on trees gives a (possibly transfinite) recursive procedure to actually compute an S-sound labelling:

Theorem 9 (Gale & Stewart; 1953). If $\mathbf{T}$ is a tree, then there is recursive procedure that computes (in less than $\mathrm{Card}(V_{\mathbf{T}})^+$ steps) the S-sound labelling ℓ.

Proposition 10. Let $\langle \mathbf{G}, s\rangle$ be a connected pointed graph. If ℓ is the S-sound labelling on $\mathbf{T}^s_{\mathbf{G}}$, then the quotient labelling $\ell/{\equiv_{Z_{\mathbf{T}}}}$ on $\mathbf{G}$ defined by

$$\ell/{\equiv_{Z_{\mathbf{T}}}}(v) := \ell(W)$$

(where W is any walk through $\langle \mathbf{G}, s\rangle$ with length $n+1$ such that $W(n) = v$) is well-defined and is the S-sound labelling for $\mathbf{G}$.

Proof. Let us show that $\equiv_{Z_{\mathbf{T}}}$ respects $\ell_{\mathbf{T}}$:

Suppose $W \equiv_{Z_{\mathbf{T}}} W'$, *i.e.*, $Z_{\mathbf{T}}(W) = Z_{\mathbf{T}}(W') = v$ for some v. Corollary 6 tells us that $\mathsf{S}(\mathbf{T}^s_{\mathbf{G}}, W)$ and $\mathsf{S}(\mathbf{T}^s_{\mathbf{G}}, W')$ are both equivalent to $\mathsf{S}(\mathbf{G}, v)$, so in particular, $\ell(W) = \ell(W')$, and the quotient labelling is sound. q.e.d.

For asymmetric combinatorial games, we don't have the symmetry of Corollary 8:

Proposition 11. If $\mathrm{val}(\mathsf{A}(\mathbf{G}, v)) = \mathsf{W}$, then $\mathrm{val}(\overline{\mathsf{A}}(\mathbf{G}, v)) \neq \mathsf{L}$. All other combinations are possible.

Proof. For player 0, a winning strategy is a strategy that forces the token into a terminal node in an odd number of moves. Such a strategy is a non-losing strategy for player 1 in the inverted game.

In Figure 1, examples for all eight combinatorially possible situations are given. q.e.d.

Because of the asymmetry indicated by Proposition 11, we define the following notion: A (partial) function $\ell\colon V_{\mathbf{G}} \to L^2$ is called a **(partial)**

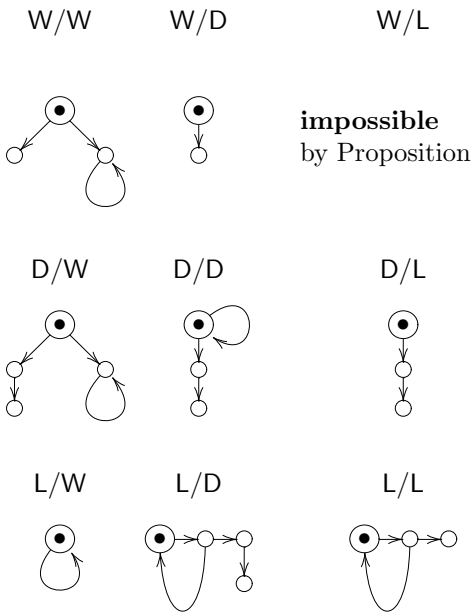

Figure 1. Examples for the eight possible value combinations in the asymmetric and the inverted asymmetric combinatorial game.

L-bilabelling on $\mathbf{G}$. We write $\ell(v) = \langle \ell_0(v), \ell_1(v) \rangle$. An L-bilabelling $\ell\colon V_{\mathbf{G}} \to L^2$ is called **A-sound** if it's total and for each vertex $v \in V$, we have

$$\ell_0(v) = \mathrm{val}(\mathsf{A}(\mathbf{G}, v)) \text{ and } \ell_1(v) = \mathrm{val}(\overline{\mathsf{A}}(\mathbf{G}, v)).$$

In analogy to Theorem 9, the Gale-Stewart analysis gives us the existence of A-sound labellings for trees $\mathbf{T}$. Among several ways to produce such a labelling, there is one procedure that was the motivation for the algorithm described in Section 4: in [Löw03], the author gives a non-monotone variant of the Gale-Stewart procedure which can be used to construct the A-sound labelling for trees.

Proposition 12. Let $\langle \mathbf{G}, s \rangle$ be a connected pointed graph. If ℓ is the A-sound labelling on $\mathbf{T}^s_{\mathbf{G}}$, then the quotient labelling $\ell/{\equiv_{Z_{\mathbf{T},\mathbf{A}}}}$ on $\mathbf{A}^s_{\mathbf{G}}$ defined by

$$\ell/{\equiv_{Z_{\mathbf{T},\mathbf{A}}}}(\langle e, v \rangle) := \ell(W)$$

(where W is any walk through $\langle \mathbf{G}, s \rangle$ with length $2n + e + 1$ such that $W(2n + e) = v$) is well-defined and is the A-sound labelling for $\mathbf{A}^s_{\mathbf{G}}$.

We would like to extend this to an A-sound bilabelling, but not all nodes

in $\mathbf{G}$ are necessarily reachable by both players. The following simple construction helps:

If $\langle \mathbf{G}, s\rangle$ is a connected pointed graph, let $\mathbf{G}^*_s$ be defined by

$$\begin{aligned} V_{\mathbf{G}^*_s} &:= V_{\mathbf{G}} \cup \{x\}, \\ E_{\mathbf{G}^*_s} &:= E_{\mathbf{G}} \cup \{\langle x, s\rangle\} \end{aligned}$$

(where $x \notin V_{\mathbf{G}}$). The connectedness of $\langle \mathbf{G}, s\rangle$ implies that

$$\mathbf{C}^{\langle 0,s\rangle}_{\mathbf{A}_{\mathbf{G}}} \cup \mathbf{C}^{\langle 0,x\rangle}_{\mathbf{A}_{\mathbf{G}^*_s}} = 2 \times V_{\mathbf{G}};$$

moreover, if W is a walk through $\langle \mathbf{T}^s_{\mathbf{G}}, \mathrm{root}_{\mathbf{G},s}\rangle$ of length $n+1$ and W' is a walk through $\langle \mathbf{T}^x_{\mathbf{G}^*_s}, \mathrm{root}_{\mathbf{G}^*_s,x}\rangle$ of length $m+1$ with $W(n) = W'(m)$ and $\mathrm{par}(n) = \mathrm{par}(m)$, then they represent exactly the same position in the game on $\mathbf{G}$ (albeit with different game histories), so if ℓ is A-sound on $\mathbf{T}^s_{\mathbf{G}}$ and ℓ^* is A-sound on $\mathbf{T}^x_{\mathbf{G}^*_s}$, then $\ell(W) = \ell^*(W')$. As a consequence, we get:

Proposition 13. If $\langle \mathbf{G}, s\rangle$ is a connected pointed graph, ℓ is an A-sound labelling on $\mathbf{T}^s_{\mathbf{G}}$ and ℓ^* is an A-sound labelling on $\mathbf{T}^x_{\mathbf{G}^*_s}$, then the bilabelling $\ell^\dagger$ defined by

$$\ell^\dagger_e(v) := \begin{cases} \ell/{\equiv_{Z_{\mathbf{T},\mathbf{A}}}}(\langle e, v\rangle) & \text{if } \langle e, v\rangle \in \mathbf{C}^{\langle 0,s\rangle}_{\mathbf{A}_{\mathbf{G}}}, \text{ or} \\ \ell^*/{\equiv_{Z_{\mathbf{T},\mathbf{A}}}}(\langle e, v\rangle) & \text{otherwise} \end{cases}$$

is welldefined, total and an A-sound bilabelling on $\mathbf{G}$.

In the following section, we shall give an algorithm to compute $\ell^\dagger$ directly without going through the tree unravelling.

4 The algorithm

We fix a graph $\mathbf{G}$. For the purpose of this section, we assume that $V_{\mathbf{G}}$ is finite. Using the lattice structure on L, we can define an ordering $\leq^*$ on L^2 as the product ordering of $\langle L, \overline{\leq}\rangle$ with $\langle L, \leq\rangle$ as depicted in Figure 2.[3]

As mentioned, a labelling procedure for the graph (instead of the unravelled tree) is a "folk result" in automata theory (see for instance [Fra97, p. 15], albeit in a more graph-theoretic context). We call this procedure **Backward Induction (Graph)** to distinguish it from the Gale-Stewart tree analysis.

We let $\mathrm{BIG}^0(v) := \mathsf{L}$ for all terminal nodes v. After that, we define

[3] Here $\overline{\leq}$ denotes the inverse ordering of $\leq$.

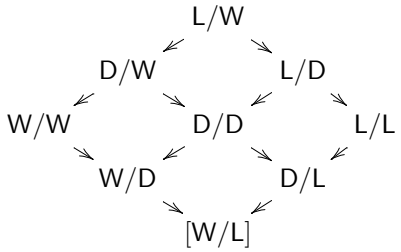

Figure 2. The ordering $\leq^*$ on L^2.

$$\mathrm{BIG}^{n+1}(v) := \begin{cases} \mathrm{BIG}^n(v) & \text{if } v \in \mathrm{dom}(\mathrm{BIG}^n), \\ \mathsf{W} & \text{if there is } \langle v, w\rangle \in E \text{ and } \mathrm{BIG}^n(w) = \mathsf{L}, \text{ or} \\ \mathsf{L} & \text{if for all } \langle v, w\rangle \in E, \text{ we have } \mathrm{BIG}^n(w) = \mathsf{W}. \end{cases}$$

For some N, we have $\mathrm{BIG}^N = \mathrm{BIG}^{N+1}$, then we let

$$\mathrm{BIG}(v) := \begin{cases} \mathrm{BIG}^N(v) & \text{if } v \in \mathrm{dom}(\mathrm{BIG}^N), \text{ or} \\ \mathsf{D} & \text{otherwise.} \end{cases}$$

This algorithm produces an S-sound labelling in $O(|V_{\mathbf{G}}| + |E_{\mathbf{G}}|)$ steps, and is essentially the quotient labelling of the Gale-Stewart labelling on $\mathbf{T}_{\mathbf{G}}$: we label a vertex $v \in V_{\mathbf{G}}$ as soon as some W with $Z_{\mathbf{T}}(W) = v$ is labelled in the Gale-Stewart construction. As the Gale-Stewart procedure, this labelling is monotonic in the sense that whenever a vertex is labelled, it will retain that label for ever.

Following this idea and injecting non-monotonicity in the spirit of [Löw03, § 5] into the procedure, we shall now give an algorithm `NMBIG` (for "non-monotonic backward induction (graph)") that produces the A-sound bilabelling.

We give the algorithm in pseudocode in Figure 3. Let us explain the two special datatypes `label` and `graph` used in the pseudocode:

Variables of type `label` can take the values `W`, `D`, and `L` representing W, D, and L. We have a binary relation $<$ defined for variables of type `label`, and `X` $<$ `Y` is `TRUE` if and only if $\mathsf{X} < \mathsf{Y}$. In addition, there is a unary function `INV` defined on `label` corresponding to the inversion function inv.

The datatype `graph` encodes the bilabelled graph structure. Let $\langle \mathbf{G}, \ell\rangle$ be a bilabelled graph with $V_{\mathbf{G}} = \{v_i\,;\, 0 \leq i \leq N\}$. If `G` is a variable of type `graph` representing $\langle \mathbf{G}, \ell\rangle$, then the following objects are defined:

```
  Procedure:MAIN(G:graph)
  for i ← 1 to Nvert[G] do
3 |  Ell[G,i,0] ← L;
4 |  Ell[G,i,1] ← W;
  for i ← 1 to Nvert[G] do
  |  if Outbound[G,i] == ∅ then
7 |  |  Ell[G,i,0] ← D;
8 |  |  Ell[G,i,1] ← L;
9 |  |  foreach j ∈ Inbound[G,i] do
  |  |  |  Label(G,j,1);
  |  |  |  Label(G,j,0);
  |  |  
  |  

  Procedure:Label(G:graph;i:integer;e:binary)
13 aux ← L;
14 foreach j ∈ Outbound[G,i] do
15 |  aux ← aux + INV[Ell[G,j,1-e]];
   |  if Ell[G,i,e] ≠ aux then
17 |  |  Ell[G,i,e] ← aux;
18 |  |  foreach j ∈ Inbound[G,i] do
   |  |  |  Label(G,j,1-e);
   |  |  
   |  
```

Figure 3. The algorithm NMBIG for non-monotonic backward induction graph labelling.

- Nvert[G], the number of vertices of the graph, *i.e.*, $N + 1$;
- for each $i \leq N$, the objects Outbound[G, i] and Inbound[G, i] representing the sets $\{v_j \,;\, \langle v_i, v_j\rangle \in E_{\mathbf{G}}\}$ and $\{v_j \,;\, \langle v_j, v_i\rangle \in E_{\mathbf{G}}\}$, respectively;
- for each $i \leq N$ and $e \in 2$, an object Ell[G, i, e] of type label, representing $\ell_e(v_i)$.

If the graph structure is stored using an adjacency matrix, then iterating over the outbound or inbound set for a particular vertex can be done in linear time with respect to the size of the vertex set.

We run the algorithm NMBIG on (a representation of) $\mathbf{G}$. For any $t \in \mathbb{N}$, we let $\ell_e^t(v_i)$ be the value of Ell[G, i, e] at time t of the algorithm,[4] and

$$\ell^t(v_i) := \langle \ell_0^t(v_i), \ell_1^t(v_i)\rangle.$$

[4]Code lines 3 and 4 make sure that the values $\ell_e(t)(v_i)$ are defined early on in the algorithm (in step $e \cdot N + i$), so from step $2 \cdot N$ onwards, ℓ^t is a total function. In the following, we shall mostly ignore these first $2 \cdot N$ steps of the algorithm.

Proposition 14. For each $i \leq N$, the sequence $\langle \ell^t(v_i)\,;\, t \in \mathbb{N}\rangle$ is $\leq^*$-decreasing in L^2. Equivalently, the sequence $\langle \ell_0^t(v_i)\,;\, t \in \mathbb{N}\rangle$ is $\overline{\leq}$-decreasing and the sequence $\langle \ell_1^t(v_i)\,;\, t \in \mathbb{N}\rangle$ is $\leq$-decreasing.

Proof. Let $t+1$ be the least number such that $\ell^{t+1}(v_i) <^* \ell^t(v_i)$ for some i. Clearly, the value of the bilabeling can only be changed by code lines 7, 8 and 17 of the algorithm.

Since the procedure `MAIN` can only change the values at terminal nodes from L/W to D/W and then to D/L, the lines 7 and 8 cannot create a decrease in $\leq^*$. Note that this means that v_i is not a terminal node since `Label(G, i, e)` is only called if there is an edge from v_i to somewhere.

Also, we cannot be in the first call of `Label(G, i, e)` at time t since the bilabelling is initialized with L/W which is the top element of $\langle L^2, \leq^*\rangle$. Consequently, there are some $s_0 < s_1 < t$ such that at both s_0 and s_1, the procedure `Label(G, i, e)` is called. Let s_0 be largest with that property. By our assumption, we have that $\ell^{t+1}(v_i) <^* \ell^t(v_i) = \ell^{s_1}(v_i)$. By code lines 13 to 15, we have

$$\ell_e^{t+1}(v_i) := \sup_{\leq}\{\mathrm{inv}(\ell_{1-e}^{s_1}(w))\,;\, \langle v_i, w\rangle \in E_{\mathbf{G}}\}, \text{ and}$$

$$\ell_e^{s_1}(v_i) := \sup_{\leq}\{\mathrm{inv}(\ell_{1-e}^{s_0}(w))\,;\, \langle v_i, w\rangle \in E_{\mathbf{G}}\}.$$

But this means that there is some w such that $\ell^{s_0}(w) <^* \ell^{s_1}(w)$, contradicting the choice of $t+1$ as minimal. q.e.d.

Proposition 15. The procedure `Label` is called at most $4 \cdot |E_{\mathbf{G}}|$ times.

Proof. By the loops 9 and 18, each call of `Label`(G, i, e) is associated with an edge $\langle v_i, w\rangle$, and by code lines 7, 8 and 17, preceded by a change of $\ell(w)$. By Proposition 14, this means that each such an edge can be used for calls of the procedure `Label` at most four times. q.e.d.

Theorem 16. The running time of `NMBIG` is $O(|V_{\mathbf{G}}| + |E_{\mathbf{G}}|^2)$. If $\mathbf{G}$ is connected, this is $O(|E_{\mathbf{G}}|^2)$.

Proof. The procedure `NMBIG` itself (without the recursive calls of `Label`) takes at most $4 \cdot |V_{\mathbf{G}}| + 2 \cdot |E_{\mathbf{G}}|$ steps. Each call of `Label` has running time $O(|E_{\mathbf{G}}|)$, so by Proposition 15, the entire running time is $O(|V_{\mathbf{G}}| + |E_{\mathbf{G}}|^2)$. q.e.d.

The critical lines in the algorithm that push the running time from linear to quadratic are the loop from line 14: every time we run `Label` for v_i, we have to check the current values of all its successors. Let $\mathbf{G}_n$ be the graph with a root v_0 and n immediate successors of the root which are terminal nodes, *i.e.*, $|E_{\mathbf{G}_n}| = n$. Then $\mathtt{Label}(\mathtt{G}, \mathtt{1}, \mathtt{e})$ is called n times (once for each terminal node) and each time its running time is at least n because it has to check each of the terminal nodes, so the total running time is at least $|E_{\mathbf{G}_n}|^2$.

The running time can be pushed to linear in the size of the edge set, at the cost of making the algorithm significantly less readable. In Section 4.1 we give the refined algorithm, and show that it computes the same labelling as `NMBIG`. The proof of the following theorem is however less tedious on the simpler algorithm:

Theorem 17. If run on the graph $\mathbf{G}$, the algorithm `NMBIG` computes the A-sound bilabelling on $\mathbf{G}$.

Proof. Again, let $\ell_e^t(v_i)$ be the value of $\mathtt{Ell}[\mathtt{G}, \mathtt{i}, \mathtt{e}]$ at time t. By Theorem 16, these values stabilize at some finite time N. Let $\ell_e(v_i) := \ell_e^N(v_i)$ be the eventual value.

This proof follows essentially the idea of the Gale-Stewart proof (a.k.a. "backwards induction"). The main ingredient of that idea is that players have strategies that force the following to be true: if W is a run of the game according to the strategy, then the sequence of time indices of the label assignments of $W(n)$ during the algorithm is a decreasing sequence of integers. By wellfoundedness of $\mathbb{N}$, it can be deduced that we hit one of the basic cases eventually. Unfortunately, in our case, the nonmonotonicity of the algorithm causes some problems: it is possible that vertex v is labelled at time t, but some successors receive their label later. In order to deal with this, we have to go through the different cases in detail.

For any $v \in V_{\mathbf{G}}$, let

$$\mathrm{ind}_e(v) := \min\{t\,;\, \ell_e^t(v) = \ell_e(v)\}$$

be the e-**index** of v_i. (Note that by Proposition 14, for all $\mathrm{ind}_e(v) \leq t \leq N$, we have that $\ell_e^t(v) = \ell_e(t)$.)

We shall discuss the properties of the six possible cases:

Case 1. If $\ell_0(v_i) = \mathsf{W}$, then there is some w with $\langle v_i, w\rangle \in E_{\mathbf{G}}$, $\ell_1(w) = \mathsf{L}$, and $\mathrm{ind}_1(w) < \mathrm{ind}_0(v_i)$.

[The vertex v_i has been labelled by code line 15, and this means that in the preceding call of code line 15 some successor was 1-labelled L. By Proposition 14, a 1-label L can never be changed anymore.]

Case 2. If $\ell_0(v_i) = \mathsf{D}$, then there is no w with $\langle v_i, w\rangle \in E_{\mathbf{G}}$ and $\ell_1(w) = \mathsf{L}$. Also, either v_i is terminal or there is some w with $\langle v_i, w\rangle \in E_{\mathbf{G}}$ and $\ell_1(w) = \mathsf{D}$, and for all such w, $\mathrm{ind}_1(w) < \mathrm{ind}_0(v_i)$.

[If v_i is terminal, the claim is trivial, so let v_i be nonterminal. Let $t := \mathrm{ind}_0(v_i)$ which is the time of a call of code line 15. Therefore, at time t, we have

$$(\star_t) \quad \begin{array}{l} \forall w\,(\langle v_i, w\rangle \in E_{\mathbf{G}} \rightarrow \ell_1^t(w) \neq \mathsf{L}) \quad \& \\ \exists w\,(\langle v_i, w\rangle \in E_{\mathbf{G}} \;\&\; \ell_1^t(w) = \mathsf{D}). \end{array}$$

By Proposition 14, none of the vertices that are 1-labelled L can change their labelling anymore and the vertices 1-labelled D can only change their label to L. Suppose that there is some $t < s \leq N$ such that $(\star_s)$ is not true anymore. Then at time s, we have a call of code line 15 and all successors of v_i are 1-labelled L. In the subsequent call of $\mathtt{Label}[\mathtt{G}, \mathtt{i}, \mathtt{0}]$, the label of v_i will be changed to W in contradiction to the assumption.]

Case 3. If $\ell_0(v_i) = \mathsf{L}$, then for all w with $\langle v_i, w\rangle \in E_{\mathbf{G}}$, we have $\ell_1(w) = \mathsf{W}$. Moreover, both v_i and all of its successors have received that label at the beginning of the algorithm (code lines 3 and 4).

[Obvious from Proposition 14 and code lines 15 and 17.]

Case 4. If $\ell_1(v_i) = \mathsf{W}$, then there is some w with $\langle v_i, w\rangle \in E_{\mathbf{G}}$ such that $\ell_0(w) = \mathsf{L}$ and both v_i and w have been labelled at the beginning of the algorithm (code lines 3 and 4).

[This is dual to Case 3.]

Case 5. If $\ell_1(v_i) = \mathsf{D}$, then there is no w with $\langle v_i, w\rangle \in E_{\mathbf{G}}$ and $\ell_0(w) = \mathsf{L}$. Also, there is some w with $\langle v_i, w\rangle \in E_{\mathbf{G}}$ and $\ell_0(w) = \mathsf{D}$, and for all such w, $\mathrm{ind}_0(w) < \mathrm{ind}_1(v_i)$.

[This is dual to Case 2, except that terminal nodes cannot be 1-labelled D.]

Case 6. If $\ell_1(v_i) = \mathsf{L}$, then for all w with $\langle v_i, w\rangle \in E_{\mathbf{G}}$, we have $\ell_0(w) = \mathsf{W}$ and $\mathrm{ind}_0(w) < \mathrm{ind}_1(v_i)$.

[This is dual to Case 1.]

With our six cases in mind, we can now define strategies for player $e \in 2$ as follows:
The strategy σ_e plays from v into some successor w such that

$$\ell_{1-e}(w) = \mathrm{inv}(\ell_e(v)),$$

and –whenever possible– such that

$$\mathrm{ind}_{1-e}(w) < \mathrm{ind}_e(v).$$

We shall show that σ_0 and σ_1 are witnesses for value $\ell_0(v)$ in $\mathsf{A}(\mathbf{G}, v)$ and value $\ell_1(v)$ in $\overline{\mathsf{A}}(\mathbf{G}, v)$, respectively. This will finish the proof of Theorem 17.

Case A: $\ell_0(v) = \mathsf{W}$.

Let W be a maximal walk through $\langle \mathbf{G}, v\rangle$ where player 0 follows σ_0 (*i.e.*, $W(2n+2) = \sigma_0(W(2n+1))$). By Cases 1 and 6, we have $\ell_0(W(2n)) = \mathsf{W}$ and $\ell_1(W(2n+1)) = \mathsf{L}$. If W is finite (say, of length $n+1$), then $W(n)$ is terminal, so $\ell(W(n)) = \mathsf{D}/\mathsf{L}$, hence n is odd and player 0 has won the game with run W.
If W is infinite, then define

$$i_k := \mathrm{ind}_{\mathrm{par}(k)}(W(k)). \qquad (\dagger)$$

By definition of σ_0 and by Cases 1 and 6, this is a strictly decreasing sequence of natural numbers which is absurd.

Case B: $\ell_1(v) = \mathsf{W}$.

Let W be a maximal walk through $\langle \mathbf{G}, v\rangle$ where player 1 follows σ_1. By Cases 3 and 4, we have $\ell_0(W(2n)) = \mathsf{L}$ and $\ell_1(W(2n+1)) = \mathsf{W}$. This implies that none of the vertices in W can be terminal, and thus W is infinite and player 1 wins the game with run W.

Case C: $\ell_0(v) = \mathsf{D}$.

Let W be a maximal walk through $\langle \mathbf{G}, v\rangle$ where player 0 follows σ_0. By Cases 2 and 5, the following three subcases cover all possibilities:

Subcase C1. There is some n such that $\ell_0(W(2n)) = \mathsf{W}$. By Case A, player 0 wins.

Subcase C2. For all k, $\ell_{\mathrm{par}(k)}(W(k)) = \mathsf{D}$ and W is finite (say, of length $n+1$). Then $W(n)$ is a terminal node, and since $\ell_{\mathrm{par}(n)}(W(n)) = \mathsf{D}$, we have that n is even, so the game is a draw.

Subcase C3. For all k, $\ell_{\mathrm{par}(k)}(W(k)) = \mathsf{D}$ and W is infinite. Now by Cases 2 and 5 and the definition of σ_0, the sequence i_k as defined in (†) is a strictly descending sequence of natural numbers, yielding a contradiction.

Similarly (using Case B instead of Case A), we can show that σ_1 is a nonlosing strategy for player 1. Together, σ_0 and σ_1 witness that $\mathrm{val}(\mathsf{A}(\mathbf{G}, v)) = \mathsf{D}$.

Case D: $\ell_0(v) = \mathsf{L}$.

By Case 3, this means that player 0 is forced into a position w with $\ell_1(w) = \mathsf{W}$. Now apply Case B.

Case E: $\ell_1(v) = \mathsf{D}$.

This is dual to Case B.

Case F: $\ell_1(v) = \mathsf{L}$.

By **Case 6.**, this means that player 1 is forced into a position w with $\ell_0(w) = \mathsf{W}$. Now apply Case A. q.e.d.

4.1 True linear time

The 'linear-time' variant of this algorithm derives from the observation that only the *number* of labelled successors of a given vertex matters to the algorithm, and not (directly) their identities. Thus if we can by bookkeeping accurately track this number without storing and modifying sets, we avoid the additional cost of the inner loop.[5]

We define the auxiliary bookkeeping variable `Succ[G,j,e,a]`. This will store the number of successors of v_j labelled with `a` for player `e`. The complete algorithm is given in Procedures `MAIN` and `Label`, figure 4. The proof of linear time shows that the same operations are performed as in the original algorithm, but for the replacement of an inner loop by a constant-time lookup.

The additional data access required by this algorithm means that we cannot ensure that the data structure is notionally consistent at every timestep of the algorithm. For instance, during the loop at line 28, the values stored in `Succ[G,k,e,oldLbl]` will be incorrect for some `k`.

[5] A related approach was used in [Gij+76] to efficiently solve Edward de Bono's "L-Game"; here instead of tracking the number of non-lost successor positions the authors keep a 'safe' move from each non-lost position and update it efficiently if the 'safe' position becomes lost. The present authors are grateful to Peter van Emde Boas for bringing this report to their attention.

```
  Procedure:MAIN(G:graph)
  initialise loop:
  for i ← 1 to Nvert[G] do
4 |  Ell[G,i,0] ← L;
5 |  Ell[G,i,1] ← W;
  |  foreach j ∈ InBound[G,i] do
7 |  |  Succ[G,j,0,L] ← Succ[G,j,0,L] + 1;
8 |  |  Succ[G,j,1,W] ← Succ[G,j,1,W] + 1;
9 main loop:
  for i ← 1 to Nvert[G] do
  |  Label(G,i,0);
  |  Label(G,i,1);

  Procedure:Label(G:graph;i:integer;e:binary)
14 oldLbl ← Ell[G,j,e];
15 if Succ[G,j,1-e,L] > 0 then
  |  newLbl ← W;
  else if Succ[G,j,1-e,D] > 0 then
  |  newLbl ← D;
  else if Succ[G,j,1-e,W] > 0 then
  |  newLbl ← L;
  else if e == 0 then
  |  newLbl ← D;
  else
  |  newLbl ← L;
  update loop:
26 if newLbl ≠ oldLbl then
27 |  Ell[G,j,e] ← newLbl;
28 |  foreach k ∈ Inbound[G,j] do
29 |  |  Succ[G,k,e,oldLbl] ← Succ[G,k,e,oldLbl] - 1;
30 |  |  Succ[G,k,e,newLbl] ← Succ[G,k,e,newLbl] + 1;
31 |  recursive call loop:
   |  foreach k ∈ Inbound[G,j] do
33 |  |  Label[G,k,1-e];
```

Figure 4. The algorithm `NMBIG-Lin` for non-monotonic backward induction labelling in linear time.

Definition 18. We define the **checkpoints** of the algorithm as all calls to `Label`, and number them $t \in \mathbb{N}$ in computation order. The labelling function $\ell_e^t(v_i)$ is the value of `Ell[G,i,e]` at *checkpoint* t, rather than computation step t.

Lemma 19. At each checkpoint $t \in \mathbb{N}$,

$$\texttt{Succ[G,j,e,a]} = \big|\{k \in \texttt{Outbound[G,j]} \,;\, \texttt{Ell[G,k,e]} == \texttt{a}\}\big|.$$

Proof. This value is initialised in lines 7 and 8, and is by inspection correct when the main loop is entered at line 9. The value is also altered at lines 29 and 30, and here lines 14 and 27 ensure that the alterations reflect the changes being made to `Ell[G,j,e]`, by the time the recursive call at line 33 is reached. q.e.d.

Proposition 20. For each $i \leq N$, the sequence $\langle \ell^t(v_i) \;;\; t \in \mathbb{N}\rangle$ is $\leq^*$-decreasing in L^2. In other words, the sequence $\langle \ell_0^t(v_i) \;;\; t \in \mathbb{N}\rangle$ is $\leq$-decreasing and the sequence $\langle \ell_1^t(v_i) \,;\, t \in \mathbb{N}\rangle$ is $\leq$-decreasing.

Proof. Let $t+1$ be the least number such that

$$\ell^t(v_i) <^* \ell^{t+1}(v_i) \qquad (*)$$

for some i. The value of the bilabelling can be changed at code lines 4 and 5 of `MAIN`, and line 27 of `Label`. As in the original algorithm, the first two of these are simply initialisation to the top value of the lattice and need not concern us.

The new value at line 27 is the result of the if-cascade from line 15. We can show that a situation as in $(*)$ can only occur at time $t+1$ if it has already occurred at some $s \leq t$, thus that no such least time exists. The proof is by cases.

Suppose $\ell_1^t(v_i) = \mathsf{D}$ and $\ell_1^{t+1}(v_i) = \mathsf{W}$. Since `Ell[G,i,1]` is initialised to `W`, there must be some $s \leq t$ least such that $\ell_1^s(v_i) \neq \mathsf{D}$ and $\ell_1^{s+1}(v_i) = \mathsf{D}$ (that is, between s and $s+1$, v_i gets labelled by line 27). By the if-cascade from line 15, at checkpoint s we have `Succ[G,i,0,L]` $= 0$, but at t `Succ[G,i,0,L]` > 0. By Lemma 19 this means that for some $j \in$ `OutBound[G,i]`, $\ell_0^s(v_j) \in \{\mathsf{W},\mathsf{D}\} \mathrel{\overline{<}} \ell_0^t(v_j) = \mathsf{L}$. But then $s < t$ and $\ell^s(v_j) <^* \ell^t(v_j)$, contradicting choice of $t+1$ least.

The cases for $\ell_1^t(v_i) = \mathsf{L}$ and $\ell_1^{t+1}(v_i) \in \{\mathsf{W},\mathsf{D}\}$ are similar. Here the if-cascade guarantees for some $s \leq t$ and $j \in$ `OutBound[G,i]` that $\ell_0^s(v_j) = \mathsf{W} \mathrel{\overline{<}} \ell_0^t(v_j) \in \{\mathsf{L},\mathsf{D}\}$. Again this gives us $s < t$ such that $\ell_0^s(v_j) <^* \ell_0^t(v_j)$, contradicting choice of $t+1$ least.

The three cases for player 0 are symmetrical. First, suppose $\ell_0^t(v_i) = \mathsf{D}$ and $\ell_0^{t+1}(v_i) = \mathsf{L}$. Initialisation of `Ell[G,i,0]` to `L` means that there is least $s \leq t$ such that $\ell_0^s(v_i) \neq \mathsf{D}$ and $\ell_0^{s+1}(v_i) = \mathsf{D}$. This means the labelling at $s+1$ was produced by line 27. By the if-cascade, at s no successor

of v_i is 1-labelled with L, and for some $j \in \texttt{OutBound[G,i]}$, $\ell_1^s(v_j) = \mathsf{D}$. But at checkpoint t the if-cascade guarantees that $\ell_1^t(v_j) = \mathsf{W}$, and thus $\ell^s(v_j) <^* \ell^t(v_j)$, contradicting choice of $t+1$ least.

For the next two cases, suppose $\ell_0^t(v_i) = \mathsf{W}$ and $\ell_0^t(v_i) \in \{\mathsf{D}, \mathsf{L}\}$. Initialisation and the if-cascade give us $s \leq t$ and $j \in \texttt{OutBound[G,i]}$ such that $\ell_1^s(v_j) = \mathsf{L}$, and $\ell_1^t(v_j) \in \{\mathsf{D}, \mathsf{W}\}$. But then $s < t$ and $\ell^s(v_j) <^* \ell^t(v_j)$, contradicting choice of $t+1$ least. q.e.d.

Proposition 21. The procedure `Label` is called at most $2 \cdot |V_G| + 4 \cdot |E_G|$ times.

Proof. The main loop at line 9 calls `Label` twice for every vertex. Lines 26 and 31 associate with each recursive call an edge and a relabelling. By Proposition 20 each edge can be used for no more than four relabellings. q.e.d.

Theorem 22. The running time of `NMBIG-Lin` is $O(|V_G| + |E_G|)$. If G is connected, this is $O(|E_G|)$.

Proof. Apart from the calls to `Label`, `MAIN` walks every vertex and every edge in G. We can associate each pass through the loop at line 28 with a recursive call to `Label`. Apart from these recursive calls, `Label` is constant-time. Since `Label` is called $O(|E_G|)$ times, the total running time is $O(|V_G| + |E_G|)$. q.e.d.

Theorem 23. If run on the graph G, the algorithm `NMBIG-Lin` computes the A-sound bilabelling on G.

Proof. It is sufficient to show that this algorithm performs the same labellings (in the same order) as `NMBIG`.

First, note that the loop from line 9 only directly relabels vertices with no successors. Such vertices are labelled D/L, as in `NMBIG`, by the if-cascade. This means that calls to `Label` from `MAIN` have exactly the same effect as in `NMBIG`.

Next, given Lemma 19 it is easy to see that the if-cascade assigns to `newLbl` the same value that line 15 of the `NMBIG` algorithm assigns to `aux`. This ensures that the same recursive calls are made as in `NMBIG`, and since the update loop of line 28 finishes before the recursive labelling loop starts, the results of these calls will also be the same. q.e.d.

References.

[Fra97] Aviezri S. **Fraenkel**, Combinatorial Game Theory Foundations Applied to Digraph Kernels, **Electronic Journal of Combinatorics** 4 (1997), R10 [The Wilf Festschrift]

[GalSte53] David **Gale** and Frank M. **Stewart**, Infinite Games with Perfect Information, *in:* [KuhTuc53, p. 245–266]

[Gij+76] Vincent W. **Gijlswijk**, Gerard A. P. **Kindervater**, Koos G. J. **van Tubergen**, and Jan J. O. O. **Wiegerinck**, Computer Analysis of E. de Bono's L-Game, Technical Report # 76-18, Department of Mathematics, University of Amsterdam 1976

[GräThoWil02] Erich **Grädel**, Wolfgang **Thomas**, and Thomas **Wilke** (*eds.*), Automata, Logics, and Infinite Games, Springer 2002 [Lecture Notes in Computer Science 2500]

[KuhTuc53] Harold W. **Kuhn** and Albert W. **Tucker** (*eds.*), Contributions to the Theory of Games, volume 2, Princeton University Press 1953 [Annals of Mathematics Studies 28]

[Löw03] Benedikt **Löwe**, Determinacy for Infinite Games with More Than Two Players with Preferences, ILLC Publications PP-2003-19

[Maz02] René **Mazala**, Infinite Games, *in:* [GräThoWil02, p. 23–42]

[Sim99] Stephen G. **Simpson**, Subsystems of Second Order Arithmetic, Springer 1999 [Perspectives in Mathematical Logic]

[Ste76] John R. **Steel**, Determinateness and Subsystems of Analysis, *PhD thesis*, University of California at Berkeley, 1976

[Tan90] Kazuyuki **Tanaka**, Weak Axioms of Determinacy and Subsystems of Analysis I: Δ^0_2 Games, **Zeitschrift für Mathematische Logik und Grundlagen der Mathematik** 36 (1990), p. 481–491

Received: October 19th, 2005;
In revised version: December 11th, 2006;
Accepted by the editors: December 11th, 2006.