Parity Games and Automata for Game Logic

Helle Hvid Hansen[1], Clemens Kupke *[2], Johannes Marti *[2], and Yde Venema[3]

[1] Delft University of Technology, Delft, The Netherlands
[2] University of Strathclyde, Glasgow, United Kingdom
[3] University of Amsterdam, Amsterdam, The Netherlands

Abstract. Parikh's game logic is a PDL-like fixpoint logic interpreted on monotone neighbourhood frames that represent the strategic power of players in determined two-player games. Game logic translates into a fragment of the monotone μ-calculus, which in turn is expressively equivalent to monotone modal automata. Parity games and automata are important tools for dealing with the combinatorial complexity of nested fixpoints in modal fixpoint logics, such as the modal μ-calculus. In this paper, we (1) discuss the semantics a of game logic over neighbourhood structures in terms of parity games, and (2) use these games to obtain an automata-theoretic characterisation of the fragment of the monotone μ-calculus that corresponds to game logic. Our proof makes extensive use of structures that we call syntax graphs that combine the ease-of-use of syntax trees of formulas with the flexibility and succinctness of automata. They are essentially a graph-based view of the alternating tree automata that were introduced by Wilke in the study of modal μ-calculus.

1 Introduction

Game logic was introduced by Parikh [23] as a modal logic for reasoning about strategic power in determined 2-player games, and it can be seen as a generalisation of PDL [16] both in terms of syntax and semantics. On the syntax side, game logic is a multi-modal language in which modalities are labelled by games, which in turn are built from atomic games, the PDL program constructs together with the operation *dual* which switches the role of the players. A modal formula $\langle\alpha\rangle\varphi$ should be read as *"player 1 has a strategy in the game α to achieve an outcome that satisfies the formula φ"*. On the semantic side, one goes from PDL to game logic by moving from Kripke frames to monotone neighbourhood frames. A game perspective on this generalisation is that nondeterministic programs (i.e., relations) are 1-player games in which the player chooses his move from a set of successors, and monotone neighbourhood frames are 2-player games where player 1 first chooses a neighbourhood U, and then player 2 chooses an element in U. The shift from Kripke frames to monotone neighbourhood frames also means that we go from normal modal logic to monotone modal logic. Just as PDL (and other fixpoint logics such as LTL and CTL*) can be viewed as a fragment of the modal μ-calculus [20, 2], game logic can be naturally viewed as

* Supported by EPSRC grant EP/N015843/1.

a fragment of the *monotone μ-calculus* [24], which is monotone (multi-) modal logic with explicit fixpoint operators. A notable difference is that PDL, LTL and CTL* are all contained in level 1 or 2 of the alternation hierarchy whereas game logic, due to the combination of dual and iteration, spans all levels of the alternation hierarchy [1]. This high level of expressiveness could be an explanation for why a completeness proof for game logic is still missing.

In this paper we contribute to the theory of game logic. We discuss the semantics of game logic over neighbourhood structures using parity games and then use these games to characterise a class of automata that is exactly as expressive as formulas in game logic. Parity games are an intuitive way of dealing with the nesting of least and greatest fixpoint operators, and together with automata they play a fundamental role in the theory of fixpoint logics [12]. For instance, parity games and automata have been used in proving complexity results for the modal μ-calculus [8, 7] and also Walukiewicz' completeness result [27] is proved by automata-theoretic means. Some of these results have been extended to the setting of coalgebraic fixpoint logic [10]. In particular, they are applicable to the monotone μ-calculus. Since monotone modal μ-calculus is expressively equivalent to a naturally defined class of *(unguarded) monotone modal automata* [11], it is of interest to find out which subclass of these automata corresponds to game logic. The main result in our paper is a characterisation of a class of unguarded monotone modal automata that effectively corresponds to game logic, in the sense that there are effective translations in both directions. This result can be seen as the game logic analogue of the characterisation of PDL in automata-theoretic terms [3]. The case of game logic, however, is more involved because composition of games does not distribute from the left over choice as is the case for the programs in PDL. This is related to the fact that in the relational semantics of PDL, diamonds distribute over disjunctions; this property, which is heavily exploited in the mentioned results on PDL, does not apply to the diamonds of game logic. Finally, note that our characterisation can also be seen as an automata-theoretic counterpart to the results in [4, sec. 3.3] that characterise a fragment of the μ-calculus that is expressively equivalent to game logic interpreted over Kripke frames.

Our characterisation goes via a class of structures that we call *syntax graphs*. Syntax graphs combine the ease-of-use of syntax trees of formulas with the flexibility and succinctness of automata. They are essentially the same as Wilke's alternating tree automata (ATAs) [29] except they are described in terms of their transition graphs, and they run on monotone neighbourhood models rather than Kripke models. Unguarded monotone modal automata can, in turn, be viewed as Wilke's ATAs with complex transition condition [29] (again with a semantics over monotone neighbourhood models). As noted in [29, 19] an ATA with complex transition conditions can be effectively translated into an equivalent ATA, and this construction is easily seen to work also for monotone semantics. Concretely, our characterisation consists of a number of conditions that define a subclass **GG** of syntax graphs that correspond to game logic formulas. We call these *game logic graphs*. A game automaton is then a monotone modal automa-

ton whose corresponding syntax graph (i.e. ATA) is in **GG**. The translation from formulas to game logic graphs is an inductive construction similar to the construction of a nondeterministic automaton from a regular expression. Conversely, the defining conditions on game logic graphs allow us to decompose a game logic graph into components that correspond to formulas.

The rest of the paper is structured as follows. In Section 2 we recall the syntax and neighbourhood semantics of game logic and describe a normal form that is needed for our results. In Section 3 we introduce the game semantics for game logic and prove it to be equivalent to the neighbourhood semantics. In Section 4 we discuss syntax graphs and their game semantics. In Section 5 we define game logic graphs and prove them to be expressively equivalent to formulas in game logic. Due to space constraints, proofs are provided in an extended version of this paper [15].

2 Game Logic

Most definitions and results in this section are from [23, 25]. The syntax of game logic is based on the syntax of propositional modal logic with the additional feature that modal operators are labelled with terms that denote games. Since we have "test games" of the form $\varphi?$, the definition of the syntax is a simultaneous recursion on the structure of formulas and games.

Definition 1. *Throughout the paper we fix a countable set* Prop *of atomic propositions (proposition letters) and a set* Gam *of atomic games. The sets* $\mathcal{F}$ *of formulas and* $\mathcal{G}$ *of game terms of game logic are defined recursively as follows:*

$$\mathcal{F} \ni \varphi ::= p \in \mathsf{Prop} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\alpha\rangle\varphi, \quad \text{where } \alpha \in \mathcal{G}$$
$$\mathcal{G} \ni \alpha ::= g \in \mathsf{Gam} \mid \alpha^d \mid \alpha \cup \alpha \mid \alpha \cap \alpha \mid \alpha;\alpha \mid \alpha^* \mid \alpha^\times \mid \varphi? \mid \varphi!, \quad \text{where } \varphi \in \mathcal{F}$$

We use the standard definitions of $\rightarrow$ *and* $\leftrightarrow$*, and note that* $\top$ *can be defined as* $p \vee \neg p$ *for any* $p \in \mathsf{Prop}$*. In the following we denote formulas by* $\varphi, \psi, \ldots$ *and game terms with* $\alpha, \beta, \rho, \ldots$*. We use the letter* χ *to denote arbitrary terms that could either be a formula or a game term.*

The formulas of game logic express strategic power in 2-player determined, zero-sum games. A formula $\langle\alpha\rangle\varphi$ says that player 1 has a strategy in the game α to ensure that the outcome of the game satisfies φ. The assumption that the games are determined and zero-sum means that in a given game α, player 2 has a strategy to achieve φ iff player 1 does not have a strategy to achieve $\neg\varphi$. Hence the formula $\neg\langle\alpha\rangle\neg\varphi$, usually written as $[\alpha]\varphi$, says that player 2 has a strategy in α to ensure an outcome that satisfies φ. For technical reasons we do not include boxes as primitive operators.

It will be convenient to refer to player 1 as Angel and player 2 as Demon. The game operations can then be explained as follows. The composition $\alpha;\beta$ is the game consisting of playing α followed by β. The angelic choice $\alpha \cup \beta$ (resp. demonic choice $\alpha \cap \beta$) is the game in which Angel (resp. Demon) chooses whether

to play α or β. The angelic iteration α^* is the game in which α is played 0 or more times, and after each time, Angel chooses whether to stop or play again, but she must stop after some finite number of iterations. The demonic iteration $\alpha^\times$ is the iterated game in which Demon chooses when to stop, and he may choose to play forever. The formula $\langle\alpha^*\rangle\varphi$ thus says that Angel has a strategy to reach a φ-state by playing α some finite number of rounds (where her strategy may depend on what Demon did in previous rounds, so that in particular, the number of rounds needed to reach φ is not determined at the start of the game). The formula $\langle\alpha^\times\rangle\varphi$ says that Angel has a strategy for maintaining φ indefinitely when playing α repeatedly. Finally, the dual game α^d is the same as α but with the roles of the two players reversed, i.e., Angel has a strategy to achieve φ in α^d iff Demon has a strategy to achieve φ in α, and vice versa.

In [23, 25], the language of game logic only contained the game operations $;, \cup, ^*, ^d$, and the demonic operations were defined as $\alpha \cap \beta = (\alpha^d \cup \beta^d)^d$ and $\alpha^\times = ((\alpha^d)^*)^d$. We take the demonic operations as primitives, since later we want to reduce formulas to dual and negation normal form.

The formal semantics of game logic is given by representing games as monotone neighbourhood frames. These are well known semantic structures in modal logic [5, 13].

Definition 2. *Let S be a set. We denote by $\mathcal{M}(S)$ the set of up-closed subsets of $\mathcal{P}(S)$, i.e., $\mathcal{M}(S) = \{N \subseteq \mathcal{P}(S) \mid \forall U, U' : U \in N, U \subseteq U' \Rightarrow U' \in N\}$. A* monotone neighbourhood frame *on S is a function $f\colon S \to \mathcal{M}(S)$. We denote by* $\mathrm{MF}(S)$ *the set of all monotone neighbourhood frames on S.*

For $f \in \mathrm{MF}(S)$ and $s \in S$, the subsets U in $f(s)$ are called the neighbourhoods of s. We point out that such neighbourhoods are not necessarily neighbourhoods in the topological sense. In particular, we do not require that a state s is an element of all its neighbourhoods. In our setting, the neighbourhoods will be the subsets that Angel can force in the game represented by f.

We note that $(\mathcal{M}(S), \subseteq)$ is a complete partial order with associated join and meet given by union and intersection of neighbourhood collections. This CPO structure lifts pointwise to a CPO $(\mathrm{MF}(S), \sqsubseteq)$ in which we also denote join and meet by $\cup$ and $\cap$.

In analogue with how the PDL program operations are interpreted in relation algebra, we interpret game operations via algebraic structure on $\mathrm{MF}(S)$.[4]

Definition 3 (Game operations). *Let $f, f_1, f_2 \in \mathrm{MF}(S)$ be monotone neighbourhood frames. We define*

- *the unit frame η_S by:* $U \in \eta_S(s)$ *iff* $s \in U$ *for $s \in S$ and $U \subseteq S$.*
- *the composition $f_1 ; f_2$ by:*

$$U \in (f_1 ; f_2)(s) \text{ iff } \{s' \in S \mid U \in f_2(s')\} \in f_1(s) \qquad \text{for } s \in S \text{ and } U \subseteq S.$$

[4] It is well-known that $\mathcal{M}$ is a monad, [14]. Readers who are familiar with monads will recognise that unit and composition correspond to the unit and Kleisli composition.

– *the Angelic choice and Demonic choice between* f_1 *and* f_2 *by:*

$$(f \cup g)(s) = f(s) \cup g(s) \qquad (f \cap g)(s) = f(s) \cap g(s), \qquad \text{for } s \in S.$$

– *the dual* f^d *by:* $\quad U \in f^d(s) \quad \text{iff} \quad S \setminus U \notin f(s) \qquad \text{for } s \in S \text{ and } U \subseteq S.$
– *the angelic iteration* $f^* := \mathrm{LFP}(\mathsf{A}_f)$,
– *the demonic iteration* $f^\times := \mathrm{GFP}(\mathsf{D}_f)$,

where $\mathrm{LFP}(\mathsf{A}_f)$ *and* $\mathrm{GFP}(\mathsf{D}_f)$ *are the least and greatest fixed points of the maps*

$$\begin{array}{rl} \mathsf{A}_f : \mathrm{MF}(S) \to \mathrm{MF}(S) & \mathsf{D}_f : \mathrm{MF}(S) \to \mathrm{MF}(S) \\ g \mapsto \eta_S \cup (f \,;\, g) & g \mapsto \eta_S \cap (f \,;\, g) \end{array}$$

Note that for any $f \in \mathrm{MF}(S)$*, the map* $g \mapsto f \,;\, g$ *is a monotone operation on* $(\mathrm{MF}(S), \sqsubseteq)$ *and hence so are* A_f *and* D_f*. By the Knaster-Tarski theorem,* A_f *and* D_f *have unique least and greatest fixed points.*

It is straightforward to verify that $\mathrm{MF}(S)$ is closed under the above operations. The following lemma lists a number of identities that will be useful in reasoning about game logic semantics.

Lemma 1. *For all* $f, g \in \mathrm{MF}(S)$*, we have:*

1. $(f^d)^d = f$
2. $(f \,;\, g)^d = f^d \,;\, g^d$
3. $(\eta_S)^d = \eta_S$
4. $(f \cup g)^d = f^d \cap g^d$
5. $(f \cap g)^d = f^d \cup g^d$
6. $f \subseteq g \;\Rightarrow\; g^d \subseteq f^d$
7. $(f^*)^d = (f^d)^\times$
8. $(f^\times)^d = (f^d)^*$

We now have all the definitions in place to define game models and the semantics of formulas and games. We first give some intuitions. A game model consists of a state space together with interpretations of atomic propositions (as subsets of the state space) and atomic games (as monotone neighbourhood frames). The semantics of complex formulas and complex games is then defined by mutual induction. For a formula φ, the semantics $[\![\varphi]\!]$ is defined via the usual definitions from monotone modal logic. For a game α, the semantics $\langle\!|\alpha|\!\rangle$ is a monotone neighbourhood frame defined via the game constructions given above. The subsets U in $\langle\!|\alpha|\!\rangle(s)$ are the sets of outcomes that Angel can "force" when playing the game α in state s.

Definition 4. *A* game model *is a triple* $\mathbb{S} = (S, \gamma, \Upsilon)$ *where* S *is a set of states,* $\gamma : \mathsf{Gam} \to \mathrm{MF}(S)$ *is a* Gam*-indexed collection of monotone neighbourhood frames, which provides an interpretation of atomic games, and* $\Upsilon : \mathsf{Prop} \to \mathcal{P}(S)$ *is a valuation of atomic propositions. For* $\varphi \in \mathcal{F}$ *and* $\alpha \in \mathcal{G}$ *we define the*

semantics $[\![\varphi]\!]_{\mathbb{S}} \subseteq S$ *and* $\langle\!|\alpha|\!\rangle_{\mathbb{S}} \in \mathrm{MF}(S)$ *by induction on the term structure:*

$$
\begin{array}{rlrl}
[\![p]\!]_{\mathbb{S}} & := \Upsilon(p) \quad \textit{for } p \in \mathsf{Prop} & [\![\neg\varphi]\!]_{\mathbb{S}} & := S \setminus [\![\varphi]\!]_{\mathbb{S}} \\
[\![\varphi_1 \vee \varphi_2]\!]_{\mathbb{S}} & := [\![\varphi_1]\!]_{\mathbb{S}} \cup [\![\varphi_2]\!]_{\mathbb{S}} & [\![\varphi_1 \wedge \varphi_2]\!]_{\mathbb{S}} & := [\![\varphi_1]\!]_{\mathbb{S}} \cap [\![\varphi_2]\!]_{\mathbb{S}} \\
[\![\langle\alpha\rangle\varphi]\!]_{\mathbb{S}} & := \{s \in S \mid [\![\varphi]\!]_{\mathbb{S}} \in \langle\!|\alpha|\!\rangle_{\mathbb{S}}(s)\} & \langle\!|\alpha;\beta|\!\rangle_{\mathbb{S}} & := \langle\!|\alpha|\!\rangle_{\mathbb{S}} \,;\, \langle\!|\beta|\!\rangle_{\mathbb{S}} \\
\langle\!|g|\!\rangle_{\mathbb{S}} & := \gamma(g) \textit{ for } g \in \mathsf{Gam} & \langle\!|\alpha^d|\!\rangle_{\mathbb{S}} & := (\langle\!|\alpha|\!\rangle_{\mathbb{S}})^d \\
\langle\!|\alpha \cup \beta|\!\rangle_{\mathbb{S}} & := \langle\!|\alpha|\!\rangle_{\mathbb{S}} \cup \langle\!|\beta|\!\rangle_{\mathbb{S}} & \langle\!|\alpha \cap \beta|\!\rangle_{\mathbb{S}} & := \langle\!|\alpha|\!\rangle_{\mathbb{S}} \cap \langle\!|\beta|\!\rangle_{\mathbb{S}} \\
\langle\!|\alpha^*|\!\rangle_{\mathbb{S}} & := (\langle\!|\alpha|\!\rangle_{\mathbb{S}})^* & \langle\!|\alpha^\times|\!\rangle_{\mathbb{S}} & := (\langle\!|\alpha|\!\rangle_{\mathbb{S}})^\times \\
\langle\!|\psi?|\!\rangle_{\mathbb{S}} & := \lambda x. \begin{cases} \eta_S(x) & \textit{if } x \in [\![\psi]\!]_{\mathbb{S}} \\ \emptyset & \textit{otherwise.} \end{cases} & \langle\!|\psi!|\!\rangle_{\mathbb{S}} & := \lambda x. \begin{cases} \eta_S(x) & \textit{if } x \notin [\![\psi]\!]_{\mathbb{S}} \\ \mathcal{P}S & \textit{otherwise.} \end{cases}
\end{array}
$$

We write $\varphi \equiv \psi$ if for all $\mathbb{S}$, $[\![\varphi]\!]_{\mathbb{S}} = [\![\psi]\!]_{\mathbb{S}}$. Similarly, we write $\alpha \equiv \beta$ if for all $\mathbb{S}$, $\langle\!|\alpha|\!\rangle_{\mathbb{S}} = \langle\!|\beta|\!\rangle_{\mathbb{S}}$. We will often omit the subscript $\mathbb{S}$, if $\mathbb{S}$ is clear from the context, or irrelevant.

The following lemma states some basic identities involving the dual operator, and a congruence property.

Lemma 2. *Let* $\varphi, \psi \in \mathcal{F}$ *and* $\alpha, \beta \in \mathcal{G}$*. We have:*

1. $(\alpha^d)^d \equiv \alpha$
2. $(\alpha;\beta)^d \equiv \alpha^d;\beta^d$
3. $(\alpha \cup \beta)^d \equiv \alpha^d \cap \beta^d$
4. $(\alpha \cap \beta)^d \equiv \alpha^d \cup \beta^d$
5. $(\alpha^*)^d \equiv (\alpha^d)^\times$
6. $(\alpha^\times)^d \equiv (\alpha^d)^*$
7. $(\psi?)^d \equiv (\neg\psi)!$
8. $(\psi!)^d \equiv (\neg\psi)?$
9. $\langle\alpha^d\rangle\varphi \equiv \neg\langle\alpha\rangle\neg\varphi$
10. *If* $\alpha \equiv \beta$ *and* $\varphi \equiv \psi$ *then* $\langle\alpha\rangle\varphi \equiv \langle\beta\rangle\psi$

We will make frequent use of the fact that all formulas and game terms can be reduced to a dual and negation normal form.

Definition 5. *A formula* $\varphi \in \mathcal{F}$*, resp. game term* $\alpha \in \mathcal{G}$*, is in* dual and negation normal form (DNNF) *if dual is only applied to atomic games and negations occur only in front of proposition letters. We denote by* $\mathcal{F}_{\mathrm{DNNF}}$ *the set of formulas in DNNF, and by* $\mathcal{G}_{\mathrm{DNNF}}$ *the set of game terms in DNNF.*

Lemma 3. *For all* $\varphi \in \mathcal{F}$*, there is a DNNF formula* $\mathrm{nf}(\varphi)$ *such that* $\varphi \equiv \mathrm{nf}(\varphi)$*. For all* $\alpha \in \mathcal{G}$*, there is a DNNF game term* $\mathrm{nf}(\alpha)$ *such that* $\alpha \equiv \mathrm{nf}(\alpha)$*.*

From now on we will generally assume that formulas are in DNNF. The following lemma lists some crucial validities that form the basis for the definition of the game semantics in the next section. It is straightforward to verify that these formulas are valid.

Lemma 4. *The following formulas are valid in all game models:*

$$
\begin{array}{rlrl}
\langle\alpha;\beta\rangle\varphi & \leftrightarrow \langle\alpha\rangle\langle\beta\rangle\varphi & \langle\alpha^d\rangle\varphi & \leftrightarrow \neg\langle\alpha\rangle\neg\varphi \\
\langle\alpha \cup \beta\rangle\varphi & \leftrightarrow \langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi & \langle\alpha \cap \beta\rangle\varphi & \leftrightarrow \langle\alpha\rangle\varphi \wedge \langle\beta\rangle\varphi \\
\langle\alpha^*\rangle\varphi & \leftrightarrow \varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle\varphi & \langle\alpha^\times\rangle\varphi & \leftrightarrow \varphi \wedge \langle\alpha\rangle\langle\alpha^\times\rangle\varphi \\
\langle\psi?\rangle\varphi & \leftrightarrow \psi \wedge \varphi & \langle\psi!\rangle\varphi & \leftrightarrow \psi \vee \varphi
\end{array}
$$

3 Game Semantics for Game Logic

In this section we will see how games provide an operational semantics for game logic. In particular, we will develop a two-player evaluation game for game logic, very much in the spirit of Berwanger [1]. Note however, that the ambient model-theoretic structures in our setting are *monotone neighbourhood structures*, whereas Berwanger restricts to (relational) Kripke structures. Our approach allows for a neat formulation of some useful additional observations involving the unfolding games related to monotone operations on full powersets [26].

3.1 Game Preliminaries

Two-player *graph games* are an important tool for fixpoint logics. We will briefly recall their definition and the related terminology. For a more comprehensive account of these games, the reader is referred to [12]. A graph game is played on a *board* B, that is, a set of *positions*. Each position $b \in B$ *belongs* to one of the two *players*, Eloise (abbr. $\exists$) and Abelard (abbr. $\forall$). Formally we write $B = B_\exists \cup B_\forall$, and for each position b we use $P(b)$ to denote the player i such that $b \in B_i$. Furthermore, the board is endowed with a binary relation E, so that each position $b \in B$ comes with a set $E[b] \subseteq B$ of *successors*. Note that we do not require the games to be strictly alternating, i.e., successors of positions in $B_\exists$ or $B_\forall$ can lie again in $B_\exists$ or $B_\forall$, respectively. Formally, we say that the *arena* of the game consists of a directed two-sorted graph $\mathbb{B} = (B_\exists, B_\forall, E)$.

A *match* or *play* of the game consists of the two players moving a pebble around the board, starting from some *initial position* b_0. When the pebble arrives at a position $b \in B$, it is player $P(b)$'s turn to move; (s)he can move the pebble to a new position of their liking, but the choice is restricted to a successor of b. Should $E[b]$ be empty then we say that player $P(b)$ *got stuck* at the position. A *match* or *play* of the game thus constitutes a (finite or infinite) sequence of positions $b_0 b_1 b_2 \dots$ such that $b_i E b_{i+1}$ (for each i such that b_i and b_{i+1} are defined). A *full play* is either (i) an infinite play or (ii) a finite play in which the last player got stuck. A non-full play is called a *partial* play. Each full play of the game has a *winner* and a *loser*. A finite full play is lost by the player who got stuck; the winning condition for infinite games is usually specified using a so-called *parity function*, ie., a function $\Omega : B \to \mathbb{N}$ that maps each position to a natural number (its *priority*) and that has finite range. An infinite play $\Pi = b_0 b_1 \dots b_n \dots \in B^\omega$ is won by Eloise if $\max\{\Omega(b) \mid b \in \mathrm{Inf}(\Pi)\}$ is even, where $\mathrm{Inf}(\Pi)$ denotes the positions from B that occur infinitely often in Π. Otherwise Abelard wins this play. A graph game with parity function Ω is a *parity game*. All graph games used in this paper are parity games, but we will not specify the parity function explicitly in simple cases (e.g. when one of the players is supposed to win all infinite plays).

A *strategy* for player i tells player i how to play at all positions where it is i's turn to move. A strategy can be represented as a *partial* function which maps partial plays $\beta = b_0 \cdots b_n$ with $P(b_n) = i$ to legal next positions (that is, to elements of $E[b_n]$), and which is undefined for partial plays $\beta = b_0 \cdots b_n$ with

$E[b_n] = \emptyset$. We say that a play $\Pi = b_1 \dots b_n \dots \in B^* \cup B^\omega$ follows a strategy f if for all positions b_j in Π on which f is defined we have $f(b_j) = b_{j+1}$. A strategy is *positional* if it only depends on the current position of the match. A strategy is *winning for player i* from position $b \in B$ if it guarantees i to win any match with initial position b, no matter how the adversary plays — note that this definition also applies to positions b for which $P(b) \neq i$. A position $b \in B$ is called a *winning position* for player i, if i has a winning strategy from position b; the set of winning positions for i in a game $\mathcal{F}$ is denoted as $\mathrm{Win}_i(\mathcal{F})$. Parity games are *positionally determined*, i.e., at each position of the game board exactly one of the players has a positional winning strategy (cf. [22, 9]).

3.2 Definition of the Evaluation Game

In order to be able to trace the unfoldings of fixpoint operators within games we need some terminology concerning the nesting of fixpoints. Firstly, we need notation for the subterm relation and the definition of a parity map for a formula.

Definition 6. *We let $\lhd \subseteq (\mathcal{F} \cup \mathcal{G})^2$ be the subterm relation on formulas and game terms, ie., $\xi_1 \lhd \xi_2$ if either $\xi_1 = \xi_2$ or ξ_1 is a proper subterm of ξ_2.*

Definition 7. *For a term $\xi \in \mathcal{F} \cup \mathcal{G}$ we let* $\mathrm{Fix}(\xi) := \{\alpha^* \mid \alpha \in \mathcal{G}, \alpha^* \lhd \xi\} \cup \{\alpha^\times \mid \alpha \in \mathcal{G}, \alpha^\times \lhd \xi\}$. *A parity function for a formula φ in DNNF is a partial map $\Omega : \mathrm{Fix}(\varphi) \to \omega$ such that*

1. *$\alpha_1 \lhd \alpha_2$ implies $\Omega(\alpha_1) < \Omega(\alpha_2)$ for all $\alpha_1, \alpha_2 \in \mathrm{Fix}(\varphi)$ with $\alpha_1 \neq \alpha_2$, and*
2. *for all $\alpha \in \mathrm{Fix}(\varphi)$, $\Omega(\alpha)$ is even iff $\alpha = \rho^\times$ is a demonic iteration.*

We define the canonical parity function $\Omega_{\mathrm{can}} : \mathrm{Fix}(\varphi) \to \omega$ associated with φ as the partial function given by $\Omega_{\mathrm{can}}(\alpha^*) = 2n + 1$ and $\Omega_{\mathrm{can}}(\alpha^\times) = 2n$ where $n = \#\mathrm{Fix}(\alpha^*)$ and $n = \#\mathrm{Fix}(\alpha^\times)$, respectively. The canonical parity function formalises the fact that any fixpoint operator dominates any other fixpoint operator in its scope.

Definition 8. *Let $\mathbb{S} = (S, \gamma, \Upsilon)$ be a game model, let $\varphi \in \mathcal{F}$ be a formula in DNNF and let $\Omega : \mathrm{Fix}(\varphi) \to \omega$ be a parity function for φ. We define the evaluation game $\mathcal{E}(\mathbb{S}, \varphi)$ as the parity graph game with the game board specified in Fig. 1 and the parity function $\Omega_{\mathcal{E}}$ given by*

$$\Omega_{\mathcal{E}}(b) := \begin{cases} \Omega(\alpha) & \text{if } b = (x, \langle\alpha\rangle\psi) \text{ for some } \alpha \in \mathrm{Fix}(\varphi) \\ 0 & \text{otherwise.} \end{cases}$$

3.3 Adequacy of Game Semantics

In this section we show that the game semantics of Definition 8 is equivalent to the standard semantics of game logic from Definition 4 where we assume w.l.o.g. that formulas are in DNNF.

Formula Part

Position b	P(b)	Moves E[b]
$(s,p), s \in \Upsilon(p)$	$\forall$	$\emptyset$
$(s,p), s \notin \Upsilon(p)$	$\exists$	$\emptyset$
$(s,\neg p), s \in \Upsilon(p)$	$\exists$	$\emptyset$
$(s,\neg p), s \notin \Upsilon(p)$	$\forall$	$\emptyset$
$(s, \varphi \wedge \psi)$	$\forall$	$\{(s,\varphi),(s,\psi)\}$
$(s, \varphi \vee \psi)$	$\exists$	$\{(s,\varphi),(s,\psi)\}$
$(s, \langle g\rangle\varphi)$	$\exists$	$\{(U, \langle g\rangle\varphi) \mid U \in \langle\!\langle g\rangle\!\rangle(s)\}$
$(U, \langle g\rangle\varphi)$	$\forall$	$\{(s,\varphi) \mid s \in U\}$
$(s, \langle g^d\rangle\varphi)$	$\forall$	$\{(U, \langle g^d\rangle\varphi) \mid U \in \langle\!\langle g\rangle\!\rangle(s)\}$
$(U, \langle g^d\rangle\varphi)$	$\exists$	$\{(s,\varphi) \mid s \in U\}$

Game Part

Position b	P(b)	Moves E[b]
$(s, \langle \alpha\,;\beta\rangle\varphi)$	$\star$	$\{(s, \langle\alpha\rangle\langle\beta\rangle\varphi)\}$
$(s, \langle \alpha\cup\beta\rangle\varphi)$	$\star$	$\{(s, \langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi)\}$
$(s, \langle \alpha\cap\beta\rangle\varphi)$	$\star$	$\{(s, \langle\alpha\rangle\varphi \wedge \langle\beta\rangle\varphi)\}$
$(s, \langle \alpha^*\rangle\varphi)$	$\star$	$\{(s, \varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle\varphi)\}$
$(s, \langle \alpha^\times\rangle\varphi)$	$\star$	$\{(s, \varphi \wedge \langle\alpha\rangle\langle\alpha^\times\rangle\varphi)\}$
$(s, \langle \psi?\rangle\varphi)$	$\star$	$\{(s, \psi \wedge \varphi)\}$
$(s, \langle \psi!\rangle\varphi)$	$\star$	$\{(s, \psi \vee \varphi)\}$

Fig. 1. Game board of the evaluation game. We use $P(b) = \star$ to express that it is irrelevant which player moves, since there is exactly one possible move.

To compare the two different semantics we need a game characterisation of the $(\_)^*$ and $(\_)^\times$-operations. As both operations are defined as fixpoints they can be characterised via fixpoint games (these games are straightforward adaptation of the unfolding game described in [26]). We provide some intuition below the definition.

Definition 9. *Let $\alpha \in \mathcal{G}$ be a game term, let $\mathbb{S} = (S, \gamma, \Upsilon)$ be a game model and let $U \subseteq S$. The games $\mathcal{F}(\mathbb{S}, \alpha^*, U)$ and $\mathcal{F}(\mathbb{S}, \alpha^\times, U)$ have the following game boards:*

Board of $\mathcal{F}(\mathbb{S}, \alpha^*, U)$:

Pos. b	*$P(b)$*	*Moves $E[b]$*
$s \in S$	$\exists$	$\begin{cases} \{\emptyset\} & \text{if } s \in U \\ \langle\!\langle\alpha\rangle\!\rangle(s) & \text{otherwise.} \end{cases}$
$U' \in \mathcal{P}(S)$	$\forall$	U'

Board of $\mathcal{F}(\mathbb{S}, \alpha^\times, U)$:

Pos. b	*$P(b)$*	*Moves $E[b]$*
$s \in S$	$\exists$	$\begin{cases} \langle\!\langle\alpha\rangle\!\rangle(s) & \text{if } s \in U \\ \emptyset & \text{otherwise.} \end{cases}$
$U' \in \mathcal{P}(S)$	$\forall$	U'

The winning conditions in these games are as usual: finite complete plays are lost by the player that gets stuck. Infinite plays of $\mathcal{F}(\mathbb{S}, \alpha^*, U)$ and $\mathcal{F}(\mathbb{S}, \alpha^\times, U)$ are won by Abelard and Eloise, respectively.

The fixpoint game $\mathcal{F}(\mathbb{S}, \alpha^*, U)$ works as follows. The objective of Eloise is to reach U in finitely many rounds of α. At a position $s \in U$, Eloise can win by choosing the move $\emptyset$ which causes Abelard to get stuck in the next step, since he must choose from the empty set of moves. At a position $s \notin U$, Eloise chooses an α-neighbourhood U' of s, and in the next step Abelard then chooses a state $s' \in U'$, and the game continues. In the game $\mathcal{F}(\mathbb{S}, \alpha^\times, U)$, the objective of Eloise is to stay in U indefinitely. At a position $s \notin U$, she therefore loses immediately (indeed, she is stuck at such positions, since her set of moves is empty). But at a position $s \in U$, the players play another round of α, and the game continues.

Lemma 5. *For all* $\mathbb{S} = (S, \gamma, \Upsilon)$, $\alpha \in \mathcal{G}$, $s \in S$ *and* $U \subseteq S$, *we have:*

$s \in \mathrm{Win}_\exists(\mathcal{F}(\mathbb{S}, \alpha^*, U))$ *iff* $U \in \langle\!|\alpha^*|\!\rangle(s)$, *and*
$s \in \mathrm{Win}_\exists(\mathcal{F}(\mathbb{S}, \alpha^\times, U))$ *iff* $U \in \langle\!|\alpha^\times|\!\rangle(s)$.

The lemma easily follows because the games $\mathcal{F}(\mathbb{S}, \alpha^*, U)$ and $\mathcal{F}(\mathbb{S}, \alpha^\times, U)$ are instances of Tarski's fixpoint games that characterise least and greatest fixpoints of a monotone operator.

The following technical lemma demonstrates that winning strategies for Eloise in the evaluation game entail the existence of certain neighbourhood sets in the game model that witness the truth of a modal formula. There is no requirement on the witness to be non-empty, e.g., $s \models \langle\alpha\rangle\bot$ if $\emptyset \in \langle\!|\alpha|\!\rangle(s)$.

Lemma 6. *Let* $\varphi \in \mathcal{F}$, *let* $\mathbb{S} = (S, \gamma, \Upsilon)$ *be a game model and consider the game* $\mathcal{E} = \mathcal{E}(\mathbb{S}, \varphi)$. *Assume that* $f_\exists$ *is a winning strategy for Eloise in* $\mathcal{E}$, *and that* $(s, \langle\alpha\rangle\psi) \in Win_\exists(\mathcal{E})$. *Let* $\mathrm{Win}_\psi(\mathcal{E}) := \{s' \in S \mid (s', \psi) \in \mathrm{Win}_\exists(\mathcal{E})\}$ *and suppose* $\mathrm{Win}_\psi(\mathcal{E}) \subseteq [\![\psi]\!]$. *Then* $\mathrm{Win}_\psi(\mathcal{E}) \in \langle\!|\alpha|\!\rangle(s)$.

The lemma is the key to prove one direction of the adequacy of our game semantics.

Proposition 1. *Let* $\varphi \in \mathcal{F}$, *let* $\mathbb{S} = (S, \gamma, \Upsilon)$ *be a game model and consider* $\mathcal{E} = \mathcal{E}(\mathbb{S}, \varphi)$. *For all* ψ *occurring in* $\mathcal{E}$ *we have* $\mathrm{Win}_\psi(\mathcal{E}) \subseteq [\![\psi]\!]_\mathbb{S}$.

The claim is proven by induction on ψ and follows easily from Lemma 6. For the second half of the adequacy theorem we again need a technical lemma.

Lemma 7. *Let* $\mathbb{S} = (S, \gamma, \Upsilon)$ *be a game model and let* $\varphi \in \mathcal{F}$. *For any position* $(s, \langle\alpha\rangle\psi)$ *of the game* $\mathcal{E} = \mathcal{E}(\mathbb{S}, \varphi)$ *and for all* $U \subseteq [\![\psi]\!]_\mathbb{S}$ *with* $U \in \langle\!|\alpha|\!\rangle(s)$ *Eloise has a strategy* $f_\exists$ *such that for each finite* $\mathcal{E}$*-play* Π *starting at* $(s, \langle\alpha\rangle\psi)$ *and following* $f_\exists$ *either Abelard gets stuck or* Π *reaches a state* $(s', \xi') \in S \times \mathcal{F}$ *that satisfies one of the following conditions: (i)* $\xi' \lhd \alpha$ *and* $s' \in [\![\xi']\!]$, *or (ii)* $\xi' = \psi$ *and* $s' \in U$.

Proposition 2. *Let* $\mathbb{S} = (S, \gamma, \Upsilon)$ *be a game model and consider the game* $\mathcal{E} = \mathcal{E}(\mathbb{S}, \varphi)$ *for some* $\varphi \in \mathcal{F}$. *There is a strategy* $f_\exists$ *for Eloise that is winning for Eloise for all game positions* (s, ψ) *such that* $s \in [\![\psi]\!]_\mathbb{S}$.

In summary, Proposition 1 and Proposition 2 imply that our game semantics for game logic is adequate:

Theorem 1. *Let* $\mathbb{S} = (S, \gamma, \Upsilon)$ *be a game model and consider the game* $\mathcal{E} = \mathcal{E}(\mathbb{S}, \varphi)$ *for some* $\varphi \in \mathcal{F}$. *Then for all positions* (s, ψ) *in* $\mathcal{E}$ *we have* $(s, \psi) \in \mathrm{Win}_\exists(\mathcal{E})$ *iff* $\mathbb{S}, s \models \psi$.

4 Syntax Graphs

In this section we introduce syntax graphs which we then use later to provide an automata-theoretic characterisation of game logic. Syntax graphs are a generalisation of syntax trees that allow cycles and sharing of subterms. Another perspective is that they are a graph-based description of the alternating tree automata from [29, 19]. We discuss the precise connection after the definition of syntax graphs and their game semantics.

4.1 Graph Basics

We first recall some basic notions and fix notation. A graph is a pair $\mathbb{G} = (V, E)$ where V is a set of vertices V and $E \subseteq V \times V$ is a set of edges. We will use the following notation: vEw iff $(v, w) \in E$ iff $w \in E(v)$, and call w a successor of v.

Let $\mathbb{G} = (V, E)$ be a graph. A *path* p in $\mathbb{G}$ is a sequence of vertices $p = v_1 \dots v_n$ such that $v_i E v_{i+1}$ for all $i < n$. We say that v_n is *reachable* from v_1 if a path $p = v_1 \dots v_n$ exists. Note that every vertex is always reachable from itself. A *cycle* $c = v_1 \dots v_n$ is any path such that $v_1 = v_n$ and $n \geq 2$.

A path $p = v_1 \dots v_n$ is *simple* if all the v_i for $i \leq n$ are distinct. A cycle $c = v_1 \dots v_n$ is *simple* if all the v_i for $i < n$ are distinct Every path can be *contracted* to a simple path with the same start and end points, To see how this works consider a path p that contains a repetition of some vertex $u \in V$. This means that p is of the form $p = qumur$, for paths q, m and r. We contract p to the path qur with the same starting and end points, in which there is one less occurrence of u. We can repeat this procedure until we obtain a simple path.

A *pointed* graph $\mathbb{G} = (V, E, v_I)$ is a graph (V, E) together with a $v_I \in V$ that we call the *initial* vertex of $\mathbb{G}$. If $\mathbb{G}$ is a graph (V, E) or a pointed graph (V, E, v) and v_I is a vertex in $\mathbb{G}$, we define $\mathbb{G}@v_I = (V', E', v_I)$ to be the *subgraph generated by* v_I *in* $\mathbb{G}$, i.e., V' is the set of vertices that are reachable from v_I and $E' = E \cap (V' \times V')$.

A pointed graph $\mathbb{G} = (V, E, v_I)$ is *reachable* if every $v \in V$ is reachable from v_I. Note that $\mathbb{G}@v_I$ is always reachable.

4.2 Syntax Graphs

We define the following sets of label symbols: $\mathsf{Lit} = \mathsf{Lb}_0 := \{p, \neg p \mid p \in \mathsf{Prop}\}$, $\mathsf{Latt} = \mathsf{Lb}_2 := \{\wedge, \vee\}$ and $\mathsf{Mod} = \mathsf{Lb}_1 := \{\langle g \rangle \mid g \in \mathsf{Gam}\} \cup \{\langle g^d \rangle \mid g \in \mathsf{Gam}\}$. The labels $\mathsf{Lb}_0, \mathsf{Lb}_1, \mathsf{Lb}_2$ can be given an arity in the expected manner, namely, for $l \in \mathsf{Lb}_i$, $\mathsf{arity}(l) = i$. We let $\mathsf{Lb} := \mathsf{Lb}_0 \cup \mathsf{Lb}_1 \cup \mathsf{Lb}_2$.

Definition 10. *A* syntax graph $\mathbb{G} = (V, E, L, \Omega)$ *is a finite graph* (V, E) *together with a* labelling function $L : V \to \mathsf{Lb}$ *and a partial* priority function $\Omega : V \rightharpoonup \omega$ *satisfying the following two conditions:*

(arity condition) *For all* $v \in V$, $|E(v)| = \mathsf{arity}(L(v))$.
(priority condition) *On every simple cycle of* (V, E) *there is at least one vertex on which* Ω *is defined.*

Later we will show that formulas correspond to syntax graphs, and game terms correspond to syntax graphs with a special atomic proposition that marks an "exit" from the graph. The idea is that a game term α is viewed as the modality $\langle\alpha\rangle$ which still needs a formula φ in order to become a formula $\langle\alpha\rangle\varphi$, and an exit marks a place in the graph where φ can be inserted.

Definition 11. *A proposition letter e is an* exit *of a syntax graph $\mathbb{G} = (V, E, L, \Omega)$ if there is a vertex $v \in V$ with $L(v) = e$ and there is no $v \in V$ with $L(v) = \neg e$.*

We say that a proposition letter p is reachable *from a vertex v in $\mathbb{G}$ if there is some vertex u that is reachable from v in $\mathbb{G}$ with $L(u) = p$ or $L(u) = \neg p$. The* priority *of a path (or cycle) $p = v_1 \dots v_n$ is defined by*

$$\Omega(p) = \max\left(\{-1\} \cup \{\Omega(v_i) \mid 1 \leq i \leq n\}\right),$$

i.e., $\Omega(p) = -1$ if Ω is undefined on all the v_i.

Due to the close connection between formulas and syntax graphs, we can define an acceptance game for syntax graphs in essentially the same way as in Definition 8, using that successors in the syntax graph can be viewed as subformulas.

Definition 12. *Let $\mathbb{G} = (V, E, L, \Omega, v_I)$ be a pointed syntax graph and $\mathbb{S} = (S, \gamma, \Upsilon, s_I)$ be a pointed game model. We define the* acceptance game $\mathcal{A} = \mathcal{A}(\mathbb{G}, \mathbb{S})$ *as a parity game with the game board as specified in Fig. 2, initial position (v_I, s_I) and priority function $\Omega_{\mathcal{A}}$ such that $\Omega_{\mathcal{A}}(v, s) = \Omega(v)$ if $\Omega(v)$ is defined and $\Omega_{\mathcal{A}}(v, s) = 0$ otherwise. If Eloise has a winning strategy in the game $\mathcal{A}(\mathbb{G}, \mathbb{S})$ then we say that $\mathbb{G}$* accepts *$\mathbb{S}$. We also write $\mathbb{S}, s \models \mathbb{G}$ to mean that Eloise has a winning strategy in the game $\mathcal{A}(\mathbb{G}, \mathbb{S})$ starting from position (v_I, s).*

Given a pointed syntax graph $\mathbb{G}$ and a formula φ, we write $\mathbb{G} \equiv \varphi$ if for all $\mathbb{S}$, Eloise has a winning strategy in $\mathcal{E}(\mathbb{S}, \varphi)$ iff she has one in $\mathcal{A}(\mathbb{G}, \mathbb{S})$.

Position b	$P(b)$	Moves $E[b]$
$(v,s), L(v) = p, s \in \Upsilon(p)$	$\forall$	$\emptyset$
$(v,s), L(v) = p, s \notin \Upsilon(p)$	$\exists$	$\emptyset$
$(v,s), L(v) = \neg p, s \in \Upsilon(p)$	$\exists$	$\emptyset$
$(v,s), L(v) = \neg p, s \notin \Upsilon(p)$	$\forall$	$\emptyset$
$(v,s), L(v) = \wedge$	$\forall$	$\{(w_0,s),(w_1,s)\}$, where $E(v) = \{w_0, w_1\}$
$(v,s), L(v) = \vee$	$\exists$	$\{(w_0,s),(w_1,s)\}$, where $E(v) = \{w_0, w_1\}$
$(v,s), L(v) = \langle g\rangle$	$\exists$	$\{(v,U) \mid U \in \langle\!|g|\!\rangle(s)\}$
$(v,U), L(v) = \langle g\rangle$	$\forall$	$\{(w,s) \mid s \in U, L(v) = \{w\}\}$
$(v,s), L(v) = \langle g^d\rangle$	$\forall$	$\{(v,U) \mid U \in \langle\!|g|\!\rangle(s)\}$
$(v,U), L(v) = \langle g^d\rangle$	$\exists$	$\{(w,s) \mid s \in U, L(v) = \{w\}\}$

Fig. 2. Game board of the acceptance game $\mathcal{A}(\mathbb{G}, \mathbb{S})$

A syntax graph is essentially a multi-modal version of an alternating tree automaton (ATA) with partial priority function as described in [29, sec. 2.2.5]. Namely, taking the transition graph of an ATA as defined in [29, sec. 2.2.4] and equipping this graph with the evident labelling function, yields a syntax graph. Conversely, given a syntax graph one constructs for each vertex a transition condition from its label and successors in the obvious manner. If desired, a partial priority function Ω can be made into a total map Ω' by defining $\Omega'(v) = \Omega(v)+2$ if $v \in V_P$ and $\Omega'(v) = 0$ otherwise. One easily adapts the notion of a run on a pointed Kripke structure from [29] to a run on a pointed game model (by dealing with modal transition conditions as in the modal positions of Definition 12) such that there exists an accepting run for the ATA on $\mathbb{S}$ iff Eloise has a winning strategy in the acceptance game for the corresponding syntax graph on $\mathbb{S}$.

As described in [29, sec. 2.2.5] and in more detail in [19, sec. 9.3.4] ATAs can be generalised to allow complex transition conditions (i.e. arbitrary formulas) without increasing their expressive power. The basic idea in transforming an ATA with complex transition condition into an equivalent ATA is to introduce new states for each node in the syntax tree of the transition conditions.

Monotone modal automata are obtained by instantiating the definition of Λ-automaton from [11] with the functor $\mathcal{M}^{\mathsf{Gam}}$ and taking Λ to be a suitable set of predicate liftings. Monotone modal automata and their unguarded variants are expressively complete for the monotone (multi-modal) μ-calculus. On the other hand, unguarded monotone modal automata are essentially the same as ATAs with complex transition condition (running on monotone neighbourhood models for a multi-modal signature), hence by the above transformation, unguarded monotone modal automata can be viewed as syntax graphs, and vice versa.

We have chosen to work with syntax graphs rather than ATAs or monotone modal automata, since we characterise the game logic fragment mainly in terms of the graph structure. In the following section, we identify a class **GG** of syntax graphs that correspond to game logic formulas. By the correspondence just outlined, we can define game automata as those unguarded monotone modal automata for which the corresponding syntax graph (ATA) is in **GG**.

5 The Game Logic Fragment

In this section we define game logic graphs, which are a class of syntax graphs that has the same expressivity over neighbourhood frames as formulas in game logic. After giving the definition of game logic graphs, we show that for each game logic formula there is a game logic graph that accepts a pointed game model iff the formula is true at the model and, vice versa, for every game logic graph there is a game logic formula that is true at a pointed game model iff the game logic graphs accepts the model.

5.1 Game Logic Graphs

The idea behind the definition of game logic graphs is that cycles in the graph correspond to formulas of the form $\langle \alpha^* \rangle \varphi$ and $\langle \alpha^\times \rangle \varphi$. Consider e.g. the axiom for

$\langle\alpha^*\rangle\varphi$ (in Lem. 4). We see that the vertex v corresponding to the disjunction in $\varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle\varphi$ has a special role as a vertex on the corresponding cycle. Namely, let v_l and v_r be the two successors of v where going to v_l means *leaving* the cycle (going to subformula φ) and going to v_r means *remaining* on the cycle (going to subformula $\langle\alpha\rangle\langle\alpha^*\rangle\varphi$). We will refer to this v as the *head* of the cycle corresponding to $\langle\alpha^*\rangle\varphi$. If the cycles in the syntax graph arise from a nesting of fixpoint formulas, and Ω is the parity function of some formula (cf. Def. 7), then certain conditions will need to hold for the cycles and Ω. This is made precise in the following definition.

Definition 13. *Given a syntax graph $\mathbb{G} = (V, E, L, \Omega)$ in which Ω is injective, we let $h := \Omega^{-1} : \mathrm{ran}(\Omega) \to V$ denote the inverse of Ω on its range. We use the abbreviation $h_n := h(n)$ and call h_n the* head of priority n. *Whenever we write h_n, we presuppose that $n \in \mathrm{ran}(\Omega)$.*

A game logic graph *is a syntax graph $\mathbb{G} = (V, E, L, \Omega)$ in which Ω is injective and the following conditions hold for all $n \in \mathrm{ran}(\Omega)$:*

(parity) *$L(h_n) = \vee$ if n is odd and $L(h_n) = \wedge$ if n is even.*
(head) *There are maps $r, l : \mathrm{ran}(\Omega) \to V$, for which we also use the abbreviations $r_n := r(n)$ and $l_n := l(n)$, such that $E(h_n) = \{l_n, r_n\}$ and*
(leave) *For all simple paths $p = l_n \dots h_n$ we have that $\Omega(p) > n$.*
(remain) *There is no simple path $h_n r_n \dots h_m$ for any $m > n$.*

A game logic graph with exit *is a syntax graph with exit $\mathbb{G} = (V, E, L, \Omega, e)$ for which (V, E, L, Ω) is a game logic graph that additionally satisfies:*

(exit) *For all $n \in \mathrm{ran}(\Omega)$ and all $v \in V$ with $L(v) = e$, there is no simple path $h_n r_n \dots v$.*

5.2 From Formulas to Game Logic Graphs

Our first result in characterising the game logic fragment of syntax graphs shows that we can translate game logic formulas into equivalent game logic graphs.

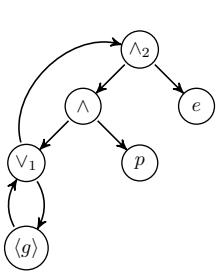
Theorem 2. *For every game $\alpha \in \mathcal{G}_{\mathrm{DNNF}}$ in which the proposition letter e does not occur, there is a pointed syntax graph $\mathbb{G}$ with exit e such that $\mathbb{G} \equiv \langle\alpha\rangle e$. For every game logic formula $\varphi \in \mathcal{F}_{\mathrm{DNNF}}$ there is a pointed syntax graph $\mathbb{G}$ such that $\mathbb{G} \equiv \varphi$.*

The proof of Theorem 2 is by a mutual induction on the structure of games and formulas, and is similar to the construction of a nondeterministic finite automaton from a regular expression [17], that is, we define constructions on syntax graphs that correspond to game operations and logical connectives. The recursive procedure itself is similar to the translation of game logic into the μ-calculus [24], with the difference that we directly translate into syntax graphs instead of formulas of the μ-calculus.

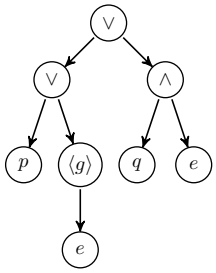
For example, we construct $\mathbb{G}_1 ; \mathbb{G}_2$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are given by the induction hypothesis by rerouting the edges that went to an exit vertex in $\mathbb{G}_1$ to go to

the initial state of $\mathbb{G}_2$. The priority function Ω for $\mathbb{G}_1 ; \mathbb{G}_2$ is unchanged on the $\mathbb{G}_2$ part, but in order to make sure Ω is injective we shift all priority values in $\mathbb{G}_1$ by adding to them a number k that preserves the parity and ensures that all priorities in the $\mathbb{G}_1$ part are higher than those in the $\mathbb{G}_2$ part. The correctness of the construction is proved by constructing winning strategies in the evaluation game from winning strategies in the acceptance game, and vice versa. A detailed proof is provided in [15].

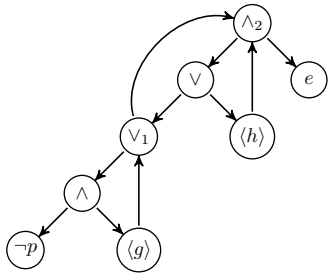
Example 1. Below we show the syntax graphs of some formulas. The initial vertex is the topmost vertex, and priorities are indicated as subscripts on the vertex labels.



$\varphi = \langle (p?; g^*)^\times \rangle e$



$\varphi = \langle (p!; g) \cup q? \rangle e$



$\varphi = \langle (((\neg p)?; g)^* \cup h)^\times \rangle e$

5.3 From Game Logic Graphs to Formulas

We now show how to transform game logic graphs into equivalent game logic formulas.

Theorem 3. *For every pointed game logic graph with exit $\mathbb{G} = (V, E, L, \Omega, e, v_I)$ there is a game term $\delta \in \mathcal{G}$, not containing e and only containing propositional letters that are reachable from v_I, such that $\mathbb{G} \equiv \langle \delta \rangle e$.*

The proof of Theorem 3 is by induction on the number of heads in the game logic graph. In the base case there are no heads which implies that there are no cycles in the graph, which makes it easy to recursively decompose the graph into a game term. In the inductive step we use a construction that removes some of the edges at the head with the highest priority and thus cutting all cycles that pass through the highest priority head. This allows us to remove the priority from this head and obtain a simpler game logic graph to which we can apply the induction hypothesis. A detailed proof is provided in [15].

Because any propositional letter e that does not occur in $\mathbb{G}$ can be added as an exit to a game logic graph $\mathbb{G}$ we obtain the following corollary from Theorem 3:

Corollary 1. *For every pointed game logic syntax graph $\mathbb{G}$ there is a formula $\varphi \in \mathcal{F}$ such that $\mathbb{G} \equiv \varphi$.*

Example 2. We apply the construction from Theorem 3 to the graph on the left in Example 1. The heads h_1 and h_2 are the disjunction with priority 1 and the

conjunction with priority 2, respectively. We start the decomposition at h_2. We then first obtain a game $\delta_2 = \underline{\lambda_2^\times}$, where $\underline{\lambda_2^\times}$ is a dummy game term that is a place holder for the game through the left child of h_2, that describes how to reach the exit from the initial state without iterating at h_2. We also apply the induction hypothesis to obtain a new game λ_2 that describes one iteration from the left node to h_2, which we replace by a fresh exit e'. In this inductive step we then need to cut h_1. At h_1 we have $\delta_1 = \underline{\lambda_1^*} \cap p?\,;p!$ and $\lambda_1 = \langle g \rangle$. We then obtain λ_2 by substituting $\underline{\lambda_1^*}$ in δ_1 with λ_1^* and thus obtain $\lambda_1 = g^* \cap (p?\,;p!)$. Substituting $\lambda_1^\times$ for $\underline{\lambda_1^\times}$ in δ_2 yields the overall game $(g^* \cap (p?\,;p!))^\times$. Hence the game graph is equivalent to the formula $\langle (g^* \cap (p?\,;p!))^\times \rangle e$.

6 Conclusion

We have provided a semantics for game logic in terms of parity games. This was the key to obtain our main technical result, the characterisation of game logic graphs, i.e., a class of parity automata that correspond to game logic formulas.

These automata open several avenues for future research: Firstly, we would like to study normal forms in game logic. In the μ-calculus, automata are the key to obtain the (semi-)disjunctive normal forms of formulas which can be used to prove further results, e.g., completeness, interpolation and the characterisation of the expressivity of the logic [18, 28, 6]. Our experience suggests that a similar normal form for game logic is out of reach, but a careful analysis of the cycle structure of game logic graphs might yield useful insights concerning the structure of game logic formulas. As a first step in this direction we are currently investigating how to obtain guarded game logic graphs and, consequently, a definition of guarded game logic formulas.

Furthermore, game logic constitutes a very general dynamic logic that makes very few assumptions on the algebraic properties of the modal operators. Therefore we believe that our game logic automata have the potential to help us understand a wider class of automata for families of dynamic logics such as coalgebraic dynamic logics [14] or many-valued dynamic logics as described in [21] or for a combination of these frameworks.

References

1. Berwanger, D.: Game Logic is strong enough for parity games. Studia Logica 75(2), 205–219 (2003)
2. Bradfield, J., Stirling, C.: Modal μ-calculi. In: Handbook of Modal Logic, pp. 721–756. Elsevier (2006)
3. Carreiro, F., Venema, Y.: PDL inside the μ-calculus: a syntactic and an automata-theoretic characterization. In: et alii, R.G. (ed.) Advances in Modal Logic, Volume 10. pp. 74–93. College Publications (2014)
4. Carreiro, F.: Fragments of Fixpoint Logics. Ph.D. thesis, University of Amsterdam (2015)
5. Chellas, B.F.: Modal logic, an introduction. Cambridge University Press (1980)

6. d'Agostino, G., Hollenberg, M.: Logical questions concerning the μ-calculus. Journal of Symbolic Logic 65, 310–332 (2000)
7. Emerson, E.A., Jutla, C.S.: The complexity of tree automata and logics of programs (extended abstract). In: Proceedings of the 29th Symposium on the Foundations of Computer Science. pp. 328–337. IEEE Computer Society Press (1988)
8. Emerson, E.A., Jutla, C.S., Sistla, P.: On model checking for the μ-calculus and its fragments. Theoretical Computer Science p. 491522 (2001)
9. Emerson, E., Jutla, C.: Tree automata, mu-calculus and determinacy. In: Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science (FoCS'91). pp. 368–377. IEEE (1991)
10. Enqvist, S., Seifan, F., Venema, Y.: Completeness for μ-calculi: a coalgebraic approach. Tech. Rep. PP-2017-04, ILLC, Universiteit van Amsterdam (2017)
11. Fontaine, G., Leal, R., Venema, Y.: Automata for coalgebras: An approach using predicate liftings. In: Automata, Languages and Programming: 37th International Colloquium ICALP'10. LNCS, vol. 6199, pp. 381–392. Springer (2010)
12. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logic, and Infinite Games, LNCS, vol. 2500. Springer (2002)
13. Hansen, H.H.: Monotonic Modal Logic. Master's thesis, University of Amsterdam (2003), iLLC Preprint PP-2003-24
14. Hansen, H.H., Kupke, C.: Weak completeness of coalgebraic dynamic logics. In: Fixed Points in Computer Science (FICS). EPTCS, vol. 191, pp. 90–104 (2015)
15. Hansen, H., Kupke, C., Marti, J., Venema, Y.: Parity games and automata for game logic (extended version) (2017), available on http://www.arxiv.org
16. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. The MIT Press (2000)
17. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (1979)
18. Janin, D., Walukiewicz, I.: Automata for the modal μ-calculus and related results. In: Proc. MFCS'95. pp. 552–562. Springer (1995), lNCS 969
19. Kirsten, D.: Alternating tree automata and parity games. In: Automata logics, and infinite games, LNCS, vol. 2500, pp. 405–411. Springer (2002)
20. Kozen, D.: Results on the propositional μ-calculus. Theoretical Computer Science 27, 333–354 (1983)
21. Madeira, A., Neves, R., Martins, M.: An exercise on the generation of many-valued dynamic logics. Journal of Log. and Alg. Meth. in Prog. 85(5), 1011 – 1037 (2016)
22. Mostowski, A.: Games with forbidden positions. Tech. Rep. 78, Instytut Matematyki, Uniwersytet Gdański, Poland (1991)
23. Parikh, R.: The logic of games and its applications. In: Topics in the Theory of Computation. No. 14 in Annals of Discrete Mathematics, Elsevier (1985)
24. Pauly, M.: Logic for Social Software. Ph.D. thesis, University of Amsterdam (2001)
25. Pauly, M., Parikh, R.: Game Logic: An overview. Studia Logica 75(2), 165–182 (2003)
26. Venema, Y.: Lectures on the modal μ-calculus (2012), available on the author's webpage: https://staff.science.uva.nl/y.venema/
27. Walukiewicz, I.: On completeness of the mu-calculus. In: Proceedings of the Eighth Annual Symposium on Logic in Computer Science (LICS '93). pp. 136–146. IEEE Computer Society (1993)
28. Walukiewicz, I.: Completeness of Kozen's axiomatisation of the propositional μ-calculus. Information and Computation 157(1-2), 142–182 (2000), lICS 1995 (San Diego, CA)
29. Wilke, T.: Alternating tree automata, parity games, and modal μ-calculus. Bulletin of the Belgian Mathematical Society 8, 359–391 (2001)