Complete Game Logic with Sabotage

Noah Abou El Wafa
Karlsruhe Institute of Technology
Karlsruhe, Germany
Carnegie Mellon University
Pittsburgh, USA
noah.abouelwafa@kit.edu

André Platzer
Karlsruhe Institute of Technology
Karlsruhe, Germany
Carnegie Mellon University
Pittsburgh, USA
platzer@kit.edu

ABSTRACT

Game logic with sabotage ($\mathrm{GL_s}$) is introduced as a simple and natural extension of Parikh's game logic with a single additional primitive, which allows players to lay traps for the opponent. $\mathrm{GL_s}$ can be used to model infinite sabotage games, in which players can change the rules during game play. In contrast to game logic, which is strictly less expressive, $\mathrm{GL_s}$ is exactly as expressive as the modal μ-calculus. This reveals a close connection between the entangled nested recursion inherent in modal fixpoint logics and adversarial dynamic rule changes characteristic for sabotage games. A natural Hilbert-style proof calculus for $\mathrm{GL_s}$ is presented and proved complete using syntactic equiexpressiveness reductions. The completeness of a simple extension of Parikh's calculus for game logic follows.

CCS CONCEPTS

• **Theory of computation** → **Logic**; **Proof theory**; **Modal and temporal logics**.

KEYWORDS

game logic, μ-calculus, proof theory, completeness, expressiveness, sabotage games

ACM Reference Format:
Noah Abou El Wafa and André Platzer. 2024. Complete Game Logic with Sabotage. In *39th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '24), July 8–11, 2024, Tallinn, Estonia.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3661814.3662121

1 INTRODUCTION

Games such as the Ehrenfeucht-Fraïssé game are invaluable tools in the study of logic [18] and some deep results about logic can be proved with the help of games [3, 26]. Logic can even be given meaning via games in the form of game-theoretical semantics [27].

Dually, logical methods are frequently used to study games [15]. Logic and games meet most directly in logics specifically designed for the study of games, such as game logic (GL) due to Parikh [35], which allows reasoning about the existence of winning strategies in a game. This requires giving exact meaning to general games, a nontrivial task for games that are not limited to a fixed number of rounds. Nested alternating least and greatest fixpoints can provide the correct denotational semantics for games, when they are used to reflect the alternating responsibilities of the respective players at their decision points in the dynamic games [35].

Fixpoints also play an important role in logic, for example in modal fixpoint logics such as the modal μ-calculus (L_μ). L_μ is where logic, games and fixpoints begin to converge. In fact game logic can be expressed in the modal μ-calculus using alternating fixpoint formulas to directly capture the semantics of alternating game play. However this first encounter is imperfect. After 20 years it was shown that GL is in fact strictly less expressive than L_μ [9].

The purpose of this paper is to remedy this situation by unifying the three fundamental concepts of logic, games, and fixpoints in a small and natural completion of game logic, which is shown to be *equivalent* to the fixpoint logic L_μ and to have a complete proof calculus. This identification of fixpoints with games *eliminates* the difference between interactive game play and alternating fixpoints! The key insight behind this paper is that, because game logic can already express sufficient adversarial dynamics to express the alternating fixpoints of L_μ and is merely lacking a suitable way of referring to fixpoints by their respective fixpoint variables, this deficiency can be solved in a parsimonious and purely game-theoretic way. This is done in *game logic with sabotage* ($\mathrm{GL_s}$), a new extension of GL. In game logic with sabotage, reference can be expressed, not through the unstructured use of fixpoint variables as in the modal μ-calculus, but by using a simple game operator $\sim a$ that *changes* the rules of subsequent game play. Playing the game $\sim a$ has the effect that the game a is reserved exclusively for one player in the future. This can be used to change the rules of a game dynamically according to rules that are explicit in the original game. This simple and natural mechanism of imperative game play is *expressively equivalent* to the functional mechanism of unstructured nested named (co)recursion with the fixpoint variables in the alternating fixpoints of L_μ. The role the sabotage $\sim a$ plays in establishing the equiexpressiveness reveals an interesting connection between games with sabotage and the nesting of fixpoints in the modal μ-calculus which have previously been studied separately.

Formulas of the modal μ-calculus are frequently easiest to understand through their corresponding validity or model-checking games [34]. This is complicated by the unstructured goto-like action a fixpoint variable induces. Game logic with sabotage avoids this problem, as $\mathrm{GL_s}$ formulas describe two-player games built up from simple connectives and, instead of fixpoint variables, players only need to consider the previously committed acts of sabotage, making game logic with sabotage a very intuitive logic with very high expressive power. By the equivalence of L_μ and $\mathrm{GL_s}$, many desirable properties of the modal μ-calculus, such as decidability and the small model property, transfer to game logic with sabotage for free. And as L_μ is precisely the bisimulation-invariant fragment of monadic second-order logic [28], this expressive completeness also holds for $\mathsf{GL_s}$. Moreover, completeness of an axiomatization of $\mathsf{GL_s}$ can be obtained through the translation. This is in contrast to the original axiomatization for game logic, for which completeness is still open after four decades [29] despite considerable attention [22]. $\mathsf{GL_s}$ promises to be a useful tool for understanding GL. For instance the completeness of $\mathsf{GL_s}$ yields a *completion* of Parikh's proof calculus for GL, which axiomatizes GL completely. To the best of our knowledge this is the only complete proof calculus for game logic to date. The embedding from game logic with sabotage to the modal μ-calculus also suggests that the same property can be expressed significantly more concisely in game logic with sabotage than in the modal μ-calculus showing that, while theoretically equivalent, they are practically very different.

The equiexpressiveness proof of L_μ and GL is modular, which simplifies the proofs and illuminates the differences between the modes of expression in game and fixpoint logics. The proof builds on a homomorphic translation between fixpoint logic and an extension of GL with the recursion from L_μ transferred to games (recursive game logic RGL). However owing to the sequential composition of games, the expressive power of RGL is the same as the highly expressive fixpoint logic with chop (FLC), which is far beyond L_μ. The next key step in the proof is the identification of the fragment of RGL formulas corresponding to L_μ formulas. Using sabotage, this fragment can be captured merely with simpler iteration games and no general form of recursion. These semantic translations are then used to transform proofs from fixpoint logics to game logics and proofs with recursive games to proofs with iteration games and sabotage. The correctness of these transformations is shown by proving the *syntactical provability* of the correctness of the translations in the proof calculi and in this way lifting *semantic equiexpressiveness* proofs to *syntactic completeness* proofs.

The contributions of this paper are fourfold. Firstly, $\mathsf{GL_s}$, a new minimal, natural, concise and intuitive extension of game logic is introduced. Secondly, it is shown that $\mathsf{GL_s}$ is expressively exactly as powerful as the modal μ-calculus and, consequently, many desirable logical properties of L_μ transfer to $\mathsf{GL_s}$. Thirdly, a sound proof calculus for $\mathsf{GL_s}$ is presented, proved complete, and completeness is transferred to obtain a complete extension of Parikh's GL calculus. Fourthly, RGL is introduced and proved equiexpressive to FLC.

Outline. Basic definitions are recalled in Section 2. Section 3 introduces game logic with sabotage ($\mathsf{GL_s}$) and another extension of GL, called recursive game logic (RGL), that will play an intermediary role in translating between the modal μ-calculus and game logic with sabotage and is of independent interest. Section 4 briefly recalls the definitions of two modal fixpoint logics: the modal μ-calculus and fixpoint logic with chop. In Section 5, the expressive power of fragments of recursive game logic is compared to modal fixpoint logics, and the equiexpressiveness of recursive game logic and fixpoint logic with chop, as well as of the modal μ-calculus and game logic with sabotage, are shown. In Section 6, an axiomatization for game logic with sabotage is presented and its completeness is proved by reduction to completeness of the modal μ-calculus. The completeness of a simple extension of Parikh's GL calculus follows. All omitted proofs are in the full version [2].

Related Work. Sabotage games have been considered to model algorithms under adversarial conditions and in learning [25, 44]. Previous work using modal logic in Sabotage Modal Logic (SML) [7, 44] differs from game logic with sabotage. Unlike in $\mathsf{GL_s}$, where sabotage is modelled as changing the meaning of the game described syntactically, SML describes sabotage as changing the structure of interpretation. The sabotage μ-calculus was investigated for modelling infinite sabotage games [41]. In contrast to $\mathsf{GL_s}$, satisfiability for SML is undecidable and lacks the finite model property [31].

Applications of game logic are reported elsewhere [36–39]. The relation of games, game logic, fixpoints and the modal μ-calculus has been considered extensively [9, 12, 17, 19, 24, 28, 34, 35, 42]. The modal μ-calculus and its relation to model checking is well-studied [10, 11, 20, 40]. Completeness for game logic was conjectured [35]. A completeness proof based on a cut-free calculus for L_μ [4] was suggested for game logic [22]. It was recently shown not to work [29]. Relative completeness of differential game logic, an extension of game logic with differential equations, has already been proved [37]. For first-order versions of game logic and the modal μ-calculus equiexpressiveness and relative completeness were shown [1].

2 PRELIMINARIES

Effectivity Functions. A monotone function $w : \mathcal{P}(X) \to \mathcal{P}(X)$ is called an *effectivity function* [21]. The semantics of a game will be given in terms of an effectivity function, where $w(A)$ denotes the winning region, i.e. the set of all states from which a given player can win into the goal region A. Such functions are naturally monotone, as any point in the winning region for a goal A is also in the winning region for the larger goal $B \supseteq A$. Let $\mathcal{W}(X)$ be the set of all effectivity functions ordered by point-wise inclusion, i.e. $w \subseteq u$ if $w(A) \subseteq u(A)$ for all $A \subseteq X$.

Definition 2.1. (1) Given a set $A \subseteq X$, the intersection effectivity function is $A_?(B) = A \cap B$ and the constant effectivity function is $c_A(B) = A$ for all $B \subseteq X$.
(2) For an effectivity function $w \in \mathcal{W}(X)$ its dual is $w^{\mathrm{d}}(A) = X \setminus w(X \setminus A)$ for all $A \subseteq X$.
(3) An effectivity function $w \in \mathcal{W}(X)$ is *relational* if there is a relation $R \subseteq X \times X$ on X such that

$$w(A) = R \circ A = \{r : \exists s \in A \;\; rRs\}.$$

(4) For an effectivity function $w \in \mathcal{W}(X)$ define

$$\mu A.w(A) = \bigcap \{A \in \mathcal{P}(X) : w(A) \subseteq A\}$$

(5) For a transformation $F : \mathcal{W}(X) \to \mathcal{W}(X)$ on $\mathcal{W}(X)$ define

$$\mu w.F(w) = \bigcap \{w \in \mathcal{W}(X) : F(w) \subseteq w\}$$

As usual $\mu A.w(A)$ and $\mu q.F(q)$ are the least fixpoints of w and F, respectively, provided F is monotone [5]. In the sequel it will be necessary to work with fixpoints of monotone functions on $\mathcal{P}(X)$ and also with fixpoints of monotone functions on $\mathcal{W}(X)$. Under some conditions the latter can be viewed pointwise as the former.

Lemma 2.2. *Suppose $F : \mathcal{W}(X) \to \mathcal{W}(X)$ is monotone and $F(u)(A) = F(c_{u(A)})(A)$ for all $u \in \mathcal{W}(X)$ and all $A \subseteq X$, then*
(1) $(\mu u.F(u))(A) = \mu B.(F(c_B)(A))$
(2) $(\nu u.F(u))(A) = \nu B.(F(c_B)(A))$

Neighbourhood and Kripke Structures. Throughout the paper fix countably infinite sets $\mathbb{P}$ of propositional constants, $\mathbb{V}$ of fixpoint variables, and $\mathbb{G}$ of atomic games, respectively.

Game logic and the modal μ-calculus can be interpreted over coalgebraic structures [16]. While the modal μ-calculus is commonly interpreted over Kripke structures, game logic was originally interpreted over the more general class of neighbourhood models [33]. The results in this paper hold for both classes of models equally.

Definition 2.3. A *monotone neighbourhood structure* is a triple $\mathcal{N}$ consisting of a set of states $|\mathcal{N}|$ and functions

$$\mathcal{N}(\cdot) : \mathbb{P} \to \mathcal{P}(|\mathcal{N}|) \qquad \mathcal{N}(\cdot) : \mathbb{G} \to \mathcal{W}(|\mathcal{N}|)$$

assigning a valuation $\mathcal{N}(P)$ to every atomic proposition $P \in \mathbb{P}$ and an effectivity function $\mathcal{N}(a)$ to every atomic game $a \in \mathbb{G}$. The structure $\mathcal{N}$ is a *Kripke structure* iff each $\mathcal{N}(a)$ is relational.

3 EXTENSIONS OF GAME LOGIC

3.1 Game Logic with Sabotage

Game logic with sabotage ($\mathrm{GL_s}$) is an extension of game logic defined by adding the sabotage action games $\sim a$. Formulas and games of $\mathrm{GL_s}$ are given by the following grammar:

$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \psi \mid \langle\alpha\rangle\varphi$$
$$\alpha ::= a \mid \sim a \mid \alpha^{\mathrm{d}} \mid ?\varphi \mid \alpha \cup \beta \mid \alpha;\beta \mid \alpha^*$$

where $P \in \mathbb{P}$ and $a \in \mathbb{G}$. Syntactically, game logic (GL) is the fragment without sabotage actions $\sim a$.

The formula $\langle\alpha\rangle\varphi$ expresses that player Angel has a winning strategy in the game α to reach one of the states in which φ is true. The test game $?\varphi$ is lost prematurely by Angel unless the formula φ is true in the current state. The choice game $\alpha \cup \beta$ allows Angel to choose between playing α or β. To play the sequential game $\alpha;\beta$ is to play β after α (unless a player already lost the play of α). The repetition game α^* allows Angel to decide after each completed round of α whether to stop playing α or to repeat α. The dual operator $^{\mathrm{d}}$ flips games around to the opponent's perspective and is used to define the Demonic sabotage, test, choice, and repetition.

The additional primitive $\sim a$ is the sabotage action. When $\sim a$ is played, Angel sabotages the atomic game a. It has the effect that, should Demon try to play a at any point in the subsequent game (by reaching a^{d}), he loses the game prematurely. But if Angel plays a after it has been sabotaged by $\sim a$ it will simply be skipped and the game continues in the same state. The atomic game a remains sabotaged throughout the game. The only thing that may change is the last player to commit the sabotage action. If Demon at some later point manages to play a sabotage action of the same atomic game $\sim a^{\mathrm{d}}$, the game will then be sabotaged in the dual way. That means the next time Angel plays a she loses the game immediately. Once a sabotage action has been played for an atomic game it can only change hands, but will not be played normally again.

Sabotage may be viewed as setting a trap for the opponent. After playing the sabotage action game $\sim a$ the *atomic game a* becomes a trap for Demon that Angel can simply evade. Viewed differently, if Angel plays $\sim a$, she claims the atomic game a for herself. The opponent is not allowed to play a and would forfeit the game by trying, unless he first claims a for himself. Playing $\sim a^{\mathrm{d}}$ dually means the game a belongs to Demon until it returns to Angel. The effect of the claim is that the subsequent rules for playing a and a^{d} *change* as in Table 1.

Table 1: Effect of Rule Changes by $\sim a$ and $\sim a^{\mathrm{d}}$

Owner of a	Rules for a	Rules for a^{d}
Neither $\emptyset$	a played normally	a^{d} played normally
Angel $\diamond$	a skipped	Angel wins a^{d}
Demon $\square$	Demon wins a	a^{d} skipped

Abbreviations and Conventions. The dual game connectives for Demon's choice, test and repetition are defined as in game logic. That is $\alpha \cap \beta$ abbreviates $(\alpha^{\mathrm{d}} \cup \beta^{\mathrm{d}})^{\mathrm{d}}$, which leaves the choice of whether to play α or β to Demon. Analogously $¿\varphi$ represents a test Demon needs to pass $(?\varphi)^{\mathrm{d}}$ and $\alpha^{\times}$ is the repetition $(\alpha^{\mathrm{d}^*})^{\mathrm{d}}$ controlled by Demon. The box modality $[\alpha]\varphi \equiv \neg\langle\alpha\rangle\neg\varphi$ and the propositional connectives $\wedge, \rightarrow, \leftrightarrow$ and $\top, \bot$ are defined as usual. By convention sequential composition ; binds stronger than choice $\cup, \cap$ and conjunction and disjunction bind stronger than implication.

Example: Crossing Bridges. A simple bridge crossing game illustrates sabotage. Suppose Angel and Demon begin on the bank of a river with two different bridges a and b. Demon begins the game by choosing one of the two bridges and sabotaging it. Subsequently Angel chooses which one of the two bridges to cross. If Angel chooses to cross the bridge that Demon has sabotaged, she loses. Angel knows which bridge has been sabotaged. The $\mathrm{GL_s}$ game

$$(\sim a^{\mathrm{d}} \cap \sim b^{\mathrm{d}}); (a \cup b)$$

captures this game. If Angel's objective is merely to cross any bridge, she has a winning strategy. That is what it means for the formula

$$\langle(\sim a^{\mathrm{d}} \cap \sim b^{\mathrm{d}}); (a \cup b)\rangle\top$$

to be satisfied in the current state. If Angel wants to ensure that she reaches a point on the other side in which P holds and only a leads to such a point, then Demon has a winning strategy to thwart Angel. The following formula says exactly that:

$$(\langle a\rangle P \wedge \langle b\rangle\neg P) \rightarrow [(\sim a^{\mathrm{d}} \cap \sim b^{\mathrm{d}}); (a \cup b)]\neg P$$

Infinite Plays. In game logic with sabotage just as in game logic it is possible for the two players to play infinitely long. Angel could for example choose to repeat the atomic game a in $(a)^*$ every time. In this case the game never comes to an end. This strategy cannot be winning, since the semantics of infinite plays is defined so that the player who causes the game to be infinite (by repeating a subgame infinitely often that is not contained in another subgame that is repeated infinitely often) loses.

Example: The Euler Path Game. An example to illustrate the potential concision of game logic with sabotage compared to the modal μ-calculus is the Euler path game, which captures the existence of an Euler path in a graph. A related, but slightly simpler problem is defining, with a formula ψ, the states corresponding to nodes from which there is a path reaching a state in which P is true using each edge a_i *at most once.* The formula

$$\langle(a_1; \sim a_1^{\mathrm{d}} \cup \ldots \cup a_k; \sim a_1^{\mathrm{d}})^*\rangle P$$

captures this. It can be viewed as modelling a game in which Angel, trying to make her way across several bridges connecting a town, can choose to take any available link a_i she likes. However upon her crossing Demon sabotages the bridge ($\sim a_i^{\mathrm{d}}$), so that the next time Angel attempts to cross (a_i), the bridge collapses and she loses. Hence Angel wins exactly if she can find a way to a place where P is true without ever crossing the same bridge twice.

For the Euler path game the additional restriction that every bridge must be taken once needs to be added. This can be done with a second sabotage mechanism as follows.

$$\langle \sim b_1^{\mathrm{d}}; \dots \sim b_k^{\mathrm{d}}; (a_1; \sim a_1^{\mathrm{d}}; \sim b_1 \cup \dots \cup a_k; \sim a_k^{\mathrm{d}}; \sim b_k)^*; b_1; \dots; b_k \rangle \top$$

The additional atomic games b_i are initially sabotaged by Demon and ensure that Angel cannot ultimately pass the checkpoints $b_1; \dots; b_k$ unless she has undone each of the initial Demonic sabotage actions ($\sim b_i^{\mathrm{d}}$), which she can do only by crossing all bridges.

The Euler path example illustrates that $\mathsf{GL_s}$ and L_μ are two substantially different specification languages. In L_μ the shortest known formula to express the Euler path property simply lists all potential Euler paths. (Here S_k is the symmetric group of degree k.)

$$\bigvee_{\sigma \in S_k} \langle a_{\sigma(1)} \rangle \langle a_{\sigma(2)} \rangle \dots \langle a_{\sigma(k)} \rangle \top$$

This also indicates that sabotage descriptions can be significantly more concise than the corresponding L_μ versions. In the example the length of the Euler path GL formula is *linear* in the size of the graph k, whereas the length of the L_μ formula is *factorial* in k.

3.2 Recursive Game Logic

The equiexpressiveness of game logic with sabotage and the modal μ-calculus will be proved in a modular way. Unlike GL the modal μ-calculus allows for nested recursive *reference* and this gap between rlGL and L_μ will be closed first by introducing an extension of game logic that extends it with an analogous form of *game* reference. However reference added to GL goes *beyond* the modal μ-calculus and the expressive power is equivalent to fixpoint logic with chop (see Section 4)! For a schematic overview of how the logics introduced in the sequel relate to one another see Figure 1 in Section 5.

Recursive game logic (RGL), an extension of game logic, admits arbitrarily recursive and corecursive games. This increases the expressive power of game logic greatly and serves as the technical intermediary connecting game logic with sabotage to the modal μ-calculus. The syntax of recursive game logic (RGL) is defined by the following grammar

$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \psi \mid \langle\alpha\rangle\varphi$$
$$\alpha ::= a \mid x \mid \alpha^{\mathrm{d}} \mid ?\varphi \mid \alpha \cup \beta \mid \alpha;\beta \mid \mathrm{r}x.\alpha$$

where $P \in \mathbb{P}$, $a \in \mathbb{G}$ and $x \in \mathbb{V}$. Additionally the restriction is placed on games that games $?\varphi$ are only allowed for *closed* formulas φ and that free variables x can only be bound by $\mathrm{r}x.\alpha$ if x appears only in the scope of an even number of dual operators $\cdot^{\mathrm{d}}$ in α. Syntactically the only difference between recursive game logic and game logic is that repetition games α^* have been replaced by *recursive subgames* of the form $\mathrm{r}x.\alpha$ and that games x to recursively call a subgame have been added.

Intuitively a recursive game $\mathrm{r}x.\alpha$ is played just like α until the subgame is called again because x is encountered. In this case the game is interrupted and the players begin another recursive subgame of $\mathrm{r}x.\alpha$. Once the players finish playing this subgame, they continue to play the interrupted game in the state they reached,

3.2.1 Notation. The formula (game) obtained from φ (α) by replacing every *free* appearance of x by the game β is denoted $\varphi\frac{\beta}{x}$ ($\alpha\frac{\beta}{x}$). The abbreviations for the usual propositional symbols and the Demonic connectives are defined just as in $\mathsf{GL_s}$. The dual *corecursive* version $\mathrm{ɹ}x.\alpha$ of a recursive subgame is defined as $(\mathrm{r}x.\alpha^{\mathrm{d}}\frac{x^{\mathrm{d}}}{x})^{\mathrm{d}}$. This game is played similarly to $\mathrm{r}x.\alpha$. The only difference is which of the players is held responsible if the game is played infinitely long. If the largest subgame that is repeated infinitely often during a play is a recursive game of the form $\mathrm{r}x.\alpha$, then Angel loses the game. If the largest such game is of the form $\mathrm{ɹ}x.\alpha$, then Demon loses.

3.2.2 Examples. An example of a game with recursion is

$$\mathrm{r}x.(c \cup a; x; b).$$

The recursive subgame *declaration* does not require either player to make a move. The first move of the game is Angel's and she gets to choose whether to play c or $a; x; b$. In the first case c is played and the game ends. If she chooses to play $a; x; b$, then, after a, the game is continued by playing the game referred to by x, which is $\mathrm{r}x.(c \cup a; x; b)$ again. However after this game is completed, b will be played still before the full game finally ends. A run of the game behaves like $a^n; c; b^n$ for some $n \geq 0$, which cannot be described in Parikh's game logic, which lacks the facilities to retain the number of games b that still have to be played after Angel chooses to stop the repetition by playing c.

3.2.3 Game Logic. Syntactically, Parikh's game logic (GL) [35] is the fragment of recursive game logic with restricted repetition. Instead of arbitrary (co)recursive games $\mathrm{r}x.\alpha$ and $\mathrm{ɹ}x.\alpha$ only iteration games of the form a^* are permitted. Iteration games are defined by

$$\alpha^* \equiv \mathrm{r}x.(\alpha; x \cup ?\top).$$

Here Angel chooses whether to play α and then continue with α^* or whether to end the game in the current state. The Demonic iteration game is $\alpha^\times \equiv (\alpha^{\mathrm{d}})^{*\mathrm{d}}$. The semantics of game logic with sabotage will be defined so that it agrees with the usual semantics, i.e. game logic with sabotage is a genuine extension of game logic.

3.3 Semantics of Game Logics

A denotational semantics for recursive game logic and game logic with sabotage can be defined in a simple and compositional way. Superficially both semantics are different from the usual semantics of game logic. However it will be shown that for GL formulas the semantics of RGL and $\mathsf{GL_s}$ agree with the standard GL semantics.

3.3.1 Semantics of Game Logic with Sabotage. The semantics of a $\mathsf{GL_s}$ game depends on the sabotage actions that players have played in the run of the game so far. To keep track of these, games and formulas of game logic with sabotage must be evaluated in a context. A *context* is a function $c : \mathbb{G} \to \{\emptyset, \diamond, \square\}$ indicating which player has last sabotaged an atomic game (recall Table 1). All contexts are assumed to have finite support, that is $c(a) = \emptyset$ for all but finitely many a. Let C be the set of all contexts and let $c_\emptyset$ be the constant context without any atomic games sabotaged, i.e. $c_\emptyset(a) = \emptyset$ for all a. For any set $U \subseteq |\mathcal{N}| \times C$ and any context $c \in C$ let $U\restriction c = \{\omega : (\omega, c) \in U\}$ be the projection on $|\mathcal{N}|$. And for $a \in \mathbb{G}$ the $\diamond$-sabotage winning region of $A \subseteq |\mathcal{N}| \times C$ is

$$A_a^\diamond = \{(\omega, c) : (\omega, c\tfrac{\diamond}{a}) \in A\}.$$

To interpret an atomic game a, it is necessary to consider the context in which it is played. If one of the players has already sabotaged a, the normal rules no longer apply. To take this into account the semantics $\mathcal{N}(a) \in \mathcal{W}(|\mathcal{N}|)$ is lifted to $\widehat{\mathcal{N}}(a) \in \mathcal{W}(|\mathcal{N}| \times C)$. For every $U \subseteq |\mathcal{N}| \times C$ the lifting is defined by $(\omega, c) \in \widehat{\mathcal{N}}(a)(U)$ iff

(1) $c(a) = \emptyset$ and $\omega \in \mathcal{N}(a)(U\restriction c)$ or
(2) $c(a) = \diamond$ and $(\omega, c) \in U$

If a has never been sabotaged (i.e. $c(a) = \emptyset$), Angel can win game a from a position ω in context c into the set U exactly if she can win a game of a played according to the usual rules into $U\restriction c$. If a belongs to Angel (i.e. $c(a) = \diamond$), she can also win exactly if the current state ω and context c are already in U. However if a belongs to Demon (i.e. $c(a) = \square$), Angel has already lost. This formalizes the effect of rule change as described in Table 1. For any context c dual context $\overline{c}$ turns Angelic sabotages into Demonic sabotages and vice versa:

$$\overline{c}(a) = \begin{cases} \emptyset & \text{if } c(a) = \emptyset \\ \square & \text{if } c(a) = \diamond \\ \diamond & \text{if } c(a) = \square \end{cases}$$

For a set $A \subseteq W \times C$ the *sabotage complement* is $A^{\mathsf{C}} = \{(\omega, c) : (\omega, \overline{c}) \notin A\}$ and for a function $w \in \mathcal{W}(W \times C)$ the *sabotage dual* is

$$w^{\mathsf{D}}(A) = w(A^{\mathsf{C}})^{\mathsf{C}}.$$

The sabotage dual extends the notion of the ordinary dual to sabotage games. For the lifted semantics $(\omega, c) \in (\widehat{\mathcal{N}}(a))^{\mathsf{D}}(U)$ iff

(1) $c(a) = \emptyset$ and $\omega \in (\mathcal{N}(a))^{\mathrm{d}}(U\restriction c)$ or
(2) $c(a) = \diamond$ or
(3) $c(a) = \square$ and $(\omega, c) \in U$

The semantics of formulas and games of game logic with sabotage with respect to a monotone neighbourhood structure is defined by mutual induction on the definition of formulas and games of game logic with sabotage.[1]

Definition 3.1. For any monotone neighbourhood structure $\mathcal{N}$ the *semantics* of a $\mathsf{GL_s}$ formula φ is a set $\mathcal{N}[\![\varphi]\!]_{\mathsf{s}} \in \mathcal{P}(|\mathcal{N}| \times C)$

$$\mathcal{N}[\![P]\!]_{\mathsf{s}} = \mathcal{N}(P) \times C \qquad \mathcal{N}[\![\langle\alpha\rangle\varphi]\!]_{\mathsf{s}} = \mathcal{N}[\![\alpha]\!]_{\mathsf{s}}(\mathcal{N}[\![\varphi]\!]_{\mathsf{s}})$$
$$\mathcal{N}[\![\neg\varphi]\!]_{\mathsf{s}} = \mathcal{N}[\![\varphi]\!]_{\mathsf{s}}^{\mathsf{C}} \qquad \mathcal{N}[\![\varphi \vee \psi]\!]_{\mathsf{s}} = \mathcal{N}[\![\varphi]\!]_{\mathsf{s}} \cup \mathcal{N}[\![\psi]\!]_{\mathsf{s}}$$

[1]Unlike for sabotage modal logic [6] the semantics is not defined in terms of a changing model. Instead the state space is enlarged to contain the states of the structure and independently keep track of the sabotage actions played. The definition is similar in spirit to the modified semantics for the sabotage μ-calculus [6]. Unlike in the definition of the modal μ-calculus augmented with sabotage [41] the traps set persist throughout multiple repetitions of game α^* instead of resetting without cause.

and of a $\mathsf{GL_s}$ game α is an effectivity function $\mathcal{N}[\![\alpha]\!]_{\mathsf{s}} \in \mathcal{W}(|\mathcal{N}| \times C)$

$$\mathcal{N}[\![a]\!]_{\mathsf{s}} = \widehat{\mathcal{N}}(a) \qquad \mathcal{N}[\![?\varphi]\!]_{\mathsf{s}}(A) = \mathcal{N}[\![\varphi]\!]_{\mathsf{s}} \cap A$$
$$\mathcal{N}[\![{\sim}a]\!]_{\mathsf{s}}(A) = A_a^\diamond \qquad \mathcal{N}[\![\alpha \cup \beta]\!]_{\mathsf{s}} = \mathcal{N}[\![\alpha]\!]_{\mathsf{s}} \cup \mathcal{N}[\![\beta]\!]_{\mathsf{s}}$$
$$\mathcal{N}[\![\alpha^{\mathrm{d}}]\!]_{\mathsf{s}} = \mathcal{N}[\![\alpha]\!]_{\mathsf{s}}{}^{\mathsf{D}} \qquad \mathcal{N}[\![\alpha;\beta]\!]_{\mathsf{s}} = \mathcal{N}[\![\alpha]\!]_{\mathsf{s}} \circ \mathcal{N}[\![\beta]\!]_{\mathsf{s}}$$
$$\mathcal{N}[\![\alpha^*]\!]_{\mathsf{s}}(A) = \mu B.(A \cup \mathcal{N}[\![\alpha]\!]_{\mathsf{s}}(B))$$

The semantics of ${\sim}a$ illustrates the role of the context. Playing the sabotage action ${\sim}a$ changes the context and assigns player $\diamond$ the game a to keep track of the Angelic sabotage.

The interpretation of negation and dualization is subtle, as these need to take into account the sabotage structure. For example in the game $\langle{\sim}a\rangle\neg\langle a\rangle\top$ the negation also *flips* the sabotage status. It can *not* be interpreted as saying Angel does not win $\langle a\rangle\top$ after Angel has sabotaged a by ${\sim}a$. Instead it means Angel does not win $\langle a\rangle\top$ if a was last sabotaged by *Demon*. The equivalent formula in normal form $\langle{\sim}a\rangle\langle a^{\mathrm{d}}\rangle\bot$ makes this clear. This subtlety can easily be avoided by working with games in *normal form* (Section 3.3.3).

3.3.2 *Semantics of Recursive Game Logic.*

Because recursive game logic contains games of the form $\mathsf{r}x.(c \cup a; x; b)$ unlike in game logic the semantics of such a game can no longer be defined as the fixpoint of a function between power sets. The plays of b that will take place after Angel chooses to play c must be taken into account.

The semantics of recursive game logic is defined with respect to both a monotone neighbourhood structure and a valuation. A *valuation* is a function $I : \mathbb{V} \to \mathcal{W}(|\mathcal{N}|)$ assigning an interpretation to every variable $x \in \mathbb{V}$. Given a valuation I, a variable $x \in \mathbb{V}$ and an effectivity function $w \in \mathcal{W}(|\mathcal{N}|)$ let $I[x \mapsto w]$ denote the valuation that agrees with I, except that $I[x \mapsto w](x) = w$.

Definition 3.2. For any monotone neighbourhood structure $\mathcal{N}$ and any valuation I define the *semantics* $\mathcal{N}[\![\varphi]\!]^I \in \mathcal{P}(|\mathcal{N}|)$ and $\mathcal{N}[\![\alpha]\!]^I \in \mathcal{W}(|\mathcal{N}|)$ by mutual induction for RGL formulas φ:

$$\mathcal{N}[\![P]\!]^I = \mathcal{N}(P) \qquad \mathcal{N}[\![\varphi \vee \psi]\!]^I = \mathcal{N}[\![\varphi]\!]^I \cup \mathcal{N}[\![\psi]\!]^I$$
$$\mathcal{N}[\![\neg\varphi]\!]^I = |\mathcal{N}| \setminus \mathcal{N}[\![\varphi]\!]^I \qquad \mathcal{N}[\![\langle\alpha\rangle\varphi]\!]^I = \mathcal{N}[\![\alpha]\!]^I(\mathcal{N}[\![\varphi]\!]^I)$$

and for RGL games α:

$$\mathcal{N}[\![a]\!]^I = \mathcal{N}(a) \qquad \mathcal{N}[\![?\varphi]\!]^I(A) = \mathcal{N}[\![\varphi]\!]^I \cap A$$
$$\mathcal{N}[\![x]\!]^I = I(x) \qquad \mathcal{N}[\![\alpha \cup \beta]\!]^I = \mathcal{N}[\![\alpha]\!]^I \cup \mathcal{N}[\![\beta]\!]^I$$
$$\mathcal{N}[\![\alpha^{\mathrm{d}}]\!]^I = (\mathcal{N}[\![\alpha]\!]^I)^{\mathrm{d}} \qquad \mathcal{N}[\![\alpha;\beta]\!]^I = \mathcal{N}[\![\alpha]\!]^I \circ \mathcal{N}[\![\beta]\!]^I$$
$$\mathcal{N}[\![\mathsf{r}x.\alpha]\!]^I = \mu u.\mathcal{N}[\![\alpha]\!]^{I[x\mapsto u]}$$

For closed formulas the superscript I is dropped. As usual the notation $\mathcal{N} \vDash \varphi$ means that $\mathcal{N}[\![\varphi]\!]^I = |\mathcal{N}|$ for all valuations I. Moreover write $\vDash \varphi$ if $\mathcal{N} \vDash \varphi$ for all *monotone neighbourhood structures* $\mathcal{N}$, and write $\vDash_K \varphi$ if $\mathcal{K} \vDash \varphi$ for all *Kripke structures* $\mathcal{K}$.

The semantics of recursive subgames is well-defined and the meaning of games $\mathsf{r}x.\alpha$ can be seen to be the least fixpoint by monotonicity of the function $u \mapsto \mathcal{N}[\![\alpha]\!]^{I[x\mapsto u]}$. The proof of this (Lemma 3.5) uses a normal form transformation for rlGL games.

3.3.3 *Normal Form.*

For some proofs it is important that negation is only applied to propositional atoms, and the duality operator is only applied to atomic games and free variables. Formulas and games that satisfy this condition are said to be in *normal form*.

Definition 3.3. By mutual recursion on RGL formulas and games define the *syntactic complement* $\overline{\varphi}$ of an RGL formula

$$\overline{P} = \neg P \qquad \overline{\neg\varphi} = \varphi \qquad \overline{\varphi \vee \psi} = \overline{\varphi} \wedge \overline{\psi} \qquad \overline{\langle\alpha\rangle\varphi} = \langle\alpha^{\mathrm{d}}\rangle\overline{\varphi}$$

and the *syntactic dual* α^{d} of a RGL game as follows

$$\begin{array}{lll} (a)^{\mathrm{d}} = a^{\mathrm{d}} & (x)^{\mathrm{d}} = x^{\mathrm{d}} & (\alpha^{\mathrm{d}})^{\mathrm{d}} = \alpha \\ (?\varphi)^{\mathrm{d}} = ¿\varphi & (\alpha \cup \beta)^{\mathrm{d}} = \alpha^{\mathrm{d}} \cap \beta^{\mathrm{d}} & (\alpha;\beta)^{\mathrm{d}} = \alpha^{\mathrm{d}};\beta^{\mathrm{d}} \\ (\mathrm{r}x.\alpha)^{\mathrm{d}} = \mathrm{ɹ}x.(\alpha^{\mathrm{d}}\frac{x^{\mathrm{d}}}{x}) & & \end{array}$$

By induction on the definition the syntactic complement and dual semantically correspond to set complements and dual functions:

Lemma 3.4. *For any* RGL *formula* φ *and for any* RGL *game* α*:*

$$\mathcal{N}[\![\overline{\varphi}]\!]^I = |\mathcal{N}| \setminus \mathcal{N}[\![\varphi]\!]^I \quad \textit{and} \quad \mathcal{N}[\![\alpha^{\mathrm{d}}]\!]^I = (\mathcal{N}[\![\alpha]\!]^I)^{\mathrm{d}}$$

A formula φ and a game α of RGL is said to be in *normal form* if negation is applied only to atomic propositions and the dual operator is applied only to atomic games and free variables. The following grammar describes the formulas and games of recursive game logic in normal form:

$$\varphi ::= P \mid \neg P \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \langle\alpha\rangle\varphi$$
$$\alpha ::= a \mid a^{\mathrm{d}} \mid x \mid x^{\mathrm{d}} \mid ?\varphi \mid ¿\varphi \mid \alpha \cup \beta \mid \alpha \cap \beta \mid \alpha;\beta \mid \mathrm{r}x.\alpha \mid \mathrm{ɹ}x.\alpha$$

with the usual assumptions that φ in $?\varphi$ or $¿\varphi$ is closed and x^{d} does not appear in the scope of a recursive game $\mathrm{r}x.\varphi$ or $\mathrm{ɹ}x.\varphi$. For every RGL formula φ the formula $\overline{\overline{\varphi}}$ is an equivalent formula in normal form by Lemma 3.4, called the *normal form of* φ. Similarly for every $\mathsf{GL_s}$ game α the game $\alpha^{\mathrm{d}^{\mathrm{d}}}$ is the equivalent *normal form of* α.

Lemma 3.5. *If* $\mathrm{r}x.\alpha$ *is an* RGL *game, then* $F : u \mapsto \mathcal{N}[\![\alpha]\!]^{I[x\mapsto u]}$ *is monotone.*

The notions of syntactic negation and syntactic dual can be extended to game logic with sabotage by defining

$$(\sim a)^{\mathrm{d}} = \sim a^{\mathrm{d}} \qquad\qquad (a^*)^{\mathrm{d}} = (a^{\mathrm{d}})^{\times}$$

Again the definition ensures that the syntactic negation and dual coincide with the semantic notions.

Lemma 3.6. *For any* $\mathsf{GL_s}$ *formula* φ *and any* $\mathsf{GL_s}$ *game* α*:*

$$\mathcal{N}[\![\overline{\varphi}]\!]_{\mathrm{s}} = \mathcal{N}[\![\varphi]\!]_{\mathrm{s}}{}^{\mathrm{C}} \qquad\qquad \mathcal{N}[\![\alpha^{\mathrm{d}}]\!]_{\mathrm{s}} = \mathcal{N}[\![\alpha]\!]_{\mathrm{s}}{}^{\mathrm{D}}$$

Analogously to game logic with sabotage a formula φ and a game α is said to be in *normal form* if negation is only applied to atomic propositions and the dual operator is only applied to atomic game and sabotage actions. The formulas and games of game logic with sabotage in normal form are given by the following grammar

$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \psi \mid \langle\alpha\rangle\varphi$$
$$\alpha ::= a \mid a^{\mathrm{d}} \mid \sim a \mid \sim a^{\mathrm{d}} \mid ?\varphi \mid ¿\varphi \mid \alpha \cup \beta \mid \alpha \cap \beta \mid \alpha;\beta \mid \alpha^* \mid \alpha^{\times}$$

As was the case for RGL, any $\mathsf{GL_s}$ formula φ has *its normal form* $\overline{\overline{\varphi}}$ and any game $\mathsf{GL_s}$ game α also has *its normal form* $\varphi^{\mathrm{d}^{\mathrm{d}}}$.

Corollary 3.7 (Normal Form). *Any formula and any game of* RGL *or* $\mathsf{GL_s}$ *is equivalent to its normal form.*

3.3.4 Semantic Compatibility. GL is a syntactic fragment of RGL (Section 3.2.3) and also the fragment of $\mathsf{GL_s}$ without sabotage actions. However the definitions of the semantics of both extensions are superficially different from the usual semantics of game logic. In the case of game logic with sabotage the semantics of the iteration games (Section 3.2.3) is in terms of a fixpoint of a monotone operator $\mathcal{P}(|\mathcal{N}|\times C) \to \mathcal{P}(|\mathcal{N}|\times C)$, whereas in Definition 3.2 they are in terms of a fixpoint of a transformation $\mathcal{W}(|\mathcal{N}|) \to \mathcal{W}(|\mathcal{N}|)$. The next two lemmas show that the two coincide with the definition in terms of operators $\mathcal{P}(|\mathcal{N}|) \to \mathcal{P}(|\mathcal{N}|)$, and thus that recursive game logic is indeed an extension of Parikh's game logic.

Lemma 3.8. *If* α *is a* GL *game then*

$$\mathcal{N}[\![\alpha^*]\!]^I(A) = \mu B.(A \cup \mathcal{N}[\![\alpha]\!]^I(B))$$

For a proof see the long version [2].

The $\mathsf{GL_s}$ semantics given to a formula, that is syntactically also a GL formula, coincide with the usual semantics:

Proposition 3.9. *If* φ *is a formula and* α *a game of* GL *then*

$$\mathcal{N}[\![\varphi]\!] = \mathcal{N}[\![\varphi]\!]_{\mathrm{s}}\restriction c_\emptyset \qquad \mathcal{N}[\![\alpha]\!](U\restriction c_\emptyset) = \mathcal{N}[\![\alpha]\!]_{\mathrm{s}}(U)\restriction c_\emptyset.$$

Game logic with sabotage is an extension of Parikh's game logic.

Proof. This is proved by a simple mutual induction on formulas and games. The case of repetition games uses Lemma 3.8. ◻

A $\mathsf{GL_s}$ formula φ *holds* in a structure $\mathcal{N}$ ($\mathcal{N} \vDash \varphi$) if $\mathcal{N}[\![\varphi]\!]_{\mathrm{s}} \supseteq |\mathcal{N}| \times \{c_\emptyset\}$. This captures the intended semantics of φ as being evaluated when no sabotage has taken place initially, by requiring the formula to hold in every state in the special context $c_\emptyset$ in which no atomic game has been sabotaged. A $\mathsf{GL_s}$ formula φ is *valid* ($\vDash \varphi$) if $\mathcal{N} \vDash \varphi$ for all *monotone neighbourhood structures* $\mathcal{N}$. Note that φ is valid iff $\mathcal{N}[\![\varphi]\!]_{\mathrm{s}} \supseteq |\mathcal{N}| \times C$ for all monotone neighbourhood structures $\mathcal{N}$. Write $\vDash_K \varphi$ if $\mathcal{K} \vDash \varphi$ for all *Kripke structures* $\mathcal{K}$. For formulas φ in the common syntactic fragment the overloading of notation for game logic formulas is justified by Proposition 3.9.

4 MODAL FIXPOINT LOGICS

The Modal μ*-Calculus.* This section recalls two modal fixpoint logics. Of particular interest is the *modal* μ*-calculus* (L_μ) [10], because of its desirable logical properties. It has decidable satisfiability and model checking problems, the finite model property and comes with a natural complete proof calculus. The syntax of L_μ is given by the following grammar:

$$\varphi ::= P \mid \neg P \mid x \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \langle a\rangle\varphi \mid [a]\varphi \mid \mu x.\varphi \mid \nu x.\varphi$$

for $P \in \mathbb{P}$, $a \in \mathbb{G}$ and $x \in \mathbb{V}$. The modal μ-calculus extends basic (multi)-modal logic with fixpoint operators $\mu x.\varphi$ and $\nu x.\varphi$. These denote the least and greatest fixpoints of φ in the sense that $\mu x.\varphi(x)$ is equivalent to $\varphi(\mu x.\varphi)$. The syntax enforces that fixpoint variables x can appear only positively in order to ensure that the semantics of fixpoint operators $\mu x.\varphi$ denote the desired extremal fixpoints.

Fixpoint Logic with Chop. An interesting extension of the modal μ-calculus is fixpoint logic with chop [32]. Although it lacks some of the nice properties of the modal μ-calculus, its high expressiveness is useful to establish a close correspondence with the game logics from the previous section via a natural translation. The following grammar defines the syntax of fixpoint logic with chop (FLC) [32]

$$\varphi ::= \mathrm{id} \mid P \mid \neg P \mid x \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \varphi \circ \psi \mid \mu x.\varphi \mid \nu x.\varphi$$

for $P \in \mathbb{P}$, $a \in \mathbb{G}$ and $x \in \mathbb{V}$. Fixpoint logic with chop is conceptually close to the modal μ-calculus. However fixpoint variables do not range over predicates (elements of $\mathcal{P}(|\mathcal{N}|)$), but over transformations (monotone functions in $\mathcal{W}(|\mathcal{N}|)$) instead. Consequently formulas denote predicate transformers which admit a natural notion of concatenation $\circ$ and identity transformation id. As in the modal μ-calculus the definition syntactically restricts to positive appearances of x, in order to ensure the well-definedness of the semantics of the fixpoint operator. The notation for syntactic substitution $\varphi\frac{\psi}{x}$ is the same as in recursive game logic.

Semantics of Fixpoint Logic with Chop. The *semantics* of fixpoint logic with chop is defined with respect to a monotone neighbourhood structure and a valuation $I : \mathbb{V} \to \mathcal{W}(|\mathcal{N}|)$. By structural induction on formulas φ define the set $\mathcal{N}[\![\varphi]\!]^I \in \mathcal{W}(|\mathcal{N}|)$

$$\begin{aligned}
\mathcal{N}[\![\mathrm{id}]\!]^I &= \mathrm{id} & \mathcal{N}[\![\varphi \vee \psi]\!]^I &= \mathcal{N}[\![\varphi]\!]^I \cup \mathcal{N}[\![\psi]\!]^I\\
\mathcal{N}[\![P]\!]^I &= \mathcal{N}(P) & \mathcal{N}[\![\varphi \wedge \psi]\!]^I &= \mathcal{N}[\![\varphi]\!]^I \cap \mathcal{N}[\![\psi]\!]^I\\
\mathcal{N}[\![\neg P]\!]^I &= |\mathcal{N}| \setminus \mathcal{N}(P) & \mathcal{N}[\![\langle a\rangle\varphi]\!]^I &= \mathcal{N}(a) \circ \mathcal{N}[\![\varphi]\!]^I\\
\mathcal{N}[\![x]\!]^I &= I(x) & \mathcal{N}[\![[a]\varphi]\!]^I &= \mathcal{N}(a)^{\mathrm{d}} \circ \mathcal{N}[\![\varphi]\!]^I\\
\mathcal{N}[\![\mu x.\varphi]\!]^I &= \mu q.\mathcal{N}[\![\varphi]\!]^{I[x\mapsto q]} & \mathcal{N}[\![\varphi \circ \psi]\!]^I &= \mathcal{N}[\![\varphi]\!]^I \circ \mathcal{N}[\![\psi]\!]^I\\
\mathcal{N}[\![\nu x.\varphi]\!]^I &= \nu q.\mathcal{N}[\![\varphi]\!]^{I[x\mapsto q]}
\end{aligned}$$

The semantics of μ and ν formulas denotes extremal fixpoints, since the semantics of FLC define a monotone function:

Lemma 4.1. *The function $F : q \mapsto \mathcal{N}[\![\varphi]\!]^{I[x\mapsto q]}$ is monotone.*

The semantics of a formula of fixpoint logic with chop is defined as a *monotone* function. To assign a truth value, the function can be evaluated at $\emptyset$ so that a formula φ holds in $\mathcal{N}$ ($\mathcal{N} \vDash \varphi$) if $\mathcal{N}[\![\varphi]\!]^I(\emptyset) = |\mathcal{N}|$ for all I. (The choice of $\emptyset$ is arbitrary but irrelevant and any FLC definable set can be used equivalently [32].) By monotonicity of the semantics this ensures that $\mathcal{N} \vDash \varphi$ iff $\mathcal{N}[\![\varphi]\!]^I(U) = |\mathcal{N}|$ for all I and all $U \subseteq |\mathcal{N}|$. Moreover write $\vDash \varphi$ if $\mathcal{N} \vDash \varphi$ for all *monotone neighbourhood structures* $\mathcal{N}$ and $\vDash_K \varphi$ if $\mathcal{K} \vDash \varphi$ for all *Kripke structures* $\mathcal{K}$.

The semantics of L_μ formulas with respect to the FLC semantics coincide with the usual semantics of the modal μ-calculus [32].

Negation in Fixpoint Logic with Chop. The negation of a formula of fixpoint logic with chop is defined syntactically as usual:

$$\begin{aligned}
&\overline{P} = \neg P & &\overline{\langle a\rangle\varphi} = [a]\overline{\varphi} & &\overline{\varphi \vee \psi} = \overline{\varphi} \wedge \overline{\psi} & &\overline{\mu x.\varphi} = \nu x.\overline{\varphi}\\
&\overline{\neg P} = P & &\overline{[a]\varphi} = \langle a\rangle\overline{\varphi} & &\overline{\varphi \wedge \psi} = \overline{\varphi} \vee \overline{\psi} & &\overline{\nu x.\varphi} = \mu x.\overline{\varphi}\\
&\overline{x} = x
\end{aligned}$$

The syntactic definition of negation corresponds semantically to complementation:

Lemma 4.2. *$\mathcal{N}[\![\overline{\varphi}]\!]^I(\emptyset) = |\mathcal{N}| \setminus \mathcal{N}[\![\varphi]\!]^{I^{\mathrm{c}}}(\emptyset)$ for all FLC formulas φ, where $I^{\mathrm{c}}(x) = (I(x))^{\mathrm{c}}$ is the pointwise complement of I.*

Proof. By a straightforward induction on FLC-formulas. □

$$\begin{tikzcd}
\mathsf{L}_* \arrow[dr, "\sharp"'] & \mathsf{L}_s \arrow[l, "\text{Lemma 5.7}"'] & \subset & \mathsf{L}_\mu \arrow[d, bend right, "\sharp"'] & \subset & \mathsf{FLC} \arrow[d, bend right, "\sharp"'] \\
 & \mathsf{GL} \arrow[u, "\reflectbox{$\flat$}"'] & \subset & \mathsf{rlGL} \arrow[u, bend right, "\reflectbox{$\flat$}"'] \arrow[d, bend right, "\natural"'] & \subset & \mathsf{RGL} \arrow[u, bend right, "\flat"'] \\
 & & & \mathsf{GL_s} \arrow[u, bend right, "c"'] & &
\end{tikzcd}$$

Figure 1: Translations between Fixpoint and Game Logics

With the syntactic negation, implication and equivalence can be defined in fixpoint logic with chop. The implication $\varphi \to \psi$ is viewed as an abbreviation for $\overline{\varphi} \vee \psi$ in FLC.

The Modal $*$-Calculus. Restricting the fixpoints in FLC to structured ones as they appear in game logic yields a logic we call the modal $*$-calculus, which is the exact modal fixpoint logic equivalent of game logic. The syntax of the *modal $*$-calculus* (L_*) is defined as

$$\varphi ::= \mathrm{id} \mid P \mid \neg\varphi \mid \varphi \vee \psi \mid \langle a\rangle\varphi \mid \varphi \circ \psi \mid \varphi^*$$

This can be viewed as a fragment of FLC by interpreting $\neg\varphi$ as $\overline{\varphi}$ and φ^* as an abbreviation for $\mu x.(\mathrm{id} \vee \varphi \circ x)$, where x is fresh.

5 EXPRESSIVENESS

The semantics of game logic and the modal μ-calculus are in many ways similar and game logic can express large parts of the modal μ-calculus. In particular it spans the *entire* fixpoint alternation hierarchy of the modal μ-calculus [8]. Nevertheless, game logic is less expressive than the modal μ-calculus [9]. This section introduces natural translations to show that, at the level of fixpoint logic with chop and recursive game logic, modal fixpoint logics and game logics can be identified *completely*. From this identification, the relationship of the expressiveness of game logic as a modal fixpoint logic and the expressiveness of the modal μ-calculus as a game logic can be determined *completely*.

Figure 1 gives a schematic overview of the translations between the logics. All inclusions in the illustration are strict. Game logic is strictly less expressive than the modal μ-calculus [9] and the modal μ-calculus is strictly less expressive than FLC [32]. The fragments rlGL and L_s are introduced in this section and the translations are presented and proved correct.

A formula φ of RGL is *well-named* if it does not bind the same variable twice and no variable appears both free and bound. Every formula is equivalent to a well-named formula by bound renaming.

5.1 Equiexpressiveness of FLC and RGL

5.1.1 Translation from fixpoint logic with chop to recursive game logic. Any formula φ of FLC can be expressed equivalently as a RGL game. The translated RGL game $\varphi^\sharp$ is defined by induction on the syntax of FLC formula φ as follows:

$$\begin{array}{lll}
(\mathrm{id})^\sharp = ?\top & (P)^\sharp = ?P; ¿\bot & (\varphi \lor \psi)^\sharp = \varphi^\sharp \cup \psi^\sharp \\
(x)^\sharp = x & (\neg P)^\sharp = ?\neg P; ¿\bot & (\varphi \land \psi)^\sharp = \varphi^\sharp \cap \psi^\sharp \\
(\langle a \rangle \varphi)^\sharp = a; \varphi^\sharp & (\mu x.\varphi)^\sharp = \mathrm{r}x.\varphi^\sharp & (\varphi \circ \psi)^\sharp = \varphi^\sharp; \psi^\sharp \\
([a]\varphi)^\sharp = a^{\mathrm{d}}; \varphi^\sharp & (\nu x.\varphi)^\sharp = \text{ɹ}x.\varphi^\sharp &
\end{array}$$

The translation $\varphi^\sharp$ of a FLC formula φ is always a RGL game in normal form. The RGL formula corresponding to φ is $\varphi^\sharp \equiv \langle \varphi^\sharp \rangle \bot$.

Proposition 5.1 ($\sharp$ Sound). *For any* FLC *formula φ the translation satisfies $\mathcal{N}[\![\varphi]\!]^I = \mathcal{N}[\![\varphi^\sharp]\!]^I$. Hence $\mathcal{N} \vDash \varphi$ iff $\mathcal{N} \vDash \varphi^\sharp$.*

Proof. By structural induction on φ. □

5.1.2 Translation from recursive game logic to fixpoint logic with chop. Conversely any formula of recursive game logic can be expressed equivalently in fixpoint logic with chop. To do this, fix two fresh variables u, v. Intuitively the purpose of these variables is to mark the end of the game, so that it can later be replaced by its game continuation. The difference between the two variables is that v marks games that end in fixpoint variables, while u marks the end of all other games. This distinction will only be important later when considering a particular subclass of formulas.

By Corollary 3.7 the translation can be defined by induction on the grammar of formulas and games in normal form. For any RGL formula φ and RGL game α define by induction an FLC formula $\varphi^\flat$

$$\begin{array}{lll}
(P)^\flat = P & (\varphi \lor \psi)^\flat = \varphi^\flat \lor \psi^\flat & (\langle \alpha \rangle \varphi)^\flat = \alpha^\flat \frac{\varphi^\flat}{\mathrm{u,v}} \\
(\neg P)^\flat = \neg P & (\varphi \land \psi)^\flat = \varphi^\flat \land \psi^\flat &
\end{array}$$

and the FLC formula $\alpha^\flat$

$$\begin{array}{lll}
(a)^\flat = \langle a \rangle \mathrm{u} & (\alpha \cup \beta)^\flat = \alpha^\flat \lor \beta^\flat & (?\psi)^\flat = \psi^\flat \land \mathrm{u} \\
(a^{\mathrm{d}})^\flat = [a]\mathrm{u} & (\alpha \cap \beta)^\flat = \alpha^\flat \land \beta^\flat & (¿\psi)^\flat = \neg(\psi)^\flat \lor \mathrm{u} \\
(x)^\flat = x \circ \mathrm{v} & (\mathrm{r}x.\alpha)^\flat = (\mu x.\alpha^\flat \frac{\mathrm{id}}{\mathrm{u,v}}) \circ \mathrm{u} & \\
(\alpha; \beta)^\flat = \alpha^\flat \frac{\beta^\flat}{\mathrm{u,v}} & (\text{ɹ}x.\alpha)^\flat = (\nu x.\alpha^\flat \frac{\mathrm{id}}{\mathrm{u,v}}) \circ \mathrm{u} &
\end{array}$$

Note that $\varphi \frac{\psi}{\mathrm{u,v}}$ denotes the formula obtained by simultaneously replacing all appearances of u and v in φ by ψ. This is different from successive substitution $\varphi \frac{\psi}{\mathrm{u}} \frac{\psi}{\mathrm{v}}$. The substitutions here are always admissible, that is no fixpoint construct captures a free variable. In fact none of the variables that are substituted (u, v) even appears in the context of a fixpoint in the translation. (The variables u, v are fresh and do not appear in the original formula or game.)

Proposition 5.2 ($\flat$ Sound). *For any well-named* RGL *formula φ and any well-named* RGL *game α in normal form*

(1) $\mathcal{N}[\![\varphi]\!]^I = \mathcal{N}[\![\varphi^\flat]\!]^I(A)$ *for any* $A \subseteq |\mathcal{N}|$
(2) $\mathcal{N}[\![\alpha]\!]^I \circ w = \mathcal{N}[\![\alpha^\flat]\!]^{I[\mathrm{u,v} \mapsto w]}$

Hence $\mathcal{N} \vDash \varphi$ iff $\mathcal{N} \vDash \varphi^\flat$.

Theorem 5.3 (Equiexpressiveness for FLC). *Recursive game logic (*RGL*) and fixpoint logic with chop (*FLC*) are equiexpressive.*

5.2 The Modal μ-Calculus as a Game Logic

In this section the precise extension of game logic that corresponds to the modal μ-calculus is identified. The lack of the fixpoint variables of the modal μ-calculus in game logic was remedied by introducing recursive subgames. This allows the modal μ-calculus to be understood as a game logic where games are played in a tail-recursive way, which captures the regularity of the modal μ-calculus in the context of recursive game logic.

A game α of recursive game logic is *right-linear* in x if it has no subgame $\beta; \gamma$ where x is free in β. A game α is *right-linear* if it contains a subgame $\mathrm{r}x.\beta$ only if β is right-linear in x. A formula φ of recursive game logic is *right-linear* if all its subgames are right-linear. The fragment of RGL consisting only of right-linear formulas and games is called *right-linear game logic* (rlGL).

The translation $\sharp$ transforms formulas of the modal μ-calculus to right-linear game logic, since the game α in all sequential games $\alpha; \beta$ introduced in the translation $\sharp$ is of the form a, a^{d}, $?P$ or $?\overline{P}$. For the converse, the translation $\flat$ can be modified to ensure that it only produces L_μ formulas. For any rlGL formula φ and any rlGL game α, the L_μ formulas $\varphi^{\reflectbox{$\flat$}}$ and $\alpha^{\reflectbox{$\flat$}}$ are defined by structural induction. The definition of $\alpha^{\reflectbox{$\flat$}}$ is identical to the definition of $\alpha^\flat$ in Section 5.1.2, except for the following cases

$$(x)^{\reflectbox{$\flat$}} = x \qquad (\alpha; \beta)^{\reflectbox{$\flat$}} = \alpha^{\reflectbox{$\flat$}} \frac{\beta^{\reflectbox{$\flat$}}}{\mathrm{u}} \qquad (\mathrm{r}x.\alpha)^{\reflectbox{$\flat$}} = \mu x.\alpha^{\reflectbox{$\flat$}}$$

Note that $\varphi^{\reflectbox{$\flat$}}$ is a modal μ-calculus formula, as it does not mention $\circ$. This is a generalization of a prior translation [22].

Proposition 5.4 ($\reflectbox{$\flat$}$ Sound). *The translation $\varphi^{\reflectbox{$\flat$}}$ of a well-named* rlGL *formula φ in normal form satisfies $\mathcal{N}[\![\varphi^{\reflectbox{$\flat$}}]\!]^I = \mathcal{N}[\![\varphi^\flat]\!]^I$.*

Theorem 5.5 (Equiexpressiveness for L_μ). *Right-linear game logic (*rlGL*) and the modal μ-calculus (L_μ) are equiexpressive.*

Proof. As noted $\varphi^\sharp$ is a formula of right-linear game logic provided φ is a formula in the modal μ-calculus. This shows that right-linear game logic is at least as expressive as the modal μ-calculus. The converse follows from Corollary 3.7 and Proposition 5.4. □

The next result is a consequence of Propositions 5.1, 5.2 and 5.4 and captures that the translations are inverse to each other.

Corollary 5.6 (Semantic Inverses). *The formulas $\vDash \varphi \leftrightarrow \varphi^{\sharp\reflectbox{$\flat$}}$ and $\vDash \psi \leftrightarrow \psi^{\reflectbox{$\flat$}\sharp}$ are valid for all well-named L_μ formulas φ and all well-named* rlGL *formulas ψ in normal form.*

5.3 Game Logic as a Fixpoint Logic

Recall from Section 4 that the modal $*$-calculus is the fragment of fixpoint logic with chop, which contains no fixpoints except in the form φ^*. Because the fixpoint structure in the modal $*$-calculus mirrors the structure in game logic, the translations between RGL and FLC also show the equiexpressiveness of the modal $*$-calculus and GL. This identifies the exact modal fixpoint logic corresponding to Parikh's original game logic.

The technical notion of formula separability will be used for the proof. A formula φ of the modal μ-calculus is *separable* if it contains fixpoints only in the forms $\mu x.(\psi \lor \rho)$ and $\nu x.(\psi \land \rho)$ where ρ does not mention x and ψ has no variable other than x free. Let L_s denote the set of separable formulas of the modal μ-calculus.

Lemma 5.7. (1) *If φ is a L_* formula, then $\varphi^\sharp$ is a* GL *formula.*
(2) *Any L_s formula is equivalent to an L_* formula.*
(3) *If φ is a well-named* GL *formula in normal form, then $\varphi^\natural$ is an L_s formula.*

Theorem 5.8 (Equiexpressiveness for GL). *Game logic (*GL*), the modal $*$-calculus (L_*), and the separable fragment of the modal μ-calculus (L_s) are equiexpressive.*

The equivalence between the separable fragment of the modal μ-calculus and game logic has been shown [14, Theorem 3.3.10]. Theorem 5.8 adds to this equivalence the modal $*$-calculus. It is still open whether game logic is equivalent to the two variable fragment of L_μ. By Theorem 5.8 this can be reduced to the question of whether every two-variable L_μ formula is expressible in L_*.

5.4 Game Logic with Sabotage as Right-linearity

Although sabotage actions are far from naturally expressible in rlGL, they do not add expressive power. This shows that game logic with sabotage is, like game logic, a fragment of the modal μ-calculus. The difficulty in embedding $\mathrm{GL_s}$ into rlGL is that the ownership information about previously committed acts of sabotage must be taken into account. This can be done by coding this information on the sabotaged atomic games into the nesting structure of the fixpoint variables. To simplify this coding, it uses simultaneous fixpoints, which do not add to the expressive power. This is captured by the following rendition adapting Beckić's Theorem to rlGL.

Theorem 5.9 (Beckić). *For variables $x_1, \ldots, x_n$ and* rlGL *games $\alpha_1, \ldots, \alpha_n$ there are* rlGL *games $\beta_1, \ldots, \beta_n$ such that*

$$\begin{pmatrix} \mathcal{N}[\![\beta_1]\!]^I \\ \mathcal{N}[\![\beta_2]\!]^I \\ \vdots \\ \mathcal{N}[\![\beta_m]\!]^I \end{pmatrix} = \mu \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} . \begin{pmatrix} \mathcal{N}[\![\alpha_1]\!]^{I[\vec{x} \mapsto \vec{w}]} \\ \mathcal{N}[\![\alpha_2]\!]^{I[\vec{x} \mapsto \vec{w}]} \\ \vdots \\ \mathcal{N}[\![\alpha_m]\!]^{I[\vec{x} \mapsto \vec{w}]} \end{pmatrix}$$

Let $\mathsf{r}_i(x_1, \ldots, x_n).(\alpha_1, \ldots, \alpha_n)$ denote the rlGL *game β_i.*

Proof. An adaptation of Beckić's Theorem [5, Lemma 1.4.2]. □

Fix for every possible context $c \in C$ a fresh variable y_c. For any formula φ and any game α of game logic with sabotage a translation α^c depending on the context c is defined. The context allows the translation to depend on the state of sabotage of atomic games. Moreover the translation of games will contain free variables y_c. Those mark the end of the game and keep track of the context in which this end has been reached. This allows a compositional definition of the translation. For a context c, a $\mathrm{GL_s}$ formula φ and a $\mathrm{GL_s}$ game α in normal form, the rlGL games φ^c and α^c are defined by mutual induction on the $\mathrm{GL_s}$ formulas φ and games α:

$$(P)^c = ?P; ¿\bot \qquad (\varphi \vee \psi)^c = \varphi^c \cup \psi^c$$
$$(\neg P)^c = ?\neg P; ¿\bot \qquad (\varphi \wedge \psi)^c = \varphi^c \cap \psi^c$$

$$(a)^c = \begin{cases} a; y_c & \text{if } c(a) = \emptyset \\ y_c & \text{if } c(a) = \diamond \\ ?\bot & \text{if } c(a) = \square \end{cases} \qquad (a^{\mathrm{d}})^c = \begin{cases} a^{\mathrm{d}}; y_c & \text{if } c(a) = \emptyset \\ ¿\bot & \text{if } c(a) = \diamond \\ y_c & \text{if } c(a) = \square \end{cases}$$

$$(\sim a)^c = y_{c\frac{\diamond}{a}} \qquad (?\varphi)^c = \varphi^c \cap y_c \qquad (\alpha \cup \beta)^c = \alpha^c \cup \beta^c$$
$$(\sim a^{\mathrm{d}})^c = y_{c\frac{\square}{a}} \qquad (¿\varphi)^c = \overline{\varphi}^c \cup y_c \qquad (\alpha \cap \beta)^c = \alpha^c \cap \beta^c$$

The translations of atomic games and sabotage games illustrates the importance of translating relative to a context. The translation of formulas $\langle\alpha\rangle\varphi$ and games $\alpha;\beta$ and $\alpha^*, \alpha^\times$ is slightly more involved. For the first two define

$$(\langle\alpha\rangle\varphi)^c = \alpha^c \frac{\varphi^\bullet;?\bot}{y_\bullet} \qquad\qquad (\alpha;\beta)^c = \alpha^c \frac{\beta^\bullet}{y_\bullet}$$

where the notation $\alpha^c \frac{\beta^\bullet}{y_\bullet}$ means that any instance of a variable y_e is replaced by β^e, the translation of β with respect to context e. This shows the role of the variables y_c as placeholders for the continuation of the game. In the translation $(\psi)^c \cap (\varphi)^c; ?\bot$ of formula $(\langle ?\psi\rangle\varphi)^c$, variable y_c is a placeholder for the formula $(\varphi)^c$.

For the translation of an iteration game, all possible ways of playing this game, depending on what has been sabotaged and how, are considered *simultaneously*. To this end, fix for a context c and fixpoint games α^* and $\alpha^\times$ a list of all contexts $c_1, \ldots, c_m$ that satisfy $c_i(a) = \emptyset$ if a and a^{d} do not appear in α. The translation of the repetition games is defined simultaneously for all c_i

$$(\alpha^*)^{c_i} = \mathsf{r}_i(z_{c_1}, \ldots, z_{c_n}).(y_{c_1} \cup \alpha^{c_1} \frac{z_{c_\bullet}}{c_\bullet}, \ldots, y_{c_n} \cup \alpha^{c_n} \frac{z_{c_\bullet}}{y_{c_\bullet}})$$
$$(\alpha^\times)^{c_i} = \text{ɹ}_i(z_{c_1}, \ldots, z_{c_n}).(y_{c_1} \cap \alpha^{c_n} \frac{z_{c_\bullet}}{y_{c_\bullet}}, \ldots, y_{c_n} \cap \alpha^{c_n} \frac{z_{c_\bullet}}{y_{c_\bullet}})$$

where the $z_{c_\bullet}$ are fresh variables. Observe that the translation of any $\mathrm{GL_s}$ game is a right-linear game logic game and the translation of any $\mathrm{GL_s}$ formula is a *closed* right-linear game logic game.

The next proposition shows that the translation is correct.

Proposition 5.10 ($\cdot^c$*Sound*). *Let $I(y_e) = U{\restriction}e$. For any $\mathrm{GL_s}$ formula φ and any $\mathrm{GL_s}$ game α in normal form*

$$\mathcal{N}[\![\varphi]\!]_{\mathrm{s}}{\restriction}c = \mathcal{N}[\![\varphi^c]\!](\emptyset) \qquad \mathcal{N}[\![\alpha]\!]_{\mathrm{s}}(U){\restriction}c = \mathcal{N}[\![\alpha^c]\!]^I(\emptyset)$$

The translation of a game logic with sabotage formula into a formula of right-linear game logic potentially grows very quickly. The upper bound on the length of the translation of a fixpoint game obtained from the above proof is[2]

$$|\alpha^c| \leq (K \cdot |\alpha|)^{(3^\ell)\uparrow\uparrow d},$$

where K is a constant, d is the fixpoint nesting depth of α and ℓ is the number of atomic games for which there are sabotage actions in α. This comes from the fact that the translation of any game α considers all of the up to 3^ℓ-many relevant contexts. The only known transformation from vectorial fixpoints to non-vectorial nested fixpoints as in Theorem 5.9 grows exponentially in the formula size. Consequently every fixpoint leads to a doubly exponential blow-up in length. In [13] it is shown that reducing vectorial fixpoints to non-vectorial fixpoints is at least as hard as solving parity games, for which the existence of a polynomial time algorithm is a longstanding open question. It has been conjectured [11] that a vectorial fixpoint formula can be exponentially smaller than the shortest equivalent non-vectorial formula.

While it is unclear to what extent this upper bound is optimal, it suggests that complex formulas of the modal μ-calculus may be expressed much more succinctly in game logic with sabotage.

[2]The up-arrow notation $n \uparrow\uparrow m$ denotes m-fold iterated exponentiation, i.e. $n \uparrow\uparrow 0 = 1$ and $n \uparrow\uparrow (m+1) = n^{n\uparrow\uparrow m}$.

5.5 Sabotage Memory

This section shows how a $\mathsf{GL_s}$ game can use sabotage to model memory. This expressive power will be used to translate from rlGL into $\mathsf{GL_s}$. It is straightforward to store Boolean information by sabotaging a fresh atomic game. However as it will be necessary for both players to retrieve the information stored in the game, two atomic games are used to store a bit. This makes it possible to define a game $?\mathbf{a}$ that Angel can skip if the value associated to a is true. If the value is false, Angel loses the game prematurely.

To encode this formally in $\mathsf{GL_s}$, consider a list $\mathbf{a} = a_1, \dots, a_n$ of atomic games and define a composite game $\mathbf{a}{:=}i$ by

$$\sim a_1^{\mathrm{d}}; \dots; \sim a_n^{\mathrm{d}}; \sim a_i$$

for all $1 \leq i \leq n$. In this context, $?\mathbf{a}{=}i$ is synonymous notation for a_i. As long as sabotage actions for any atomic game a_i only appear in the context of $\mathbf{a}{:=}\cdot$ then, after a game of the form $\mathbf{a}{:=}i$ has been played once, Angel can win the game $?\mathbf{a}{=}j$ exactly if the last time a subgame of the form $\mathbf{a}{:=}j$ has been played was with $i = j$.

In case $n = 2$ the list $\mathbf{a} = (a_1, a_2)$ is used to memorize a binary value. Writing $\mathbf{a}!$ for $\mathbf{a}{:=}1$ is viewed as setting the value of $\mathbf{a}$ to true. Similarly the game $\neg\mathbf{a}!$ representing $\mathbf{a}{:=}2$ is understood to set the value of $\mathbf{a}$ to false. Then the game $?\mathbf{a}$ can be defined as $?\mathbf{a}{=}1$ and has the desired property described above. Dually, define $?\overset{\mathrm{d}}{\mathbf{a}}$ to stand for a_2^{d}. This game is skipped if the value for $\mathbf{a}$ is true and Demon loses otherwise. Similarly Demon tests if the value for $\mathbf{a}$ is false by playing $?\overset{\mathrm{d}}{\neg}\mathbf{a}$, defined as a_1^{d}. Note that $(\neg\mathbf{a}!)^{\mathrm{d}}$ is equivalent to $\mathbf{a}!$, $(?\neg\mathbf{a})^{\mathrm{d}}$ is equivalent to $?\overset{\mathrm{d}}{\mathbf{a}}$ and $(?\mathbf{a})^{\mathrm{d}}$ is equivalent to $?\overset{\mathrm{d}}{\neg}\mathbf{a}$.

5.6 Right-linear Game Logic as Sabotage

With the help of sabotage memory, it is possible to express every right-linear game logic formula in game logic with sabotage and consequently also every modal μ-calculus formula. This shows that game logic with sabotage is an *expressive completion* of game logic as a fragment of the modal μ-calculus.

The challenge of the converse translation from rlGL to $\mathsf{GL_s}$ is that the arbitrarily nested recursive games of rlGL need to be turned into structured repetition games of game logic with sabotage. Using sabotage, players can force the behaviour of nested recursive games onto structured repetition games. To facilitate this, fix fresh atomic games x_1 and x_1 for every variable x and let $\mathsf{x} = (\mathsf{x}_1, \mathsf{x}_2)$ as sabotage memory. By mutual induction define the translation of a rlGL formula in normal form into a $\mathsf{GL_s}$ formula $\varphi^{\natural}$

$$(P)^{\natural} = P \qquad (\varphi \vee \psi)^{\natural} = \varphi^{\natural} \vee \psi^{\natural} \qquad (\langle\alpha\rangle\varphi)^{\natural} = \langle\alpha^{\natural}\rangle\varphi^{\natural}$$
$$(\neg P)^{\natural} = \neg P \qquad (\varphi \wedge \psi)^{\natural} = \varphi^{\natural} \wedge \psi^{\natural}$$

and the translation of a normal form rlGL game into a $\mathsf{GL_s}$ game $\alpha^{\natural}$

$$(a)^{\natural} = a \qquad (?\varphi)^{\natural} = ?\varphi^{\natural} \qquad (\alpha \cup \beta)^{\natural} = \alpha^{\natural} \cup \beta^{\natural}$$
$$(a^{\mathrm{d}})^{\natural} = a^{\mathrm{d}} \qquad (¿\varphi)^{\natural} = ¿\varphi^{\natural} \qquad (\alpha \cap \beta)^{\natural} = \alpha^{\natural} \cap \beta^{\natural}$$
$$(\alpha; \beta)^{\natural} = \alpha^{\natural}; \beta^{\natural} \qquad (\mathrm{r}x.\alpha)^{\natural} = \neg\mathsf{x}!; (?\neg\mathsf{x}; \mathsf{x}!; \alpha^{\natural}\tfrac{\neg\mathsf{x}!}{x})^{*}; ?\mathsf{x}$$
$$(x)^{\natural} = x \qquad (\text{ɹ}x.\alpha)^{\natural} = \mathsf{x}!; (?\overset{\mathrm{d}}{\mathsf{x}}; \neg\mathsf{x}!; \alpha^{\natural}\tfrac{\mathsf{x}!}{x})^{\times}; ?\overset{\mathrm{d}}{\neg}\mathsf{x}$$

The translation is into a $\mathsf{GL_s}$ game with variables, where in $\mathsf{GL_s}$ the variables are viewed as atomic games. This is necessary for a compositional definition. For a rlGL game α the GL formula $\alpha^{\natural}$ is defined to be $\langle\alpha^{\natural}\rangle\bot$.

Intuitively the translation of the fixpoints uses rule changes to remove the choices from the repetition game. In a normal α^* game it is Angel's choice whether to continue playing the game α or not. However in general $\mathrm{r}x.\alpha$ games, this choice is made differently. If the variable x is reached the game *must* be repeated. If the game ends without reaching this variable it *must not* be repeated. The translation enforces this deterministic behaviour of the repetition game in a * iteration game via tests. Although the * iteration game theoretically allows Angel to stop prematurely, Angel is constrained by the complementary tests $?\mathsf{x}$ and $?\neg\mathsf{x}$ at the beginning of the loop body and after the loop. Throughout the play of α the players use x to *memorize* whether a variable x has been reached in which case $?\mathsf{x}$ stops Angel from ending the game prematurely. Otherwise Angel cannot safely repeat the game.

Proposition 5.11 ($\natural$ Sound). *For any well-named formula φ of rlGL in normal form, the translation $\varphi^{\natural}$ is a $\mathsf{GL_s}$ formula with*

$$\mathcal{N}[\![\varphi^{\natural}]\!]_{\mathrm{s}}\restriction c_{\emptyset} = \mathcal{N}[\![\varphi]\!].$$

Theorem 5.12 (Equiexpressiveness). *Game logic with sabotage, right-linear game logic and the modal μ-calculus are equiexpressive.*

Proof. By Propositions 5.10 and 5.11 and Theorem 5.5. □

This completes the picture of the relative expressiveness of the game logics and fixpoint logics in Figure 1. The equiexpressiveness of $\mathsf{GL_s}$ and the L_μ means that game logic with sabotage inherits many of the nice properties of the modal μ-calculus for free.

Theorem 5.13 (Meta properties). *Game logic with sabotage has the small model property and its satisfiability problem is decidable.*

Proof. The modal μ-calculus has these properties [40, 43], and they transfer to game logic with sabotage by Theorem 5.12. □

According to the definition of RGL and $\mathsf{GL_s}$, games may contain tests of arbitrary formulas. For example the formula $\langle ?(\langle a\rangle P)\rangle P$ is a well-formed game logic formula. This *rich-test* version is in contrast to the *poor-test* version in which only tests of Boolean combinations of literals (i.e. formulas P and $\neg P$) are allowed. As a corollary of the equiexpressiveness results it follows that rich-tests (and even anything beyond literals) do not add expressive power.

Corollary 5.14 (Tests). *The poor-test versions of game logic, right-linear game logic, recursive game logic and game logic with sabotage are equiexpressive with their respective rich-test versions.*

Proof. For game logic and right-linear game logic this can be seen by translating into the corresponding fragment of fixpoint logic with chop via $\flat$ or ɗ, since the backward translation via $\sharp$ yields an equivalent (Corollary 5.6) poor-test formula, since the translation $\sharp$ only introduces tests of literals. If φ is a $\mathsf{GL_s}$ formula then $\varphi^{c_\emptyset}$ is an equivalent rlGL formula. By the above there is an equivalent poor-test rlGL formula ψ and hence $\psi^{\natural}$ is a poor-test $\mathsf{GL_s}$ formula equivalent to φ. In fact, these merely test literals. □

6 AXIOMATIZATION

This section introduces natural proof calculi for rlGL and $\mathrm{GL_s}$. Kozen's original calculus for L_μ and its monotone variant are recalled, since completeness for these game logics is obtained from completeness for the modal μ-calculus.

6.1 Proof Calculi for the Modal μ-Calculus

Because this paper is also concerned with the modal μ-calculus interpreted over neighbourhood structures, the *monotone modal μ-calculus* mL_μ [23], the restriction of Kozen's calculus for the modal μ-calculus for neighbourhood structures, is recalled here. The monotone modal μ-calculus consists of all propositional tautologies together with all instances of the following axioms:

$$(\mu)\ \varphi\frac{\mu x.\varphi}{x} \rightarrow \mu x.\varphi$$

$$(\alpha)\ \sigma x.\varphi \leftrightarrow \sigma y.\varphi\frac{y}{x} \quad (y \text{ fresh}, \sigma \in \{\mu, \nu\})$$

The rules of the proof calculus are:

$$(\mathrm{MP})\ \frac{\varphi \quad \varphi \rightarrow \psi}{\psi} \qquad (\mathrm{M}_a)\ \frac{\varphi \rightarrow \psi}{\langle a\rangle\varphi \rightarrow \langle a\rangle\psi} \qquad (\mathrm{FP}_\mu)\ \frac{\varphi\frac{\psi}{x} \rightarrow \psi}{\mu x.\varphi \rightarrow \psi}$$

Write $\mathsf{mL}_\mu \vdash \varphi$ if there is a Hilbert style proof of φ in the monotone modal μ-calculus. Note that this is a subset of Kozen's proof calculus for the modal μ-calculus [30]. Adding the following two axioms yields the full Kozen calculus.

$$([]\top)\ [a]\top \qquad ([]\wedge)\ [a]\varphi \wedge [a]\psi \rightarrow [a](\varphi \wedge \psi)$$

Write $\mathsf{L}_\mu \vdash \varphi$ if there is a Hilbert-style proof in this calculus of the formula φ. The reverse implication of $[]\wedge$ is derivable by M_a. Kripke's distribution axiom (K) is also derivable in this calculus. (See the long version [2].) The two proof calculi for L_μ are complete:

Proposition 6.1 (Walukiewicz [45]). *Kozen's calculus is sound and complete with respect to Kripke structures. That is* $\vDash_K \varphi$ *iff* $\mathsf{L}_\mu \vdash \varphi$ *for* L_μ *formulas* φ.

Proposition 6.2 (Enqvist, Seifan, Venema [23]). *The monotone modal μ-calculus is sound and complete with respect to monotone neighbourhood structures. That is* $\vDash \varphi$ *iff* $\mathsf{mL}_\mu \vdash \varphi$ *for* L_μ *formulas* φ

6.2 Proof Calculi for Game Logics

6.2.1 Parikh's Proof Calculus for Game Logic. Parikh proposed a similar Hilbert-style proof calculus for game logic [35]. It consists of all propositional tautologies as axioms together with the axioms

$$(\cup)\ \langle\alpha \cup \beta\rangle\varphi \leftrightarrow \langle\alpha\rangle\varphi \vee \langle\beta\rangle\psi \qquad ({}^d)\ \langle\alpha^{\mathrm{d}}\rangle\varphi \leftrightarrow \neg\langle\alpha\rangle\neg\varphi$$

$$(;)\ \langle\alpha;\beta\rangle\varphi \leftrightarrow \langle\alpha\rangle\langle\beta\rangle\varphi \qquad (?)\ \langle ?\varphi\rangle\psi \leftrightarrow \varphi \wedge \psi$$

$$(*)\ \varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle\varphi \rightarrow \langle\alpha^*\rangle\varphi$$

and the following rules

$$(\mathrm{MP}_G)\ \frac{\varphi \quad \varphi \rightarrow \psi}{\psi} \qquad (\mathrm{M}_G)\ \frac{\varphi \rightarrow \psi}{\langle\alpha\rangle\varphi \rightarrow \langle\alpha\rangle\psi} \qquad (\mathrm{FP}_*)\ \frac{\rho \vee \langle\alpha\rangle\psi \rightarrow \psi}{\langle\alpha^*\rangle\rho \rightarrow \psi}$$

If a GL formula is provable in this calculus, write $\mathrm{GL} \vdash \varphi$. If φ is provable in the same calculus with the additional two axioms $[]\top$ and $[]\wedge$, write $\mathrm{GL} + G \vdash \varphi$.

6.2.2 A Proof Calculus for Right-linear Game Logic. The proof calculus for game logic can be extended to a proof calculus for right-linear game logic by adding the following axioms and rule:

$$(\mathrm{BR})\ \langle\sigma x.\alpha\rangle\varphi \leftrightarrow \langle\sigma y.\alpha\frac{y}{x}\rangle\varphi \quad (y \text{ fresh}, \sigma \in \{\mathsf{r}, \text{ɹ}\})$$

$$(\mathsf{r})\ \langle\alpha\frac{\mathsf{r}x.\alpha}{x}\rangle\varphi \rightarrow \langle\mathsf{r}x.\alpha\rangle\varphi$$

$$(\mathrm{FP}_\mathsf{r})\ \frac{\langle\alpha\frac{\beta;?\psi;¿\bot}{x}\rangle\varphi \rightarrow \langle\beta\rangle\psi}{\langle\mathsf{r}x.\alpha\rangle\varphi \rightarrow \langle\beta\rangle\psi} \quad (\alpha \text{ right-linear in } x)$$

Note that the axiom $*$ and the rule FP_* do not need to be added to the calculus explicitly as these are derivable from r and FP_r respectively. The rule FP_r is a version of the least fixpoint rule for right-linear games. As before for a rlGL formula φ write $\mathrm{rlGL} \vdash \varphi$ if φ is provable in this calculus and $\mathrm{rlGL} + G \vdash \varphi$ if φ is provable with the two additional axioms $[]\top$ and $[]\wedge$.

A more general proof calculus for full recursive game logic is of interest as well. Because there cannot be a recursive and complete such calculus, only the calculus for right-linear game logic is considered here.

Remark 6.3. In the three calculi restricting rule M_G to range only over atomic games a, dual atomic games a^{d} and variables x does not weaken the proof calculi, since the more general rule is derivable. With the more general M_G it is clear that if $\mathrm{GL} \vdash \varphi$, then $\mathrm{GL} \vdash \varphi\frac{\alpha}{x}$ by substituting free occurrences of x across the entire proof.

Theorem 6.4 (rlGL Soundness). *For any* rlGL *formula* φ

(1) $\vDash \varphi$ *if* $\mathrm{rlGL} \vdash \varphi$
(2) $\vDash_K \varphi$ *if* $\mathrm{rlGL} + G \vdash \varphi$

Proof. The proof is a straightforward extension of the soundness proof for Parikh's game logic calculus. Soundness of the rule FP_r is shown in the long version [2]. □

Proposition 6.5. *The following are derivable in the* rlGL *calculus:*

$$(¿)\ \langle ¿\varphi\rangle\psi \leftrightarrow (\varphi \rightarrow \psi)$$

$$(\cap)\ \langle\alpha \cap \beta\rangle\varphi \leftrightarrow \langle\alpha\rangle\varphi \wedge \langle\beta\rangle\psi$$

$$(\overleftarrow{\mathsf{r}})\ \langle\mathsf{r}x.\alpha\rangle\varphi \rightarrow \langle\alpha\frac{\mathsf{r}x.\alpha}{x}\rangle\varphi$$

$$(\mathrm{FP}_\text{ɹ})\ \frac{\langle\beta\rangle\psi \rightarrow \langle\alpha\frac{\beta;?\psi;¿\bot}{x}\rangle\rho}{\langle\beta\rangle\psi \rightarrow \langle \text{ɹ}x.\alpha\rangle\rho} \quad (\alpha \text{ right-linear in } x)$$

$$(\mathrm{RL})\ \langle\alpha\rangle\varphi \leftrightarrow \langle\alpha\frac{x;?\varphi;¿\bot}{x}\rangle\varphi \quad (\alpha \text{ right-linear in } x)$$

Axioms $¿$, $\cap$ and rule $\mathrm{FP}_\text{ɹ}$ are the dual versions to the Angelic axioms $?$, $\cup$ and the Angelic rule FP_r. Axiom $\overleftarrow{\mathsf{r}}$ is the reverse version of r and axiom RL captures the right-linearity of α in x syntactically.

6.3 Completeness for Right-linear Game Logic

The translations between right-linear game logic and the modal μ-calculus show not only equiexpressiveness, but also that the proof calculi are equivalent. This enables the transfer of completeness from the modal μ-calculus to right-linear game logic.

The translations between right-linear game logic and the modal μ-calculus have been proved sound semantically. In order to use these to relate the proof calculi, the soundness of the translation needs to be proved in the calculus itself. Since each calculus can only talk about formulas in its respective language the relevant soundness here is that of Corollary 5.6. This is proved by induction on a well-order on all formulas defined in the long version [2].

Lemma 6.6 (Provable Inverses). (1) $\mathsf{rlGL} \vdash \varphi \leftrightarrow \varphi^{\natural\sharp}$ *for any well-named* rlGL *formula* φ *in normal form*
(2) $\mathsf{mL}_\mu \vdash \varphi \leftrightarrow \varphi^{\sharp\natural}$ *for any well-named* L_μ *formula* φ

The key is that proofs in the modal μ-calculus can be turned into right-linear game logic proofs. Since the modal μ-calculus is complete and has the same expressive power as right-linear game logic it follows that any formula φ is provable up to translation.

Theorem 6.7 (Equipotency). *Right-linear game logic and the modal μ-calculus prove the same formulas (modulo translation).*
(1) $\mathsf{mL}_\mu \vdash \varphi$ *iff* $\mathsf{rlGL} \vdash \varphi^\sharp$ *for closed well-named* L_μ *formulas* φ
(2) $\mathsf{rlGL} \vdash \varphi$ *iff* $\mathsf{mL}_\mu \vdash \varphi^\natural$ *for closed well-named* rlGL *formulas* φ *in normal form*

As the translations are semantically correct and preserve provability, completeness of rlGL follows with Proposition 6.2.

Theorem 6.8 (rlGL Completeness). *For any* rlGL *formula* φ
(1) $\vDash \varphi$ *iff* $\mathsf{rlGL} \vdash \varphi$
(2) $\vDash_K \varphi$ *iff* $\mathsf{rlGL} + G \vdash \varphi$

Proof. (1) The $\Leftarrow$ implication is by Theorem 6.4. For the $\Rightarrow$ implication, consider first the case that φ is closed. By BR assume without loss of generality that φ is well-named. By Corollary 3.7 and Lemma 6.10 assume that φ is in normal form. The following chain of equivalences proves the first claim of the theorem

$$\begin{array}{lll} & \vDash \varphi & \\ \text{iff} & \vDash \varphi^\natural & \text{(Propositions 5.2 and 5.4)} \\ \text{iff} & \mathsf{mL}_\mu \vdash \varphi^\natural & \text{(Proposition 6.2)} \\ \text{iff} & \mathsf{rlGL} \vdash \varphi & \text{(Theorem 6.7)} \end{array}$$

For non-closed φ, for each free variable x in φ fix fresh atomic games b_x and let $\tilde{\varphi}$ be the closed formula obtained from φ by replacing all x by b_x. Then $\vDash \varphi$ iff $\vDash \tilde{\varphi}$ and $\mathsf{rlGL} \vdash \varphi$ iff $\mathsf{rlGL} \vdash \tilde{\varphi}$, so the equivalence holds, since in the proof calculus free variables and atomic games are interchangeable.

(2) Analogous to (1). See the long version [2] for details. □

6.4 Completeness of Game Logic with Sabotage

The complete proof calculus for GL can be extended to a complete proof calculus for $\mathsf{GL_s}$ by adding axioms for the sabotage actions. To simplify the formulation of these axioms some notation is introduced. The axioms will affect subformulas that occur potentially deep inside a formula. A *formula-context* $C(\bullet_1, \dots, \bullet_n)$ is a $\mathsf{GL_s}$ formula φ with distinguished atomic games $\bullet_i$. The $\mathsf{GL_s}$ formula $C(\alpha_\bullet)$ is obtained from φ by replacing every $\bullet_i$ by α_i, and the subscript is dropped if there is only one $\bullet_i$. A formula-context C is said to be *a-free* if it does not mention a, a^{d}, $\sim a$ or $\sim a^{\mathrm{d}}$.

The sabotage axioms for $\mathsf{GL_s}$ are summarized in Figure 2. Axioms $\sim$, $\wr$, $\approx$ and $\wr\wr$ syntactically capture the immediate effect of a

$$\begin{array}{llll} (\sim) & \langle \sim a; a\rangle\varphi \leftrightarrow \langle\sim a\rangle\varphi & (\approx) & \langle\sim a; \sim a\rangle\varphi \leftrightarrow \langle\sim a\rangle\varphi \\ (\wr) & \neg\langle\sim a^{\mathrm{d}}; a\rangle\varphi & (\wr\wr) & \langle\sim a; \sim a^{\mathrm{d}}\rangle\varphi \leftrightarrow \langle\sim a^{\mathrm{d}}\rangle\varphi \end{array}$$

$$\begin{array}{lll} (\oslash) & \langle\sim a\rangle C(\alpha_\bullet; ¿\bot) \leftrightarrow C(\sim a; \alpha_\bullet; ¿\bot) & (\text{if } C \text{ is } a\text{-free}) \\ (\simeq) & C(\sim a) \leftrightarrow C(?\top) & (\text{if } a, a^{\mathrm{d}} \notin C) \\ (\cong) & \langle\sim a\rangle(C(\sim a^{\mathrm{d}}) \leftrightarrow C(\sim a^{\mathrm{d}}; \sim b^{\pm\mathrm{d}})) & (\text{if } \sim a \text{ guards } b \text{ in } C\ ) \\ (\|) & \langle\sim a\rangle(C(a) \leftrightarrow C(a; \sim b^{\pm\mathrm{d}})) & (\text{if } a \text{ remembers } \sim b^{\pm\mathrm{d}} \text{ in } C) \\ (\Upsilon) & \langle \mathrm{a}{:=}i\rangle(C(\beta) \leftrightarrow C(\bigcup_{1\le j\le n} ?\mathrm{a}{=}j; \mathrm{a}{:=}j; \beta)) & (\sim a_i^{\pm\mathrm{d}} \text{ only in } \mathrm{a}) \end{array}$$

Figure 2: The Axioms for Game Logic with Sabotage

sabotage action and axiom $\oslash$ allows reasoning about effects deep within a formula. Axioms $\simeq$ and $\cong$ allow the uniform removal of sabotage actions which do not have any effect. In $\simeq$ the atomic games a, a^{d} are never played, so sabotaging a does not change anything. In $\cong$ the sabotage action $\sim a$ is ineffective because b is guarded. An atomic game b is said to be *guarded by $\sim a$ in C* if b, b^{d} and $\sim a$ appear only in the form $a; b$, $a; b^{\mathrm{d}}$, and $a; \sim a$ respectively. Guardedness ensures that b can never be played after $\sim a^{\mathrm{d}}$ has been played. The $\sim a^{\pm\mathrm{d}}$ in the axiom stands for either $\sim a$ or $\sim a^{\mathrm{d}}$ everywhere. A close syntactic relationship between the value of a and a sabotage action persists through a play and this is captured by axiom $\|$, where *a remembers $\sim b^{\pm\mathrm{d}}$ in C* if $\sim a$ appears in C only as $\sim a; \sim b^{\pm\mathrm{d}}$. This condition ensures that whenever Angel can play a, this is because she has previously sabotaged a and at the same time b was sabotaged. This sabotage remains in effect, so that the additional sabotage $\sim b^{\pm\mathrm{d}}$ has no effect on the play. In axiom Υ it is assumed that the games $\sim a_i$, $\sim a_i^{\mathrm{d}}$ appear only in the memory games $\mathrm{a}{:=}j$ in C and β. Axiom Υ is sound, since at any stage during the play of C there will be a value in the range $1, \dots, n$ associated to the sabotage memory $\mathrm{a} = a_1, \dots, a_n$. The value can be determined by branching over all possible values and re-assigning the determined value afterwards is sound as it has no effect, but is useful in inductive proofs.

The proof calculus for game logic with sabotage is Parikh's original proof calculus for game logic (Section 6.2.1) together with the additional axioms from Figure 2. If there is a Hilbert-style proof of φ in this calculus consisting only of $\mathsf{GL_s}$ formulas, write $\mathsf{GL_s} \vdash \varphi$. The proof calculus for $\mathsf{GL_s}$ can be modified to Kripke structures, by adding the two axioms $[]\top$ and $[]\wedge$. Write $\mathsf{GL_s} + G \vdash \varphi$ if there is a proof of φ in this extension.

Theorem 6.9 ($\mathsf{GL_s}$ Soundness). *For any* $\mathsf{GL_s}$ *formula* φ
(1) $\vDash_K \varphi$ *if* $\mathsf{GL_s} + G \vdash \varphi$
(2) $\vDash \varphi$ *if* $\mathsf{GL_s} \vdash \varphi$

The equivalence of a formula with its normal form (Corollary 3.7) can be proved syntactically.

Lemma 6.10 (Provable Normal Form). *Any formula and any game of* RGL *or* $\mathsf{GL_s}$ *is* provably *equivalent to its normal form.*

6.5 Proof Transformations

This section shows that the translation respects the proof calculus. Combined with the semantic correctness of the translation this enables the transfer of completeness from rlGL to $\mathsf{GL_s}$. The key fact needed about the translation is that the sabotage paraphrasing of recursive games *provably* behaves the same as the extremal fixpoint it denotes. As a consequence, rlGL proofs can be translated to $\mathsf{GL_s}$.

Proposition 6.11 ($\natural$ Transformation). *For a well-named right-linear game logic formula φ in normal form if $\mathsf{rlGL} \vdash \varphi$ then $\mathsf{GL_s} \vdash \varphi^\natural$.*

Lemma 6.12 (Provable Inverse). *Suppose φ is a formula of game logic with sabotage in normal form then $\mathsf{GL_s} \vdash (\varphi^{c_\emptyset})^\natural \to \varphi$.*

Theorem 6.13 (Game Logic with Sabotage Completeness). *Game logic with sabotage is sound and complete. That is for all $\mathsf{GL_s}$ formulas φ:*

$$\vDash \varphi \qquad \textit{iff} \qquad \mathsf{GL_s} \vdash \varphi$$

Proof. The $\Leftarrow$ implication holds by Theorem 6.9. For $\Rightarrow$ by Corollary 3.7 and Lemma 6.10 assume that φ is in normal form. If φ is a valid formula of $\mathsf{GL_s}$, the translation $\langle \varphi^{c_\emptyset} \rangle \bot$ is a valid right-linear game logic formula by Proposition 5.10, closed and in normal form. Hence $\mathsf{rlGL} \vdash \varphi$ by Theorem 6.8. and by Proposition 6.11 also $\mathsf{GL_s} \vdash (\varphi^{c_\emptyset})^\natural$. Finally by Lemma 6.12 and MP_G, $\mathsf{GL_s} \vdash \varphi$. □

6.6 Completion of Parikh's Calculus for GL

Game logic with sabotage is the *expressive completion* of game logic as a fragment of the modal μ-calculus (Section 5). Next a *completion* of Parikh's proof calculus for game logic (GL) is obtained from the complete $\mathsf{GL_s}$ proof calculus from Section 6.4.

The axiomatization of game logic with sabotage is an extension of game logic with the set of sabotage axioms from Figure 2. No additional rules are added. By Theorem 6.13 every valid GL formula is provable in the $\mathsf{GL_s}$ calculus. Such a proof is almost a GL proof, except that MP_G may introduce $\mathsf{GL_s}$ formulas that are not expressible in GL. In this case the sabotage actions in the introduced formula can be viewed as atomic games from a distinguished set of atomic games. For a set of of atomic games $\Gamma \subset \mathbb{G}$ let S_Γ be the set of axioms from Figure 2 which do *not* mention atomic games from Γ. Let $\mathcal{S}_\Gamma$ be the set of GL formulas obtained from S_Γ by replacing all sabotage actions $\sim a$ by some fresh atomic game $\tilde{a}$. Taken as axioms these GL formulas (!) suffice to complete Parikh's proof calculus for game logic. Write $\mathsf{GL} + \mathcal{S}_\Gamma \vdash \varphi$ if there is a proof of φ in Parikh's calculus from these axioms and $\mathsf{GL} + G + \mathcal{S}_\Gamma \vdash \varphi$ if there is a proof with the additional axioms $[]\top$ and $[]\wedge$.

Theorem 6.14 (GL Completeness). *For any GL formula φ let Γ be the set of atomic games in φ. Then:*

(1) $\vDash \varphi$ *iff* $\mathsf{GL} + \mathcal{S}_\Gamma \vdash \varphi$
(2) $\vDash_K \varphi$ *iff* $\mathsf{GL} + G + \mathcal{S}_\Gamma \vdash \varphi$

Proof. For $\Leftarrow$ (Soundness), suppose $\mathsf{GL} + \mathcal{S}_\Gamma \vdash \varphi$ and consider a monotone neighbourhood structure $\mathcal{N}$. A proof for $\mathsf{GL_s} \vdash \varphi$ can be obtained by uniformly replacing every one of the fresh atomic games $\tilde{a}$ (from $\mathcal{S}_\Gamma$) by $\sim a$ in the proof $\mathsf{GL} + \mathcal{S}_\Gamma \vdash \varphi$. Every instance of an axiom from $\mathcal{S}_\Gamma$ is an instance of the original $\mathsf{GL_s}$ version. By soundness (Theorem 6.9) and Proposition 3.9 conclude that $\vDash \varphi$.

For $\Rightarrow$ (Completeness), suppose φ is a valid GL formula. As in the proof of Theorem 6.13 obtain a proof of $\mathsf{GL_s} \vdash \varphi$. By the construction of this proof (Proposition 6.11) the $\mathsf{GL_s}$ proof of φ contains games of the form $\sim a$ only for atomic games that do not appear in φ. The $\mathsf{GL_s}$ proof of φ can be transformed into a GL proof by uniformly replacing every game of the form $\sim a$ by the fresh atomic game $\tilde{a}$. Instances of the axioms from Figure 2 in the original proof become instances of axioms in $\mathcal{S}_\Gamma$. Hence the modified version of the proof is a GL proof of φ from the axioms in $\mathcal{S}_\Gamma$.

The case for Kripke structures is analogous using $\mathsf{GL} + G$. □

The results in this section pave the way for using the proof calculus for $\mathsf{GL_s}$ to resolve the question of completeness of Parikh's axiomatization for GL through a proof transformation by eliminating instances of axioms from $\mathcal{S}_\Gamma$.

7 CONCLUSION

This paper studies how logic, games, and fixpoints meet by introducing two different extensions of game logic. The first, game logic with sabotage, allows players to sabotage their opponent, while the second, recursive game logic, adds recursive games.

Not only is game logic with sabotage ($\mathsf{GL_s}$) well-suited for describing and investigating games with rule changes by logical means but, surprisingly, game logic with sabotage has a number of additional advantages over game logic. Unlike game logic (GL), the extension $\mathsf{GL_s}$ allows *exactly* the right amount of state to increase its expressive power to match the modal μ-calculus, without sacrificing the desirable logical properties of game logic.

The advantage of recursive game logic (RGL) is that it allows the description of games featuring arbitrarily nested (co)recursive games. Unlike ordinary game logic, the extended version is significantly more expressive than the modal μ-calculus, although it remains syntactically close to GL. This paper identified the fragment of RGL that corresponds exactly to the modal μ-calculus in expressiveness and transferred completeness of the modal μ-calculus to obtain a complete and natural proof calculus for this fragment.

It was shown that game logic with sabotage and the modal μ-calculus are equivalent in expressiveness via a *syntactically provable translation* going through the right-linear fragment of recursive game logic. Completeness of the natural Hilbert style proof calculus for game logic with sabotage $\mathsf{GL_s}$ was obtained as a consequence. This is in contrast to game logic GL for which completeness of the natural proof calculus is not known [29]. Completeness of game logic with sabotage was used to obtain the completeness of a modest extension of Parikh's proof calculus for game logic GL.

Future Research. The completeness of game logic with sabotage suggests an interesting approach to studying proof calculi for game logic. It reduces the problem of the completeness of Parikh's axiomatization of game logic to eliminating instances of the new axioms in a proof. Equiexpressiveness with L_μ indicates that atomic games for sabotage are worth studying further.

The translation from $\mathsf{GL_s}$ into L_μ leads to a non-elementary blow-up in formula length. This raises the question whether this increase is necessary and if better algorithms exist that directly target the model checking and satisfiability problems of $\mathsf{GL_s}$.

ACKNOWLEDGMENTS

An Alexander von Humboldt Professorship supported this research.

REFERENCES

[1] Noah Abou El Wafa and André Platzer. 2022. First-Order Game Logic and Modal Mu-Calculus. arXiv:2201.10012

[2] Noah Abou El Wafa and André Platzer. 2024. Complete Game Logic with Sabotage. arXiv:2404.09873

[3] S. Abramsky and P.-A. Mellies. 1999. Concurrent games and full completeness. In *Proceedings. 14th Symposium on Logic in Computer Science*. IEEE, Trento, Italy, 431–442. https://doi.org/10.1109/LICS.1999.782638

[4] Bahareh Afshari and Graham E. Leigh. 2017. Cut-free completeness for modal mu-calculus. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. IEEE Computer Society, Reykjavik, 1–12. https://doi.org/10.1109/LICS.2017.8005088

[5] André Arnold and Damian Niwinski. 2001. *Rudiments of mu-calculus*. Studies in Logic and the Foundations of Mathematics, Vol. 146. North Holland Publishing Co., Amsterdam.

[6] Guillaume Aucher, Johan van Benthem, and Davide Grossi. 2015. Sabotage Modal Logic: Some Model and Proof Theoretic Aspects. In *Logic, Rationality, and Interaction - 5th International Workshop, LORI 2015 Taipei, Taiwan, October 28-31, 2015, Proceedings (LNCS, Vol. 9394)*, Wiebe van der Hoek, Wesley H. Holliday, and Wen-Fang Wang (Eds.). Springer, Taipei, 1–13. https://doi.org/10.1007/978-3-662-48561-3_1

[7] Guillaume Aucher, Johan van Benthem, and Davide Grossi. 2018. Modal logics of sabotage revisited. *J. Log. Comput.* 28, 2 (2018), 269–303. https://doi.org/10.1093/LOGCOM/EXX034

[8] Dietmar Berwanger. 2003. Game Logic is Strong Enough for Parity Games. *Studia Logica* 75, 2 (2003), 205–219. https://doi.org/10.1023/A:1027358927272

[9] Dietmar Berwanger and Giacomo Lenzi. 2005. The Variable Hierarchy of the μ-Calculus Is Strict. In *STACS 2005*, Volker Diekert and Bruno Durand (Eds.). Springer, Berlin, Heidelberg, 97–109. https://doi.org/10.1007/978-3-540-31856-9_8

[10] Julian Bradfield and Colin Stirling. 2007. Modal Mu-Calculi. In *Handbook of Modal Logic*, Patrick Blackburn, Johan Van Benthem, and Frank Wolter (Eds.). Studies in Logic and Practical Reasoning, Vol. 3. Elsevier, Amsterdam, 721–756. https://doi.org/10.1016/S1570-2464(07)80015-2

[11] Julian Bradfield and Igor Walukiewicz. 2018. The mu-calculus and Model Checking. In *Handbook of Model Checking*, Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). Springer International Publishing, Cham, 871–919. https://doi.org/10.1007/978-3-319-10575-8_26

[12] Julian C. Bradfield. 1996. The Modal mu-calculus Alternation Hierarchy is Strict. In *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings (LNCS, Vol. 1119)*, Ugo Montanari and Vladimiro Sassone (Eds.). Springer, Pisa, 233–246. https://doi.org/10.1007/3-540-61604-7_58

[13] Florian Bruse, Oliver Friedmann, and Martin Lange. 2015. On guarded transformation in the modal μ-calculus. *Log. J. IGPL* 23, 2 (2015), 194–216. https://doi.org/10.1093/JIGPAL/JZU030

[14] Facundo Carreiro. 2015. *Fragments of fixpoint logics*. Ph. D. Dissertation. University of Amsterdam.

[15] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. 2007. Strategy Logic. In *CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings (LNCS, Vol. 4703)*, Luís Caires and Vasco Thudichum Vasconcelos (Eds.). Springer, Lisbon, 59–73. https://doi.org/10.1007/978-3-540-74407-8_5

[16] Corina Cîrstea, Clemens Kupke, and Dirk Pattinson. 2009. EXPTIME Tableaux for the Coalgebraic μ-Calculus. In *Computer Science Logic*, Erich Grädel and Reinhard Kahle (Eds.). Springer, Berlin, Heidelberg, 179–193. https://doi.org/10.1007/978-3-642-04027-6_15

[17] Pierre Clairambault. 2009. Least and Greatest Fixpoints in Game Semantics. In *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings (LNCS, Vol. 5504)*, Luca de Alfaro (Ed.). Springer, York, 16–31. https://doi.org/10.1007/978-3-642-00596-1_3

[18] Andrzej Ehrenfeucht. 1961. An application of games to the completeness problem for formalized theories. *Fundamenta Mathematicae* 49 (1961), 129–141. https://api.semanticscholar.org/CorpusID:118038695

[19] E. Allen Emerson and Charanjit S. Jutla. 1991. Tree Automata, Mu-Calculus and Determinacy. In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*. IEEE Computer Society, San Juan, 368–377. https://doi.org/10.1109/SFCS.1991.185392

[20] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. 2001. On model checking for the μ-calculus and its fragments. *Theor. Comput. Sci.* 258, 1-2 (2001), 491–522. https://doi.org/10.1016/S0304-3975(00)00034-7

[21] Sebastian Enqvist, Helle Hvid Hansen, Clemens Kupke, Johannes Marti, and Yde Venema. 2019. Completeness for Game Logic. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*. IEEE, Vancouver, 1–13. https://doi.org/10.1109/LICS.2019.8785676

[22] Sebastian Enqvist, Fatemeh Seifan, and Yde Venema. 2018. Completeness for the modal μ-calculus: Separating the combinatorics from the dynamics. *Theor. Comput. Sci.* 727 (2018), 37–100. https://doi.org/10.1016/j.tcs.2018.03.001

[23] Sebastian Enqvist, Fatemeh Seifan, and Yde Venema. 2019. Completeness for μ-calculi: A coalgebraic approach. *Ann. Pure Appl. Log.* 170, 5 (2019), 578–641. https://doi.org/10.1016/J.APAL.2018.12.004

[24] Alessandro Facchini, Yde Venema, and Fabio Zanasi. 2013. A Characterization Theorem for the Alternation-Free Fragment of the Modal µ-Calculus. In *Proceedings of the Twenty-Eighth Annual IEEE Symposium on Logic in Computer Science (LICS 2013)*. IEEE Computer Society Press, New Orleans, 478–487. https://doi.org/10.1109/LICS.2013.54

[25] Nina Gierasimczuk, Lena Kurzen, and Fernando R. Velázquez-Quesada. 2009. Learning and Teaching as a Game: A Sabotage Approach. In *Logic, Rationality, and Interaction*, Xiangdong He, John Horty, and Eric Pacuit (Eds.). Springer, Berlin, Heidelberg, 119–132.

[26] Yuri Gurevich and Leo Harrington. 1982. Trees, Automata, and Games. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber (Eds.). ACM, San Francisco, 60–65. https://doi.org/10.1145/800070.802177

[27] Jaakko Hintikka. 1982. Game-theoretical semantics: insights and prospects. *Notre Dame Journal of Formal Logic* 23, 2 (1982), 219–241.

[28] David Janin and Igor Walukiewicz. 1996. On the Expressive Completeness of the Propositional mu-Calculus with Respect to Monadic Second Order Logic. In *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings (LNCS, Vol. 1119)*, Ugo Montanari and Vladimiro Sassone (Eds.). Springer, Pisa, 263–277. https://doi.org/10.1007/3-540-61604-7_60

[29] Johannes Kloibhofer. 2023. A note on the incompleteness of Afshari & Leigh's system Clo. https://doi.org/10.48550/arXiv.2307.06846 arXiv:2307.06846 [math.LO]

[30] Dexter Kozen. 1983. Results on the Propositional μ-Calculus. *Theor. Comput. Sci.* 27, 3 (1983), 333–354. https://doi.org/10.1016/0304-3975(82)90125-6

[31] Christof Löding and Philipp Rohde. 2003. Model Checking and Satisfiability for Sabotage Modal Logic. In *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings (LNCS, Vol. 2914)*, Paritosh K. Pandya and Jaikumar Radhakrishnan (Eds.). Springer, Mumbai, 302–313. https://doi.org/10.1007/978-3-540-24597-1_26

[32] Markus Müller-Olm. 1999. A Modal Fixpoint Logic with Chop. In *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings (LNCS, Vol. 1563)*, Christoph Meinel and Sophie Tison (Eds.). Springer, Trier, 510–520. https://doi.org/10.1007/3-540-49116-3_48

[33] Sara Negri. 2017. Proof theory for non-normal modal logics: The neighbourhood formalism and basic results. *IfCoLog Journal of Logics and their Applications* 4, 4 (2017), 1241–1286. http://www.collegepublications.co.uk/downloads/ifcolog00013.pdf

[34] Damian Niwinski and Igor Walukiewicz. 1996. Games for the mu-Calculus. *Theor. Comput. Sci.* 163, 1&2 (1996), 99–116. https://doi.org/10.1016/0304-3975(95)00136-0

[35] Rohit Parikh. 1983. Propositional game logic. In *24th Annual Symposium on Foundations of Computer Science*. IEEE, Tucson, 195–200. https://doi.org/10.1109/SFCS.1983.47

[36] Marc Pauly and Rohit Parikh. 2003. Game Logic - An Overview. *Stud Logica* 75, 2 (2003), 165–182. https://doi.org/10.1023/A:1027354826364

[37] André Platzer. 2015. Differential Game Logic. *ACM Trans. Comput. Log.* 17, 1 (2015), 1. https://doi.org/10.1145/2817824

[38] André Platzer. 2017. Differential Hybrid Games. *ACM Trans. Comput. Log.* 18, 3 (2017), 19:1–19:44. https://doi.org/10.1145/3091123

[39] André Platzer. 2018. *Logical Foundations of Cyber-Physical Systems*. Springer, Cham. https://doi.org/10.1007/978-3-319-63588-0

[40] Vaughan R. Pratt. 1981. A Decidable mu-Calculus: Preliminary Report. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*. IEEE Computer Society, Nashville, 421–427. https://doi.org/10.1109/SFCS.1981.4

[41] Philipp Rohde. 2006. On the μ-Calculus Augmented with Sabotage. In *Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-31, 2006, Proceedings (LNCS, Vol. 3921)*, Luca Aceto and Anna Ingólfsdóttir (Eds.). Springer, Vienna, 142–156. https://doi.org/10.1007/11690634_10

[42] Colin Stirling. 1996. Games and Modal Mu-Calculus. In *Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop, TACAS '96, Passau, Germany, March 27-29, 1996, Proceedings (LNCS, Vol. 1055)*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer, Passau, 298–312. https://doi.org/10.1007/3-540-61042-1_51

[43] Robert S. Streett and E. Allen Emerson. 1989. An Automata Theoretic Decision Procedure for the Propositional Mu-Calculus. *Inf. Comput.* 81, 3 (1989), 249–264. https://doi.org/10.1016/0890-5401(89)90031-X

[44] Johan van Benthem. 2005. An Essay on Sabotage and Obstruction. In *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, Dieter Hutter and Werner Stephan (Eds.). Springer, Berlin,

Heidelberg, 268–276. https://doi.org/10.1007/978-3-540-32254-2_16

[45] Igor Walukiewicz. 1995. Completeness of Kozen's Axiomatisation of the Propositional mu-Calculus. In *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, June 26-29, 1995*. IEEE Computer Society, San Diego, 14–24. https://doi.org/10.1109/LICS.1995.523240