Parity games and automata for game logic

Hansen, Helle Hvid; Kupke, Clemens; Marti, Johannes; Venema, Yde

DOI
10.1007/978-3-319-73579-5_8

Publication date
2018

Document Version
Accepted author manuscript

Published in
Dynamic Logic. New Trends and Applications - 1st International Workshop, DALI 2017, Proceedings

Citation (APA)
Hansen, H. H., Kupke, C., Marti, J., & Venema, Y. (2018). Parity games and automata for game logic. In *Dynamic Logic. New Trends and Applications - 1st International Workshop, DALI 2017, Proceedings* (Vol. 10669 LNCS, pp. 115-132). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 10669 LNCS). Springer. https://doi.org/10.1007/978-3-319-73579-5_8

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.

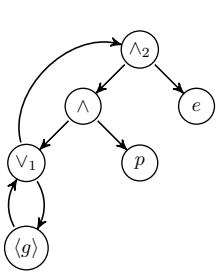
Theorem 2. *For every game $\alpha \in \mathcal{G}_{\mathrm{DNNF}}$ in which the proposition letter e does not occur, there is a pointed syntax graph $\mathbb{G}$ with exit e such that $\mathbb{G} \equiv \langle\alpha\rangle e$. For every game logic formula $\varphi \in \mathcal{F}_{\mathrm{DNNF}}$ there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

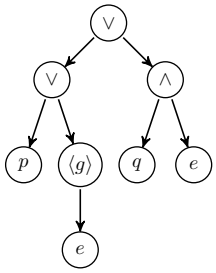
For example, we construct $\mathbb{G}_1 ; \mathbb{G}_2$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are given by the induction hypothesis by rerouting the edges that went to an exit vertex in $\mathbb{G}_1$ to go to

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].

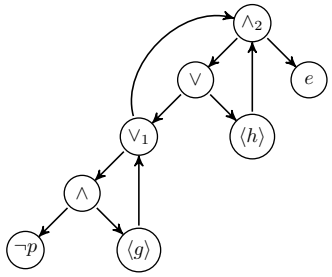
Example 1. Below we show the syntax graphs of some formulas. The initial vertex is the topmost vertex, and priorities are indicated as subscripts on the vertex labels.



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



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



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

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)

A Proofs from Section 2

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

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

Proof. Items 1, 3, 4, 5, 6 are straightforward. To see why item 2 holds:

$$\begin{array}{rl} U \in (f\,;g)^d(s) & \text{iff } S \setminus U \notin (f\,;g)(s) \\ \text{(by mon.)} & \text{iff } \forall V \in f(s)\ \exists t \in V : S \setminus U \notin g(t) \\ & \text{iff } \{s' \in S \mid S \setminus U \in g(s')\} \notin f(s) \\ & \text{iff } \{s' \in S \mid S \setminus U \notin g(s')\} \in f^d(s) \\ & \text{iff } \{s' \in S \mid U \in g^d(s')\} \in f^d(s) \\ & \text{iff } U \in (f^d\,;g^d)(s) \end{array}$$

Item 7: First observe that $U \in (f^*)^d(s)$ iff $S \setminus U \notin f^*(s) = \mathrm{LFP}(\mathsf{A}_f)(s)$ iff

$$\text{there is a } g \in \mathrm{MF}(S) : S \setminus U \notin g(s) \text{ and } \eta_S(s) \cup (f\,;g)(s) \subseteq g(s) \qquad (1)$$

and $U \in (f^d)^\times(s) = \mathrm{GFP}(\mathsf{D}_{f^d})$ iff

$$\text{there is a } h \in \mathrm{MF}(S) : U \in h(s) \text{ and } h(s) \subseteq \eta_S(s) \cap (f^d\,;h)(s) \qquad (2)$$

To see that (1) implies (2), take $h = g^d$. First, $S \setminus U \notin g(s)$ iff $U \in g^d(s) = h(s)$. Second, $g^d(s) \subseteq \eta_S(s)$ follows from $\eta_S(s) \subseteq g(s)$ and items 3 and 6. Similarly, $g^d(s) \subseteq (f^d\,;g^d)(s)$ follows from $(f\,;g)(s) \subseteq g(s)$ and items 2, 3 and 6. The implication (2) $\Rightarrow$ (1) follows by a similar argument taking $g = h^d$. Item 8 is proved in a similar manner as item 7. □

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* $\langle\alpha\rangle\varphi \equiv \langle\beta\rangle\psi$

Proof. Items 1-6 follow from Lemma 1, using the compositional semantics of the game operations. We show item 7 (item 8 can be proved similarly).

$$\begin{array}{rl} \text{Case } s \in [\![\psi]\!]_{\mathbb{S}} : U \in \langle\!|((\psi)?)^d|\!\rangle(s) & \text{iff } S \setminus U \notin \langle\!|(\psi)?|\!\rangle(s) \\ & \text{iff } s \notin S \setminus U \\ & \text{iff } s \in U \\ & \text{iff } U \in \eta_S(s) = \langle\!|\neg\psi!|\!\rangle(s). \end{array}$$

$$\begin{array}{ll} \text{Case } s \in [\![\psi]\!]_{\mathbb{S}} : U \in \langle\!|((\psi)?)^d|\!\rangle(s) & \text{iff } S \setminus U \notin \langle\!|(\psi)?|\!\rangle(s) \\ & \text{iff } S \setminus U \notin \emptyset \\ & \text{iff } \mathit{true} \\ & \text{iff } U \in \mathcal{P}(S) = \langle\!|\neg\psi!|\!\rangle(s). \end{array}$$

Items 9 and 10 are also straightforward to prove using the standard semantics in Definition 4. □

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 $\alpha \equiv \mathrm{nf}(\alpha)$.*

Proof. We define $\mathrm{nf}(\varphi)$ and $\mathrm{nf}(\alpha)$ inductively over formulas and game terms. On atomic propositions and atomic games, $\mathrm{nf}(-)$ acts as identity. On formulas with main connective different from $\neg$ and game terms with main constructor different from ^{d}, $\mathrm{nf}(-)$ just distributes over the main connective/constructor. In particular, $\mathrm{nf}(\langle\alpha\rangle\varphi) = \langle\mathrm{nf}(\alpha)\rangle\mathrm{nf}(\varphi)$. For the remaining cases, $\mathrm{nf}(\varphi)$ and $\mathrm{nf}(\alpha)$ are defined as follows:

$$\begin{array}{ll|ll} \mathrm{nf}(\neg p) & = \neg p, & \mathrm{nf}(g^d) & = g^d, \\ \mathrm{nf}(\neg\neg\varphi) & = \mathrm{nf}(\varphi), & \mathrm{nf}((\alpha^d)^d) & = \mathrm{nf}(\alpha), \\ \mathrm{nf}(\neg(\varphi \wedge \psi)) & = \mathrm{nf}(\neg\varphi) \vee \mathrm{nf}(\neg\psi), & \mathrm{nf}((\alpha \cup \beta)^d) & = \mathrm{nf}(\alpha^d) \cap \mathrm{nf}(\beta^d), \\ \mathrm{nf}(\neg(\varphi \vee \psi)) & = \mathrm{nf}(\neg\varphi) \wedge \mathrm{nf}(\neg\psi), & \mathrm{nf}((\alpha \cap \beta)^d) & = \mathrm{nf}(\alpha^d) \cup \mathrm{nf}(\beta^d), \\ \mathrm{nf}(\neg\langle\alpha\rangle\varphi) & = \langle\mathrm{nf}(\alpha^d)\rangle\mathrm{nf}(\neg\varphi), & \mathrm{nf}((\alpha;\beta)^d) & = \mathrm{nf}(\alpha^d);\mathrm{nf}(\beta^d), \\ \mathrm{nf}((\varphi?)^d) & = \mathrm{nf}(\neg\varphi)!, & \mathrm{nf}((\alpha^*)^d) & = \mathrm{nf}(\alpha^d)^\times, \\ \mathrm{nf}((\varphi!)^d) & = \mathrm{nf}(\neg\varphi)?, & \mathrm{nf}((\alpha^\times)^d) & = \mathrm{nf}(\alpha^d)^*, \end{array}$$

We prove that $\varphi \equiv \mathrm{nf}(\varphi)$ and $\alpha \equiv \mathrm{nf}(\alpha)$ by induction on the length of formulas and game terms, i.e. the number of symbols, not counting parentheses. For example the length of $\langle(p?;g^d)^*\rangle(\neg p \wedge q)$ is 10.

Base case: For atomic propositions and atomic games (which have length 1), it is trivial. Similarly for formulas and games of length 2. So assume the lemma holds for formulas and games of strictly shorter length.

Step: The induction step is straightforward using the IH and Lemma 2. For example, $\mathrm{nf}(\neg\langle\alpha\rangle\varphi) = \langle\mathrm{nf}(\alpha^d)\rangle\mathrm{nf}(\neg\varphi) \equiv \langle\alpha^d\rangle\neg\varphi \equiv \neg\langle\alpha\rangle\neg\neg\varphi \equiv \neg\langle\alpha\rangle\varphi$ where the second step uses the IH and Lemma 2(10). Note here that the length of α^d and $\neg\varphi$ are both less than the length of $\neg\langle\alpha\rangle\varphi$. The third step uses Lemma 2(9). □

B Proofs from Section 3

As a preparation for proving Lemma 6 we need to introduce some terminology.

Definition 14. *Let $\varphi \in \mathcal{F}$, let $\mathbb{S} = (S, \gamma, \Upsilon)$ be a game model and consider the game $\mathcal{E} = \mathcal{E}(\mathbb{S}, \varphi)$. Furthermore we let $f_\exists$ denote a strategy for Eloise. For $(s, \langle\alpha\rangle\psi)$ we denote by $\mathrm{Suc}_{f_\exists}(s, \langle\alpha\rangle\psi) \subseteq S$ the collection of all states $s' \in S$ such*

that there exists a (possibly partial) $\mathcal{E}$-play that follows $f_\exists$, that is of the form $(s, \langle\alpha\rangle\psi) \dots (s', \psi)$ and no position in between $(s, \langle\alpha\rangle\psi)$ and (s', ψ) is of the form (s'', ψ).

Remark 1. We spell out the definition of Suc for the test cases, as these cases tend to cause unnecessary confusion. For $\alpha = \xi?$ we have $\mathrm{Suc}_{f_\exists}(s, \langle\alpha\rangle\psi) = \{s\}$. For $\alpha = \xi!$ we have $\mathrm{Suc}_{f_\exists}(s, \langle\alpha\rangle\psi) = \emptyset$ if $f_\exists$ requires to move from $(s, \langle\alpha\rangle\psi)$ to (s, ξ) and we have $\mathrm{Suc}_{f_\exists}(s, \langle\alpha\rangle\psi) = \{s\}$ if $f_\exists$ requires to move from $(s, \langle\alpha\rangle\psi)$ to (s, ψ).

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 (i) $\mathrm{Suc}_{f_\exists}(s, \langle\alpha\rangle\psi) \in \langle\!|\alpha|\!\rangle(s)$, and (ii) $\mathrm{Win}_\psi(\mathcal{E}) \in \langle\!|\alpha|\!\rangle(s)$.*

Proof. The second item follows easily from the first, as all states that are reachable via some winning strategy of Eloise from a winning position must again be winning positions. In other words, $\mathrm{Suc}_{f_\exists}(s, \langle\alpha\rangle\psi) \subseteq \mathrm{Win}_\psi(\mathcal{E})$, hence by monotonicity (i) implies (ii). The first item can be proven by induction on the structure of α. We only discuss the fixpoint operators and angelic tests. All other cases are a matter of routine checking.

Case $\alpha = \alpha'^*$. By our assumption Eloise has a winning strategy at position $(s, \langle\alpha\rangle\psi)$ in $\mathcal{E}$. We show that she also has a winning strategy in the game $\mathcal{F} = \mathcal{F}(\mathbb{S}, \alpha^*, \mathrm{Suc}_{f_\exists}(s, \langle\alpha\rangle\psi)) \in \langle\!|\alpha|\!\rangle(s)$ at position s. To see this we equip Eloise with a strategy in $\mathcal{F}$ such that for any play

$$sU_1s_1 \dots U_ns_n$$

there is a "shadow play" in $\mathcal{E}$ of the form

$$(s, \langle\alpha'^*\rangle\psi) \dots (s_1, \langle\alpha'^*\rangle\psi) \dots (s_n, \langle\alpha'^*\rangle\psi)$$

that follows Eloise's winning strategy in $\mathcal{E}$. Suppose this connection has been established for n rounds (ie. for all plays of $\mathcal{F}$ of where Eloise moved at most n times) we are going to describe how to extend it to games with $n + 1$ rounds: Consider a $\mathcal{F}$-play of n rounds ending in position s_n and assume that Eloise has played according to the strategy that we are providing for her. By assumption there is an $\mathcal{E}$-play according to Eloise's winning strategy in $\mathcal{E}$ that reaches state $(s_n, \langle\alpha'^*\rangle\psi)$.

Subcase Eloise's $\mathcal{E}$-strategy $f_\exists$ requires a move to (s_n, ψ). In this case we have $s_n \in \mathrm{Suc}_{f_\exists}(s, \langle\alpha'^*\rangle\psi) \subseteq \mathrm{Win}_\exists$. Therefore the $\mathcal{F}$-play ending in s_n is complete and winning for Eloise as required.

Subcase Eloise's $\mathcal{E}$-strategy requires to move to $(s_n, \langle\alpha'\rangle\langle\alpha'^*\rangle\psi)$. As Eloise's strategy is winning we have $(s_n, \langle\alpha'\rangle\langle\alpha'^*\rangle\psi) \in \mathrm{Win}_\exists$ and by I.H. we get $U_{n+1} := \mathrm{Suc}_{f_\exists}((s_n, \langle\alpha'\rangle\langle\alpha'^*\rangle\psi) \in \langle\!|\alpha'|\!\rangle(s_n)$. We define Eloise's strategy in $\mathcal{F}$ to move from s_n to U_{n+1}. If $U_{n+1} = \emptyset$ we have that the $\mathcal{F}$-play is complete and Eloise wins as required. Otherwise Abelard picks some $s_{n+1} \in U_{n+1}$. By the definition of U_{n+1} this implies that the $\mathcal{E}$-shadow play can be prolonged from $(s_n, \langle\alpha'^*\rangle\psi)$ to $(s_{n+1}, \langle\alpha'^*\rangle\psi)$ following Eloise's winning strategy $f_\exists$.

In either subcase we showed how Eloise can prolong the $\mathcal{F}$-play - unless Abelard gets stuck - such that there is a suitable parallel $\mathcal{E}$-play that follows Eloise's winning strategy $f_\exists$. Furthermore, by Def. 14 of Suc we have that α^* is the fixpoint with the highest priority that is unfolded within this parallel $\mathcal{E}$-play. This means - as the $\mathcal{E}$-play is winning for Eloise- the parallel $\mathcal{E}$-play has to stop eventually which implies that Abelard has to get eventually stuck in the $\mathcal{F}$-play as well. In other words, Eloise wins each $\mathcal{F}$-play that starts in s and that is played according to the strategy that we devised for her. Thus $U \in \langle\!|\alpha'^*|\!\rangle(s)$ as required.

Case $\alpha = \alpha'^\times$. The proof for this case is analogous to the previous one.

Case $\alpha = \xi?$. By assumption Eloise has a winning strategy at $\langle\xi?\rangle\psi$ and we have $\mathrm{Win}_\psi \subseteq [\![\psi]\!]$. As $(s, \langle\xi?\rangle\psi)$ is winning for Eloise, we also have $(s, \psi) \in \mathrm{Win}_\exists$ and thus $s \in \mathrm{Win}_\psi \subseteq [\![\psi]\!]$. Therefore $\langle\!|\xi?|\!\rangle(s) = \eta_S(s)$ and hence $\mathrm{Suc}_{f_\exists}(s, \langle\xi?\rangle\psi) = \{s\} \in \langle\!|\xi?|\!\rangle(s)$ as required. □

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 $s' \in U$.*

Proof. The claim is proven by induction on α.

$\alpha = g$ If $U \in \langle\!|g|\!\rangle(s)$ for some $U \subseteq [\![\psi]\!]$, Eloise's strategy consists of moving to U, which obviously fulfils the conditions of the lemma.

$\alpha = \alpha_1; \alpha_2$ Let $U \in \langle\!|\alpha_1; \alpha_2|\!\rangle(s)$ with $U \subseteq [\![\psi]\!]$. This implies $U \in (\langle\!|\alpha_1|\!\rangle \,;\, \langle\!|\alpha_2|\!\rangle)(s)$, ie., $U' := \{s' \in S \mid U \in \langle\!|\alpha_2|\!\rangle(s')\} \in \langle\!|\alpha_1|\!\rangle(s)$. It is easy to see that $U' \subseteq [\![\langle\alpha_2\rangle\psi]\!]$. Eloise has to move in $\mathcal{E}$ from position $(s, \langle\alpha_1; \alpha_2\rangle)$ to $(s, \langle\alpha_1\rangle\langle\alpha_2\rangle\psi)$. To the latter position we can apply the I.H. which yields a strategy g for Eloise such that:

- Any play following g starting in $(s, \langle\alpha_1\rangle\langle\alpha_2\rangle\psi)$ either leads to some position $(s', \langle\alpha_2\rangle\psi)$ with $s' \in U'$ or a position (s', ξ) with $s' \in [\![\xi]\!]$ and $\xi \lhd \alpha_1$ or Abelard gets stuck before any of the above happens.
- Applying for each state of the form $\sigma = (s', \langle\alpha_2\rangle\psi)$ with $s' \in U'$ the I.H. yields a strategy g_σ such that each $\mathcal{E}$-play that starts at $(s', \langle\alpha_2\rangle\psi)$ and that follows g_σ ends in a state of the form (s'', ψ) with $s'' \in [\![\psi]\!]$ or to a state of the form (s'', ξ') with $s'' \in [\![\xi']\!]$ or Abelard gets stuck.

These facts show that a combination of the g_σ strategies with g will describe a suitable strategy $f_\exists$ for Eloise at position $(s, \langle \alpha_1; \alpha_2 \rangle \psi)$.

$\alpha = g^d$ If $U \in \langle\!|g^d|\!\rangle(s)$ for some $U \subseteq [\![\psi]\!]$ then for all $U' \in \langle\!|g^d|\!\rangle(s)$ we have $U' \cap U \neq \emptyset$. For any possible choice $U' \in \langle\!|g^d|\!\rangle(s)$ by Abelard we define Eloise's response according to her strategy $f_\exists$ to be some element of $U \cap U'$.

$\alpha = \xi?$ If $U \in \langle\!|\xi?|\!\rangle(s)$ for some $U \subseteq [\![\psi]\!]$ we have $s \in U$ and Eloise will move to position (s, ψ).

$\alpha = \xi!$ If $U \in \langle\!|\xi!|\!\rangle(s)$ for some $U \subseteq [\![\psi]\!]$ then either $s \in [\![\xi]\!]$ of $s \in U$. In the first case Eloise's strategy is to move to (s, ξ) In case $s \in U$, Eloise's strategy is to move to (s, ψ).

$\alpha = \alpha'^*$ Consider some $U \in \langle\!|\alpha'^*|\!\rangle(s)$ with $U \subseteq [\![\psi]\!]$. Thus Eloise has a winning strategy in $\mathcal{F} = \mathcal{F}(\mathbb{S}, \alpha'^*, U)$ at position s. Suppose her strategy in $\mathcal{F}$ is to move to some $U_\exists \subseteq S$.

Subcase If $s \in U$ we can assume that w.l.o.g. $U_\exists = \emptyset$. Then by I.H. there is a strategy for Eloise in $\mathcal{E}$ such that in any play following that strategy either Abelard gets stuck or a position (s', ξ') is reached with $\xi' \lhd \alpha'$ and $s' \in [\![\xi']\!]$. As $\xi' \lhd \langle \alpha'^* \rangle \psi$ this case meets the requirements of the lemma.

Subcase Otherwise we know that $U_\exists \in \langle\!|\alpha'|\!\rangle(s)$ and - as Eloise's strategy in $\mathcal{F}$ is winning - we can assume that $U_\exists \subseteq \mathrm{Win}_\exists(\mathcal{F}) = U$. Furthermore $U \in \langle\!|\alpha'|\!\rangle \, ; \langle\!|\alpha'^*|\!\rangle(s)$ and hence $V := \{s' \in S \mid U \in \langle\!|\alpha'^*|\!\rangle(s')\} \in \langle\!|\alpha'|\!\rangle(s)$. Because $V \subseteq \{s' \in S \mid [\![\psi]\!] \in \langle\!|\alpha'^*|\!\rangle(s')\}$ we have $V \subseteq [\![\langle \alpha'^* \rangle \psi]\!]$. This implies that $U \subseteq V$ as $s' \in U$ implies $U \in \langle\!|\alpha'^*|\!\rangle(s')$.

Collectively this shows that $U_\exists \in \langle\!|\alpha'|\!\rangle(s)$ and $U_\exists \subseteq U \subseteq V \subseteq [\![\langle \alpha'^* \rangle \psi]\!]$, ie, we can apply the I.H. to U_E and $\langle \alpha'^* \rangle \psi$.

We obtain a strategy g for Eloise in $\mathcal{E}$ such that in any play Π following g starting at $(s, \langle \alpha' \rangle \langle \alpha'^* \rangle \psi)$ either Abelard gets stuck or Π reaches a position (s', ξ') with $s \in [\![\xi']\!]$ (in these two cases we do not continue the play as the conditions of the lemma are met) or such that Π reaches a position $(s'', \langle \alpha'^* \rangle \psi)$ with $s'' \in U_\exists$. In the latter case we note that there is a shadow play $s \to U_\exists \to s''$ of $\mathcal{F}$ that follows Eloise's winning strategy and thus we can apply the same reasoning that we applied to $(s, \langle \alpha'^* \rangle \psi)$ to the new state $(s'', \langle \alpha'^* \rangle \psi)$ and so forth.

This implies that the second subcase can only occur finitely often as any corresponding $\mathcal{F}$-play has to be won be Eloise and thus needs to be finite. In other word, after finitely many iterations of our argument, either Abelard will get stuck or the play will reach a position of the form (s''', ψ) with $s''' \in [\![\psi]\!]$ as required. □

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 $s \in [\![\psi]\!]_{\mathbb{S}}$.*

Proof. The claim is proven by induction on the order $\lhd$ on formulas. We provide the details for some important cases:

$\psi = p$ In this case $s \in [\![p]\!]$ implies $s \in \Upsilon(p)$ and Eloise wins the play immediately.
$\psi = \neg p$ similar to previous case.
$\psi = \psi_1 \wedge \psi_2$ In this case $s \in [\![\psi_1 \wedge \psi_2]\!]$ means that $s \in [\![\psi_i]\!]$ for $i = 1, 2$. A move of Abelard from (s, ψ) to (s, ψ_i) will be answered by Eloise following her winning strategy $f_\exists$ that is defined at (s, ψ_i) by the I.H.

$\vdots$

$\psi = \langle\alpha\rangle\xi$ In this case, by Lemma 7, Eloise has a strategy f such that each play following f eventually reaches a position of the form (s', ξ') with $s' \in [\![\xi']\!]$, $\xi' \neq \psi$ and $\xi' \lhd \psi$ or Abelard gets stuck. We define Eloise's strategy at $(s, \langle\alpha\rangle\xi)$ to be f until in the resulting play one of those two possibilities happens: If Abelard gets stuck, Eloise wins the play and we are done. If the play reaches a position of the form (s', ξ') with $s' \in [\![\xi']\!]$ we let Eloise continue to play her winning strategy $f_\exists$ that exists by I.H. □

C Proofs from Section 5

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 $\mathbb{G} \equiv \varphi$.*

Proof. The proof is by a mutual induction on the structure of games and formulas. As mentioned before, games will correspond to syntax graphs with exit, and formulas will correspond to syntax graphs. The base cases consist of atomic games, their duals and literals. For all cases we describe the construction of the syntax graph and we argue that it satisfies the conditions from Definition 13. We leave it to the reader to verify, using the game semantics from Definitions 8 and 12 that that construction is adequate in the sense that $\mathbb{G} \equiv \langle\alpha\rangle e$ in the case of games and that $\mathbb{G} \equiv \varphi$ in the case for formulas. These arguments are tedious but in most cases straight-forward.

Atomic games. For $\alpha = g$ or $\alpha = g^d$ where $g \in \mathsf{Gam}$, we define the pointed syntax graph with exit $\mathbb{G} = (V, E, L, \Omega, e, v_I)$ by taking $V = \{v_I, w\}$ to be any two element set, $E = \{(v_I, w)\}$, $L(v_I) = \langle\alpha\rangle$ and $L(w) = e$. The priority function Ω can be taken to be the empty map, because there are no cycles in (V, E). Hence, $\mathbb{G}$ is trivially a game logic graph with exit e.

Literals. For $\varphi = p$ or $\alpha = \neg p$ for $p \in \mathsf{Prop}$, we define the pointed syntax graph $\mathbb{G} = (V, E, L, \Omega, v_I)$ by taking $V = \{v\}$ to be any singleton set, $E(v) = \emptyset$, $L(v) = \alpha$, $v_I = v$ and Ω as the empty map, since there are no cycles in (V, E). Again, $\mathbb{G}$ is trivially a game logic graph.

For the induction step, we define operations on syntax graphs that correspond to game constructs and formula constructs.

Demonic choice. Let $\alpha = \alpha_1 \cap \alpha_2$ be such that e does not occur in α. By the induction hypothesis, there are pointed syntax graphs with exit $\mathbb{G}_1 =$

$(V_1, E_1, L_1, \Omega_1, e, v_{I1})$ and $\mathbb{G}_2 = (V_2, E_2, L_2, \Omega_2, e, v_{I2})$ such that $\mathbb{G}_1 \equiv \langle\alpha_1\rangle e$ and $\mathbb{G}_2 \equiv \langle\alpha_2\rangle e$. We define $\mathbb{G}_1 \cap \mathbb{G}_2 = (V, E, L, \Omega, e, v_I)$ by taking $V = \{v_I\} + V_1 + V_2$ and $E = E_1 \cup E_2 \cup \{(v_I, v_{I1}), (v_I, v_{I2})\}$, and defining the labelling by $L(v_I) = \wedge$, $L(v) = L_i(v)$ if $v \in V_i$ for $i = 1, 2$. When defining the priority function Ω we lift all the priorities in $\mathbb{G}_1$ above all those in $\mathbb{G}_2$ to guarantee that Ω is injective. Thus, let k be the smallest even number that is strictly larger than the maximal priority occurring in $\mathbb{G}_2$. Then set

$$\Omega(v) = \begin{cases} k + \Omega_1(v) & \text{if } v \in V_1 \text{ and } \Omega_1(v) \text{ is defined,} \\ \Omega_2(v) & \text{if } v \in V_2 \text{ and } \Omega_2(v) \text{ is defined.} \end{cases}$$

Since the cycles of $\mathbb{G}_1$ and $\mathbb{G}_2$ cannot interfere with each other, $\mathbb{G}_1 \cap \mathbb{G}_2$ is a game logic graph with exit e.

Angelic choice. The syntax graph $\mathbb{G}_1 \cup \mathbb{G}_2$ is defined analogously to $\mathbb{G}_1 \cap \mathbb{G}_2$ above, just replacing the conjunction at the initial state with a disjunction.

Composition. Let $\alpha = \alpha_1 ; \alpha_2$. By induction hypothesis there are pointed syntax graphs with exit $\mathbb{G}_1 = (V_1, E_1, L_1, \Omega_1, e, v_{I1})$ and $\mathbb{G}_2 = (V_2, E_2, L_2, \Omega_2, e, v_{I2})$ such that $\mathbb{G}_1 \equiv \langle\alpha_1\rangle e$ and $\mathbb{G}_2 \equiv \langle\alpha_2\rangle e$. We define $\mathbb{G}_1 ; \mathbb{G}_2 = (V, E, L, \Omega, e, v_I)$ by taking $V = V_1 + V_2$ and

$$E(v) = \begin{cases} g[E_1(v)] & \text{if } v \in V_1, \\ E_2(v) & \text{if } v \in V_2 \end{cases} \qquad \text{where } g(v) = \begin{cases} v_{I2} & \text{if } L(v) = e, \\ v & \text{if } L(v) \neq e. \end{cases}$$

That is, we reroute all edges in $\mathbb{G}_1$ that lead to the exit of $\mathbb{G}_1$ to the initial state of $\mathbb{G}_2$. Note that all vertices that were labelled with e in $\mathbb{G}_1$ become unreachable from v_I in $\mathbb{G}_1 ; \mathbb{G}_2$. The labelling is unchanged, i.e., $L = L_1 \cup L_2$. Again, to ensure the priority map is injective, let k be the smallest even number that is strictly larger than the maximal priority occurring in $\mathbb{G}_2$, and define

$$\Omega(v) = \begin{cases} k + \Omega_1(v) & \text{if } v \in V_1 \text{ and } \Omega_1(v) \text{ is defined,} \\ \Omega_2(v) & \text{if } v \in V_2 \text{ and } \Omega_2(v) \text{ is defined.} \end{cases}$$

Finally, we set $v_I = v_{I1}$. We check that $\mathbb{G}_1 ; \mathbb{G}_2$ satisfies the conditions from Definition 13. Condition **(parity)** holds, since it holds for $\mathbb{G}_1$ and $\mathbb{G}_2$. Condition **(leave)** holds since every cycle of form $h_n l_n \dots h_n$ in $\mathbb{G}_1 ; \mathbb{G}_2$ either stays completely $\mathbb{G}_1$ or completely in $\mathbb{G}_2$, because there are no connections from the $\mathbb{G}_2$-part into the $\mathbb{G}_1$-part of $\mathbb{G}_1 ; \mathbb{G}_2$. To check Condition **(remain)**, note that all priorities in the $\mathbb{G}_2$-part of $\mathbb{G}_1 ; \mathbb{G}_2$ are lower than those in the $\mathbb{G}_1$-part and there are no paths from the $\mathbb{G}_2$-part into the $\mathbb{G}_1$-part. To see that Condition **(exit)** holds observe that every path from a head in the $\mathbb{G}_1$-part of $\mathbb{G}_1 ; \mathbb{G}_2$ to the exit e in $\mathbb{G}_1 ; \mathbb{G}_2$ needs to pass through the $\mathbb{G}_2$-part and hence by the definition of E gives rise to a path to an exit e in $\mathbb{G}_1$.

Angelic tests. Let $\alpha = \varphi?$ for some $\varphi \in \mathcal{F}_{\mathrm{DNNF}}$ in which e does not occur. By induction hypothesis, there is a pointed syntax graph $\mathbb{G} = (V, E, L, \Omega, v_I)$ with $\mathbb{G} \equiv \varphi$. We define a pointed syntax graph with exit $\mathbb{G}? = (V', E', L', \Omega', e, v'_I)$ such that $\mathbb{G}? \equiv \langle\varphi?\rangle e$ using the axiom $\langle\varphi?\rangle\psi \leftrightarrow \varphi \wedge \psi$.

We define $\mathbb{G}?$ by adding a new initial vertex v_I and an auxiliary vertex w', i.e., we take $V' = \{v'_I, w'\} + V$, where v'_I and w' are distinct, and $E' =$

$\{(v'_I, v_I), (v'_I, w')\} \cup E$. The labelling function is defined by $L'(v'_I) = \wedge$, $L'(w') = e$, and for all $v \in V$, $L'(v) = L(v)$. Finally, we take $\Omega' = \Omega$. Since all cycles in $\mathbb{G}?$ are already cycles in $\mathbb{G}$, it follows that $\mathbb{G}?$ satisfies Conditions **(head)**, **(leave)** and **(remain)**, since $\mathbb{G}$ does. To see that Condition **(exit)** holds observe that the new exit vertex w' is not reachable from any of the heads in the $\mathbb{G}$-part of $\mathbb{G}?$

Demonic tests. We define $\mathbb{G}!$ analogously to $\mathbb{G}?$, replacing the conjunction at v'_I with a disjunction.

Angelic iteration. Let $\alpha = \beta^*$. By induction hypothesis, there is a a pointed syntax graph with exit $\mathbb{G} = (V, E, L, \Omega, e, v_I)$ such that $\mathbb{G} \equiv \langle\beta\rangle e$. We define $\mathbb{G}^* = (V', E', L', \Omega', e, v'_I)$ using that $\langle\alpha^*\rangle e \leftrightarrow e \vee \langle\alpha\rangle\langle\alpha^*\rangle e$. We take $V' = \{v'_I, w'\} + V$ where v'_I and w' are distinct, and define the labelling by $L'(v'_I) = \vee$, $L'(w') = e$ and $L'(v) = L(v)$ for all $v \in V$. The edge relation is given by adding edges from v'_I to v_I and w' and rerouting edges in $\mathbb{G}$ with an exit as destination to have destination v'_I: $E'(v'_I) = \{v_I, w'\}$, $E(w') = \emptyset$ and for all $v \in V$: $E(v) = g[E(v)]$ where $g(v) = v'_I$ if $L(v) = e$, and $g(v) = v$ if $L(v) \neq e$. Note that all vertices that were labelled with e in $\mathbb{G}$ become unreachable from v'_I in $\mathbb{G}^*$. To define the priority function let n be the smallest odd priority that is strictly larger than any priorities occurring in $\mathbb{G}$, and define Ω' by

$$\Omega'(v') = \begin{cases} n & \text{if } v' = v'_I, \\ \Omega(v') & \text{if } v' \in V \text{ and } \Omega(v') \text{ is defined.} \end{cases}$$

This definition of $\mathbb{G}^*$ satisfies the priority condition for syntax graphs, because all simple cycles in $\mathbb{G}^*$ either stay completely inside $\mathbb{G}$ in which case they satisfy the priority condition because $\mathbb{G}$ does, or they go out of $\mathbb{G}$, in which case they pass through v'_I and hence contain a vertex in the domain of Ω'. Condition **(parity)** is satisfied at h_n because n is chosen to be odd and $L(h_n) = \vee$. For all other heads in $\mathbb{G}$ it is satisfied because $\mathbb{G}$ is a game logic graph.

To check Condition **(head)** we let $l_n = w'$ and $r_n = v_I$. Condition **(leave)** is then trivially satisfied for h_n. It is satisfied for all other heads h'_m in the V-part because either the path stays entirely in the $\mathbb{G}$-part in which case the condition is satisfied by the assumption on $\mathbb{G}$, or the cycles leaves the $\mathbb{G}$-part in which case it passes through h_n with priority n.

Condition **(remain)** is satisfied by the head h_n because by the choice of n there are no heads of higher priority than n. To see that Condition **(remain)** is satisfied by all other heads h_m, with $m < n$ in the $\mathbb{G}$-part assume for a contradiction that there is a simple path $p = h_m r_m \dots h_k$ for some $k > m$. If p stays entirely in the $\mathbb{G}$-part of $\mathbb{G}^*$ then we obtain a contradiction with the assumption that $\mathbb{G}$ is a game logic graph. So we only need to consider the case where p at one point leaves the $\mathbb{G}$-part. But by inspecting the definition of E one finds that this can only happen if in $\mathbb{G}$ there is a corresponding path $q = h_m r_m \dots v$ where $L(v) = e$, which with the exception of the end point is an initial segment of the simple path p. Hence, q is almost a simple path with the possible exception that there might be a contraction with the end point v. Because $L(h_m) \neq e$, since h_m is a head, we know that such a contraction again

leads to a path of the form $h_m r_m \dots v$. As we can repeat this contractions until we obtain a simple path we can then assume that $q = h_m r_m \dots v$ is actually a simple path in $\mathbb{G}$ where $L(v) = e$. This contradicts Condition **(exit)** for the game logic graph with exit $\mathbb{G}$.

Last, we check that Condition **(exit)** is satisfied by $\mathbb{G}^*$ with exit e. This is trivially the case for the head $h_n = v_I'$. For any other head we might reason similarly to the last part of the argument in the previous section. If there was a simple path $r_m \dots w'$ then this path must pass through v_I' and hence there would be a path in $\mathbb{G}$ of the form $h_m r_m \dots v$ in $\mathbb{G}$ with $L(v) = e$, which contradicts the Condition **(exit)** for $\mathbb{G}$.

Finally, let us make a remark about how to show that $\mathbb{G}^* \equiv \langle \beta^* \rangle e$. The argument uses the fixpoint game from Definition 9 and Lemma 5. It is similar to the reasoning about λ^* in the proof of Theorem 3 below.

Demonic iteration. We define $\mathbb{G}^\times$ similarly to $\mathbb{G}^*$, but we label the initial node v_I' with $\wedge$ instead of $\vee$, and we take n to be the least even priority that is strictly larger than any priority in $\mathbb{G}$.

Conjunction and disjunction. In the case $\varphi = \varphi_1 \wedge \varphi_2$ or $\varphi = \varphi_1 \vee \varphi_2$, we define $\mathbb{G}_1 \wedge \mathbb{G}_2$ and $\mathbb{G}_1 \vee \mathbb{G}_2$ of two pointed syntax graphs $\mathbb{G}_1$ and $\mathbb{G}_2$ basically in the same way as $\mathbb{G}_1 \cap \mathbb{G}_2$ and $\mathbb{G}_1 \cup \mathbb{G}_2$, except that we do not need to handle the exits.

Modal operators. In the case $\varphi = \langle \alpha \rangle \psi$, we obtain by induction hypothesis a pointed syntax graph $\mathbb{G}_1$ with exit e and a pointed syntax graph $\mathbb{G}_2$ such that $\mathbb{G}_1 \equiv \langle \alpha \rangle e$ and $\mathbb{G}_2 \equiv \psi$. We define the pointed syntax graph $\langle \mathbb{G}_1 \rangle \mathbb{G}_2$ similarly to the composition $\mathbb{G}_1 \,;\, \mathbb{G}_2$, but since there is no exit in $\mathbb{G}_2$ there is also no exit in $\langle \mathbb{G}_1 \rangle \mathbb{G}_2$. □

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 $\mathbb{G} \equiv \langle \delta \rangle e$.*

Proof. We prove the theorem by induction on the size of the domain of Ω.

In case $\mathrm{dom}(\Omega) = \emptyset$, this means that $\mathbb{G}$ has no cycles, and so we can do a straightforward subinduction on the edge relation (or more precisely, on the well-order consisting of the inverse of the transitive closure of the edge relation). That is, we can define, for every vertex $v \in V$ a game δ_v as follows:

$$\delta_v := \begin{cases} (L(v)?)\,;(L(v)!) & \text{if } L(v) \text{ is a literal distinct from } e \\ (p \vee \neg p)? & \text{if } L(v) = e \text{ and } p \neq e \\ \delta_{v_1} \cap \delta_{v_2} & \text{if } L(v) = \wedge \text{ and } E(v) = \{v_1, v_2\} \\ \delta_{v_1} \cup \delta_{v_2} & \text{if } L(v) = \vee \text{ and } E(v) = \{v_1, v_2\} \\ L(v); \delta_u & \text{if } L(v) \in \{g, g^d \mid g \in \mathsf{Gam}\} \text{ and } E(v) = \{u\} \end{cases}$$

It is routine to prove that this definition is correct, i.e., that $\mathbb{G}@v \equiv \langle \delta_v \rangle e$ for all $v \in V$. From this it follows that $\mathbb{G} \equiv \langle \delta_{v_I} \rangle e$

In case $\mathrm{dom}(\Omega) \neq \emptyset$ we may by assumption consider the head h with highest priority in $\mathrm{dom}(\Omega)$; let l and r be the successors of h (i.e., we omit subscripts).

Let us first note for further reference that:

$$h \text{ is not reachable from } l \text{ in } \mathbb{G}. \tag{3}$$

To see that $h = h_n$ is not reachable from $l = l_n$ in $\mathbb{G}$ assume for a contradiction that there is a path $p = l_n \dots h_n$ in $\mathbb{G}$. We can contract duplications of vertices on p until we have a simple path $p' = l_n \dots h_n$ in $\mathbb{G}$. By Condition **(leave)** it follows that $\Omega(p') > n$, which is impossible because n was chose to be the maximal priority occurring in $\mathbb{G}$.

We define the two pointed syntax graphs with exit $\mathbb{G}'$ and $\mathbb{G}''$: Let $\underline{\lambda^*}$ be a fresh atomic game, and let e' be a fresh variable.

– $\mathbb{G}' := (V, E', L', \Omega', e, v_I)$, where

$$E' := E \setminus \{(h, r)\}$$
$$L'(u) := \begin{cases} L(u) & \text{if } u \neq h \\ \underline{\lambda^*} & \text{if } u = h \end{cases}$$
$$\Omega'(u) := \begin{cases} \Omega(u) & \text{if } u \neq h \\ \uparrow & \text{if } u = h \end{cases}$$

In words, we obtain $\mathbb{G}'$ from $\mathbb{G}$ by dropping the edge from h to r, relabelling h with the dummy atomic game $\underline{\lambda^*}$, and removing it from the domain of Ω. Observe that every path that exists in $\mathbb{G}'$ also exists in $\mathbb{G}$.

We need to argue that $\mathbb{G}'$ is a game logic graph with exit.

It is not immediately obvious that $\mathbb{G}'$ satisfies the priority condition on syntax graphs because we removed the priority from the state h. This is not a problem, however, because if there was any cycle in $\mathbb{G}'$ that passes through h it would by the definition of E' need to continue to l and hence lead to a contradiction with (3).

It satisfies Condition **(parity)** because all it the remaining heads in $\mathbb{G}'$ are also heads in $\mathbb{G}$ and their label did not change.

To see that $\mathbb{G}'$ satisfies Condition **(leave)** take any simple path $p = l_m \dots h_m$ in $\mathbb{G}'$. Because $h = h_n$, where n is maximal priority in $\mathbb{G}$, we have that $m < n$. This path p also exists $\mathbb{G}$. We distinguish cases depending on whether h_n lies on p. If h_n does not lie on p then clearly $\Omega'(p) = \Omega(p)$ and $\Omega(p) > m$ because $\mathbb{G}'$ satisfies Condition **(leave)**. We can show that it is not possible that h lies on p. Assume this was the case. Then h is followed by $l = l_n$ on p, because p is also a path in $\mathbb{G}'$ and the only connection out of h in $\mathbb{G}'$ is via l. So we have a path of the form $p = l_m \dots h_n l_n \dots h_m$. Because there is an edge from h_m to l_m we can rotate this path until it is of the form $p' = l_n \dots h_m l_n \dots h_n$. By Condition **(leave)** it follows that $\Omega(p') > n$, which is not possible, because n is the maximal priority in $\mathbb{G}$.

Condition **(remain)** holds in $\mathbb{G}'$ because any path of form $h_m r_m \dots h_k$ with $k > m$ in $\mathbb{G}'$ would also exist and violate Condition **(remain)** in $\mathbb{G}$.

Similarly, Condition **(exit)** holds in $\mathbb{G}'$ because any violating path would also do so in $\mathbb{G}$.

– $\mathbb{G}'' := (V, E'', L'', \Omega'', e', r)$, where e' is a fresh propositional letter not occurring in $\mathbb{G}$ and

$$E'' := E \setminus \{(h, r), (h, l)\}$$
$$L''(u) := \begin{cases} L(u) & \text{if } u \neq h \\ e' & \text{if } u = h \end{cases}$$
$$\Omega''(u) := \begin{cases} \Omega(u) & \text{if } u \neq h \\ \uparrow & \text{if } u = h \end{cases}$$

In words, we obtain $\mathbb{G}''$ from $\mathbb{G}$ by initialising it at r, dropping the edges from h to l and r, relabelling h with a new exit variable e', and removing h from the domain of Ω.
Observe that every path that exists in $\mathbb{G}''$ also exists in $\mathbb{G}$.
We need to argue that $\mathbb{G}''$ is a game logic graph with exit. It satisfies Condition **(parity)** because all it's heads are also heads in $\mathbb{G}$ and their label did not change.
Preservation of Conditions **(parity)** and 13 from $\mathbb{G}$ to $\mathbb{G}''$ is equally trivial as in the case of $\mathbb{G}'$ discussed above.
To see that Condition **(leave)** holds in $\mathbb{G}''$ consider any path $p = l_m \dots h_m$ for some $m < n$. Note that $h = h_n$ can not lie on p. It could only be its last vertex because h has no successors in $\mathbb{G}''$ but then $h_n = h_m$ which is impossible because $n < n$ and the function h is injective. Hence $\Omega'(p) = \Omega(p)$ and $\Omega(p) > m$ because $\mathbb{G}$ satisfies Condition **(leave)**.
To see that Condition **(exit)** holds in $\mathbb{G}''$ assume for a contradiction that there was a path $p = h_m \dots v$ with $L(v) = e$ in $\mathbb{G}$. Because h_m is a head in $\mathbb{G}$ we have that $m < n$. Because $h = h_n$ is the only vertex in $\mathbb{G}$ that has the propositional letter e in $\mathbb{G}''$ it follows that p is of the form $h_m \dots h_n$ in $\mathbb{G}$. This is a contradiction to Condition **(leave)** for $\mathbb{G}$.

We make one more observations about our constructions that we need later:

$$e \text{ is not reachable from } r \text{ in } \mathbb{G}''. \tag{4}$$

To check that e is not reachable from r in $\mathbb{G}''$ assume for a contradiction that there is a path $p = r_n \dots u$ in $\mathbb{G}''$ from r_n to some vertex u that is labelled with e. This path p does not contain h_n because h_n has no successors in $\mathbb{G}''$ and h_n is distinct from the last vertex u on p because h_n it is labelled with e' in $\mathbb{G}''$ which was fresh and hence distinct from the letter e that labels u. The path $p = r_n \dots u$ then also exists in $\mathbb{G}$ where it extends to a path of the form $h_n p = h_n r_n \dots u$. This path can be contracted to a simple path that is still of this form because no contraction is possible at the initial vertex h_n since h_n does not lie on p. Thus we obtain a contradiction with Condition **(exit)** for the game logic graph $\mathbb{G}$ with exit e.
The induction hypothesis applies to both $\mathbb{G}'$ and $\mathbb{G}''$, so that:

– There is an $\mathsf{Gam} \cup \{\underline{\lambda^*}\}$-game δ_0 not containing e or e' such that $\mathbb{G}' \equiv \langle \delta_0 \rangle e$.
– There is a Gam-game λ not involving e or e' such that $\mathbb{G}'' \equiv \langle \lambda \rangle e'$

To see that δ_0 does not contain e' observe that e' is only used in the definition of $\mathbb{G}''$, where it is assumed to be fresh.

To see that λ does not contain e note that the induction hypothesis guarantees that all propositional letters that occur in δ_0 are reachable in $\mathbb{G}''$, but by (4) we have that e is not reachable from r, which is the initial vertex of $\mathbb{G}''$.

Now define

$$\delta := \delta_0[\lambda^*/\underline{\lambda^*}],$$

that is, everywhere in δ_0 we substitute the actual game λ^* for the formal atomic game $\underline{\lambda^*}$. It will be our purpose to prove that

$$\mathbb{G} \equiv \langle\delta\rangle e. \tag{5}$$

Before we set up our proof, we need some auxiliary definitions. First, a Gam-model is a game model over the set Gam of atomic games.

- Given a Gam-model $\mathbb{S} = (S, \gamma, \Upsilon)$, we define $\mathbb{S}^\bullet$ as the $\mathsf{Gam} \cup \{\underline{\lambda^*}\}$-model $(S, \gamma^\bullet, \Upsilon)$, where $\gamma^\bullet(g) := \gamma(g)$ if $g \neq \underline{\lambda^*}$ and $\gamma^\bullet(\underline{\lambda^*}) := \gamma_{\lambda^*}$, that is, $\gamma^\bullet(\underline{\lambda^*})$ is the actual interpretation of λ^* by γ.

The following claim then should be obvious (its proof is a standard inductive substitution argument).

Claim 1 For any Gam-model $\mathbb{S} = (S, \gamma, \Upsilon)$ we have that

$$\gamma^\bullet(\delta_0) = \gamma(\delta).$$

The key observation in our proof is the following claim.

Claim 2 For any Gam-model $\mathbb{S} = (S, \gamma, \Upsilon)$ and any state $s_I \in S$ we have that

$$\mathbb{S}, s_I \Vdash \mathbb{G} \quad \text{iff} \quad \mathbb{S}^\bullet, s_I \Vdash \mathbb{G}'.$$

To see how (5) follows from this, observe that for any Gam-model $\mathbb{S} = (S, \gamma, \Upsilon)$ and any state $s_I \in S$ we have

$$\begin{array}{lll} \mathbb{S}, s_I \Vdash \mathbb{G} & \text{iff} \quad \mathbb{S}^\bullet, s_I \Vdash \mathbb{G}' & \text{(Claim 2)} \\ & \text{iff} \quad \mathbb{S}^\bullet, s_I \Vdash \langle\delta_0\rangle e & \text{(induction hypothesis on } \mathbb{G}') \\ & \text{iff} \quad \mathbb{S}, s_I \Vdash \langle\delta\rangle e & \text{(Claim 1)} \end{array}$$

Hence it suffices to prove the key claim.

Proof of Claim 2. We only consider the direction from left to right, the proof for the opposite direction being simpler/similar. Let f be a positional winning strategy for Eloise in the evaluation game $\mathcal{E} := \mathcal{E}(\mathbb{G}, \mathbb{S}^\bullet)$, and suppose that (v_I, s_I) is a winning position in this game. We need to supply Eloise with a winning strategy in the game $\mathcal{E}' := \mathcal{E}(\mathbb{G}', \mathbb{S})$, starting at position (v_I, s_I). Observe that the game boards of these two games are almost the same: the only difference lies in positions of the form (h, s), since $L(h) = \vee \neq \underline{\lambda^*} = L'(h)$. For this reason, the strategy f can be used in $\mathcal{E}'$ as well, for all positions (v, s) where $v \neq s$.

Eloise's strategy in position (v_I, s) will be as follows:

1. She starts with playing the strategy f. It is easy to see that this guarantees her to win any match that does not pass through a position of the form (h, s).
2. If, on the other hand, at some stage a position of the form (h, s) is reached, she continues as follows. Let $U \subseteq S$ be the set of states t such that $(h, t) \in \mathrm{Win}_\exists(\mathcal{E})$ and at this position (h, t) Eloise's strategy f picks the left successor of h:
$$U := \{t \in S \mid (h, t) \in \mathrm{Win}_\exists(\mathcal{E}) \text{ and } f(h, t) = (l, t)\}.$$
We claim that U is a legitimate move for Eloise in $\mathcal{E}'$ at position (h, s):
$$U \in \gamma^\bullet(\underline{\lambda^*})(s), \tag{6}$$
and let Eloise play this move indeed.
3. Suppose that Abelard responds to this move by picking $t \in U$, so that (l, t) is the next position in the $\mathcal{E}'$-match. Then by definition of U we have that $(h, t) \in \mathrm{Win}_\exists(\mathcal{E})$ and Eloise's winning strategy f in $\mathcal{E}$ picks the position (l, t) at (h, t). This means in particular that (l, t) is a winning position for Eloise in $\mathcal{E}$.
Since h is the only vertex at which $\mathbb{G}$ and $\mathbb{G}'$ differ, but from (3) we know that h is not reachable from l in $\mathbb{G}$, it follows that the the syntax graphs $\mathbb{G}@l$ and $\mathbb{G}'@l$ are identical. This means that from this moment on, Eloise can resume playing her winning $\mathcal{E}$-strategy f in $\mathcal{E}'$ again, and it is easy to see that any resulting full $\mathcal{E}'$-match is a win for Eloise.

It is left to prove (6). By definition of $\gamma^\bullet$ this means that we need to show that $U \in \gamma_{\lambda^*}(s)$, and by Lemma 5 it suffices to show that $s \in \mathrm{Win}_\exists(\mathcal{F})$, where $\mathcal{F} := \mathcal{F}(\lambda^*, \mathbb{S}, U)$ is the game defined in Definition 9. In other words, we have to supply Eloise with a winning strategy for position s in $\mathcal{F}$.

To define this strategy $\overline{f}$, take an arbitrary point $t \in S$, and make the following case distinction:

- If $(h, t) \notin \mathrm{Win}_\exists(\mathcal{E})$, Eloise makes a random move (we will make sure that this situation never occurs).
- If $(h, t) \in \mathrm{Win}_\exists(\mathcal{E})$ and $t \in U$, $\overline{f}$ picks the empty set as a move for Eloise. Clearly this is legitimate and it causes an immediate win for Eloise in $\mathcal{F}$.
- If $(h, t) \in \mathrm{Win}_\exists(\mathcal{E})$ but $t \notin U$, then by definition of U Eloise's winning strategy f picks the right successor r of h at the position (h, t) in $\mathcal{E}$. This means that $(r, t) \in \mathrm{Win}_\exists(\mathcal{E})$. Define W_t to be the set of points $u \in S$ for which there is a partial f-guided $\mathcal{E}$-match $\Pi = (v_0, s_0) \cdots (v_n, s_n)$ such that $(v_0, s_0) = (r, t)$, $(v_n, s_n) = (h, u)$ and $v_i \neq h$ for all $0 < i < n$. It should then be obvious that
$$W_t \subseteq \{u \in S \mid (h, u) \in \mathrm{Win}_\exists(\mathcal{E})\}. \tag{7}$$
We define $\overline{f}(t) := W_t$, that is we let W_t be Eloise's move at position t in $\mathcal{F}$, and first show its legitimacy:
$$W_t \in \gamma_\lambda(t). \tag{8}$$

In order to prove (7) we turn to the evaluation game $\mathcal{E}'' := \mathcal{E}(\mathbb{G}'', \mathbb{S}^\circ)$, where $\mathbb{S}^\circ := (S, \gamma, \Upsilon[e' \mapsto W_t])$. Here we use $\Upsilon[e' \mapsto W_t]$ to denote the valuation that is just like Υ but maps the propositional letter e' to the set $W_t \subseteq S$. It follows from the inductive hypothesis on $\mathbb{G}''$ that

$$(r, t) \in \mathrm{Win}_\exists(\mathcal{E}'') \quad \text{iff} \quad \mathbb{S}^\circ, t \Vdash \langle\lambda\rangle e',$$

so that by definition of $\mathbb{S}^\circ$ we find that

$$(r, t) \in \mathrm{Win}_\exists(\mathcal{E}'') \quad \text{iff} \quad W_t \in \gamma_\lambda(t).$$

Hence, in order to prove (8) it suffices to show that

$$(r, t) \in \mathrm{Win}_\exists(\mathcal{E}''), \tag{9}$$

that is, we have to supply Eloise with a winning strategy in the game $\mathcal{E}''$ at position (r, t). However, given the similarities between $\mathbb{G}$ and $\mathbb{G}''$ and between $\mathbb{S}$ and $\mathbb{S}^\circ$, she can simply use her $\mathcal{E}$-strategy f once more. Consider a full $\mathcal{E}''$-match Π where she plays like this. If Π does not pass through a position of the form (h, u) then this match is also an $\mathcal{E}$-match, and it is easy to see that Eloise wins. On the other hand, if Π does pass through such a position (h, u), this can only mean that in $\mathcal{E}''$ this is the final position of Π. It then follows by definition of W_t that $u \in W_t$, so that $u \in \Upsilon[e' \mapsto W_f](e')$, and again Eloise is the winner of the $\mathcal{E}''$-match Π. This proves (9).

It remains to prove that this strategy $\overline{f}$ is winning for Eloise at position s in $\mathcal{F}$. So consider a full $\overline{f}$-guided match Σ in $\mathcal{F}$ starting at s. It follows from (7) that all positions t reached in this match are such that $(h, t) \in \mathrm{Win}_\exists(\mathcal{E})$, so that we can make the following case distinction.

If some position t of Σ belongs to U, then $\overline{f}$ picks the empty set and Eloise wins immediately. If, on the other hand, all positions of Σ are such that $(h, t) \in \mathrm{Win}_\exists(\mathcal{E})$ but $t \notin U$, then we only have to worry that Σ might be infinite (since Eloise has a legitimate move W_t for all t on Σ, Σ being finite means that Abelard got stuck). So assume for contradiction that Σ is of the form $\Sigma = t_0 W_0 t_1 W_1 \cdots$ with $t_0 = s$. Given the definition of $\overline{f}$ it is easy to see that for every $n \in \omega$ there is a partial f-guided $\mathcal{E}$-match $\Sigma_n = (h, t_n) \cdots (h, t_{n+1})$. But then we could glue these partial matches together, forming one infinite $\mathcal{E}$-match $\Sigma' = (h, t_0) \cdots (h, t_1) \cdots$. This provides the desired contradiction, since this Σ' would be a loss for Eloise since it passes h infinitely often, whereas at the same time it is guided by Eloise's winning strategy f. In other words, $\overline{f}$ is a winning strategy for Eloise in the game $\mathcal{F}$ starting from s. □

This finishes the proof of Theorem 3. □