

Game Quantification Patterns^{*}

Dietmar Berwanger¹ and Sophie Pinchinat²

RWTH Aachen, Germany
IRISA Rennes, France

Abstract. We analyse two basic approaches of extending classical logics with quantifiers interpreted via games: Propositional Game Logic of Parikh and Alternating-Time Temporal Logic of Alur, Henzinger, and Kupferman. Although the two approaches are historically remote and they incorporate operationally orthogonal paradigms, we trace the formalisms back to common foundations and argue that they share remarkable similarities in terms of expressive power.

1 Introduction

The metaphor of games is at the basis of a rich and intuitive language for reasoning about interaction. Over the past three decades, substantial efforts have been made to integrate the elements of this language into logical formalisms (see [20] for a comprehensive survey).

We discuss two basic approaches towards formal reasoning about games: the Propositional Logic of Games introduced by Parikh [16] in 1983, which is the first formalism to incorporate games into a logic of computation, and the framework of Alternating-Time Temporal Logics of Alur, Henzinger, and Kupferman [2] introduced 15 years later, which is arguably the most influential game-based formalism in Computer-Science applications by today.

Both formalisms emerged from well-established logics for reasoning about the dynamics of computation. Parikh's Game Logic GL extends the Program Dynamic Logic (PDL) of Fischer and Ladner [9] by adding a dualisation operation that turns the description of a program into one of an interactive protocol.¹ The main representative of Alternating-Time Logics, ATL^{*}, generalises the Computation-Tree Logic CTL^{*} of Emerson and Halpern [8] to speak about the course of events in a multi-agent system. The two formalisms at the outset represent different specification paradigms: PDL captures an internal view on the execution of a program whereas CTL^{*} reflects an external view on the dynamics of a computation. Accordingly, PDL quantifies over relations between program states, whereas CTL^{*} quantifies over computation traces. Nevertheless, when viewed as extensions of the basic monomodal logic K with recursion mechanisms [19], the two formalisms turn out to have similar expressive power: they

^{*} This research was supported by the Deutsche Forschungsgemeinschaft (DFG), the European COMBEST project, and the Indo-French research network TIMED-DISCOVERI.

¹ The term Game Logic with the abbreviation GL has also been used in [2] to denote a formalism that is unrelated to Parikh's Game Logic.

can both be embedded into the second alternation level and into the two-variable fragment of the μ -calculus; moreover, PDL with an additional loop-operator subsumes CTL*.

The casting of PDL and CTL* into logics of interaction occurs at two levels. At a local level, the basic modal quantifier which ranges over possible outcomes of a computation step is replaced in GL and ATL* with atomic-game operators associated to the outcome of an interactive event. However, atomic games a priori do not feature utilities; these arise only at a global level as winning conditions over plays, i.e., sequences of interactive events. To build rules for forming plays and winning conditions, GL and ATL* use logical constructs which largely preserve their meaning from the underlying logics of computation: Boolean and linear-time connectives, choice and iteration operators, and higher-order quantifiers over sequences of events.

On a first view, GL and ATL* may be seen as formalisms for reasoning about complex games composed from atomic ones. However, the analysis of rational behaviour in a game embedded within another game is notoriously difficult, if not hopeless. (Most of the questions raised 1971 in the seminal work of Howard on metagames [13] have remained unsolved so far.) In fact, the logics we consider do not pursue this aim; as the atomic games to which their semantics refer lack utilities, they are not games in the strict sense, but rather *game forms*, that is, descriptions of outcome functions. Essentially, both GL and ATL* lead a cut between two basic elements of game-oriented reasoning: the local *outcome* of an interactive event which is represented in the model, and the global *utility* drawn from a sequence of events which is determined by the formula. This separation between interdependent action and interactive decision-making reflects in the fundamental semantic constructs of the two formalisms. In Game Logic, the dualisation operation corresponds to a swap of capabilities, rather than utilities, between the players. In Alternating-Time Logics, atomic-game events are performed by (coalitions of) agents that are not equipped with subjective utility functions. The formal interpretation of these constructs sometimes contradicts the game-theoretic intuition delivered by the natural-language description of the logics. For a critical discussion on such aspects and recent approaches towards defining more natural semantics for ATL*, see [1] and [5].

Nevertheless, there is a sense in which GL and ATL* recover the proposition of compositional game-based reasoning: the *semantic games* of these logics do arise as compositions of atomic game forms via logical formulae. Semantic games, also called *model-checking* games, are zero-sum games associated to the question of whether a formula holds in a model or not [12]. Typically, there are two players, a Verifier who performs existential choices (e.g., decomposition of disjunctions, assignment of existentially quantified variables) and a Falsifier who performs universal choices (e.g., decomposition of conjunction, assignment of universally quantified variables); the Verifier can ensure to win if, and only if, the formula holds in the model. The correspondence between logics of computation and their semantic games is usually mediated via a specific automata model. For a general background on model-checking via games and automata see [11].

In this paper, we discuss terminological and technical challenges arising from the combination of interactive and compositional reasoning. We put forward the thesis that the formalisms of Game Logic and Alternating-Time Logics are effectively confined to the scope of determined two-player games with perfect information composed from atomic game forms. Game-theoretic concepts beyond this scope, e.g., those inherent to non-zero-sum games, imperfect information, or to games with more than two players essentially cannot be captured.² To substantiate this claim, we argue that the model-checking games for GL and ATL* —which characterise their semantics— are determined two-player games with perfect information.

The first part of the paper, Section 2, details the concept of an atomic game which is at the basis of the semantics of the two logics. We introduce distinct terms for notions that tend to be confounded in the literature. We maintain that a partial description of a game which lacks a utility function shall be called a game form. Likewise, an actor who can choose an action but who is not equipped with a utility function shall rather be called agent than player. We introduce the notion of an untyped game to denote abstract descriptions of interactive situation where actions are not yet assigned to the players. This representation subsumes the notions of effectivity function and that of concurrent game which underlie the semantics of Game Logic and Alternating-Time Logics, respectively.

The definition of GL and ATL* is deliberately postponed until the terminological issues are settled. Originally, the semantics of the two logics is defined on different kinds of models: neighbourhood structures, or Montague-Scott models, and concurrent game structures. We introduce a common interpretation domain of *extensive game structures* based on untyped game forms, which generalise both neighbourhood and concurrent game structures. In Section 3, we present the semantics of GL and ATL*. To relate their expressive power, we show that ATL* is invariant under replacing atomic game forms while preserving the effectivity. As a consequence, it follows that the meaning of a ATL*-formula is determined by its meaning over neighbourhood models, that is, over the interpretation domain of GL-formulae. This invariance result relies on the notion of *sequentialisation* of an untyped game form which represents the only scenarios where choices are made and communicated to the other agents in a certain order.

Finally in Section 4, we introduce an automata-theoretic formalism that subsumes both GL and ATL* to describe how the recursion mechanisms of GL and

² This statement may seem to contradict the purpose of Alternating-Time Logics which is motivated as a formalism for speaking about (concurrent) games with several players. The contradiction can be traced back to a common terminological inaccuracy. A player incorporates different functions in a game: he is an *agent* with the capacity to perform actions, and, at the same time, he is a rational *decision maker* able to choose which action to perform. When we have game models of computational systems in mind, there are good reasons to distinguish between the two functions. The range of actions available to a player is typically determined by the design of the system whereas rational decisions on which actions to choose depend on the system specification. In light of this, the players invoked in the original definition of Alternating-Time Logics should be understood as non-deliberate agents [7].

ATL* are reflected through structural properties of automata, thus explaining limitations of their expressive power. The translation of GL and ATL* into automata, implicitly defines a notion of model-checking games for the two logics.

2 Atomic games

As outlined in the introduction, at the core of our analysis are games that involve two players; we will call them *Ego* and *Alter*. The basic model is that of zero-sum games with two possible utility values -1 and 1, representing a win or a loss, respectively. Such a game is represented in *normal form* by a tuple (S^E, S^A, Z, π, u^E) , where S^E and S^A are the sets of *strategies* or *actions* available to Ego and Alter, respectively, Z is the set of possible *outcomes* determined by the *play* function $\pi : S^A \times S^E \rightarrow Z$, and $u^E : Z \rightarrow \{-1, 1\}$ is a *utility* function associating to every outcome a winning or losing value for Ego. We sometimes write $s \hat{=} t$ to denote $\pi(s, t)$.

We investigate different ways in which (descriptions of) games are composed out of (descriptions of) their parts. This section fixes our terminology for speaking about parts that are atomic in the sense that they involve only one round of interaction. The central notion is that of a game form, – a partial representation of a game which omits utilities. With effectivity functions and agent forms, we introduce two particular representations of game forms that will be used to define Game Logic and Alternating-Time Logics.

Game forms, types. At the most abstract level, an *untyped game form* is a tuple $\Gamma = (S, Z, \pi)$ specifying a set of strategies that is not associated to any particular player, a set of possible outcomes, and a (partial) play function $\pi : S \times S \rightarrow Z$. A game *type* α identifies a subset $S_\alpha \subseteq S$ of strategies in an untyped game form. The purpose of a game type, or simply type, is to designate the strategies available to the players for playing their part in a game. For each concrete modelling domain, a collection ACT of types is fixed beforehand. Types come in pairs: for every type $\alpha \in \text{ACT}$ there is a *dual type* $-\alpha \in \text{ACT}$, the dual of which is again α . The instantiation of an untyped game form Γ with a type α yields the (typed) game form $\Gamma_\alpha := (S_\alpha^E, S_{-\alpha}^A, Z, \pi)$. While the play function π might be only partially defined in Γ , we require it to be complete in every game form Γ_α with $\alpha \in \text{ACT}$.

For instance, a matrix $p : [m] \times [n] \rightarrow Z$ can be viewed as an untyped game form $\Gamma = (S, Z, p)$ where the set of strategies consists of all row and column indices, $S = [\max\{m, n\}]$. (We denote by $[n]$ the set $[1, \dots, n]$.) There are two natural types $\text{ACT} = \{\text{row}, \text{col}\}$ with $\text{col} = -\text{row}$, which associate the sets of rows and columns to the two players. The game form $\Gamma_{\text{row}} = ([m]^E, [n]^A, Z, p)$ represents the scenario in which Ego chooses a row and Alter, simultaneously, chooses a column whereas the dual $\Gamma_{\text{col}} = ([n]^E, [m]^A, Z, p)$ represents the scenario where Ego chooses a column and Alter chooses a row. By associating a utility $u : Z \rightarrow \{-1, 1\}$ for Ego to matrix entries, we obtain the games (Γ_{row}, u) and (Γ_{col}, u) . Notice that these two games are in general different.

Our notion of game type is not standard in Game Theory; it is related to the established notion of a player type in games with incomplete information only in the loose sense that it makes an abstract description of a game more concrete. Intuitively, untyped game forms allow us to specify actions by abstracting from concrete players or agents who may perform them. Our example illustrates two uses of types, the motivations for which are both rather particular to Computer-Science applications. On the one hand, different types can be applied to an untyped game form to define several games on the basis of a single description. On the other hand, types provide us with a way to separate the concept of a player from that of an agent: we may, e.g., first describe which actions are available to be performed by an agent and later use a type to specify whether it is Ego or Alter who can choose an action of this agent.

Agent forms. We view descriptions of interactive events performed by several agents as a class of game forms with a particular representation. Let us fix a number n of agents. We refer to a list of elements $x = (x_i)_{i \in [n]}$, one for each agent, as a *profile*. A *coalition* is a set of agents $C \subseteq [n]$; the *complementary coalition* is $-C := [n] \setminus C$. For a profile x and a coalition C , we write x_C to denote the list $(x_i)_{i \in C}$. Then, an *agent form* is a tuple $(S_1 \dots S_n, Z, \pi)$ where S_i is the set of actions available to agent i , Z is the set of possible outcomes, and $\pi : \times_{i=1}^n S_i \rightarrow Z$ is a partial play function. For each coalition $C \subseteq [n]$, we derive the set $S_C := \{s_C \mid s \in \times_{i=1}^n S_i\}$ of joint actions available to C . The agent form represents the untyped game form $\Gamma = (S, Z, \pi)$ over the set of strategies $S := \{S_C \mid C \subseteq [n]\}$. Types on the domain of n -agent forms correspond to coalitions of agents, hence, $\text{ACT} = 2^{[n]}$. Every coalition $C \subseteq [n]$ induces a type that is associated to the set S_C ; the dual type is associated to S_{-C} . Accordingly, the typed game form Γ_C describes the scenario in which player Ego acts *in the capacity* of the coalition C whereas Alter acts in the capacity of the complementary coalition. Thus, agent forms can be understood as a representation artifice to describe 2^n different game forms by one structure.

Effectivity functions and neighbourhood forms. The concept of effectivity function introduced by Moulin and Peleg [15] describes the power that a player has to force the outcome of a game within a target set. We assume the perspective of Ego when we refer to the effectivity of a game form. The *effectivity* $f(\Gamma)$ of a game form $\Gamma = (S^E, S^A, Z, \pi)$ (for Ego) is defined by

$$f(\Gamma) := \{X \subseteq Z \mid (\exists s \in S^E) (\forall t \in S^A) s \hat{\ } t \in X\}.$$

Clearly, the effectivity of a game is upwards closed in the sense that, with every set $X \in f(\Gamma)$, the closure $\lceil X \rceil := \{Y \mid X \subseteq Y \subseteq Z\}$ is included in $f(\Gamma)$.

When an untyped game form Γ is fixed, we write $f(\alpha)$ to denote the effectivity of Γ_α . Consider for example the game form described by the matrix in the left of Figure 1. For the two types selecting rows and columns, respectively, we obtain $f(\text{row}) = \lceil \{\{p, q\}, \{p, r\}\} \rceil$ and $f(\text{col}) = \lceil \{\{p\}, \{q, r\}\} \rceil$. We may also

view the matrix as a description of an agent form with, say, agent 1 in charge of selecting rows and agent 2 in charge of selecting columns. Then, there are four different types, one for each coalition $C \subseteq \{1, 2\}$. Besides $f(\{1\})$ and $f(\{2\})$ which coincide with $f(\text{row})$ and $f(\text{col})$, we obtain $f(\{1, 2\}) = [\{\{p\}, \{q\}, \{r\}\}]$ and $f(\emptyset) = \{\{p, q, r\}\}$.

Effectivity functions correspond to a particularly simple kind of game forms which we call *neighbourhood forms*. For a set Z of outcomes, a neighbourhood form is given by a set $F \subseteq 2^Z$. It describes the sequential scenario where Ego first chooses a set $X \in F$, then Alter chooses an element $x \in X$ which then constitutes the outcome of the game. For a fixed set ACT of types, we define untyped neighbourhood forms as the disjoint union of the typed neighbourhood forms over all types in ACT.

3 Logics and Models

The game forms discussed in Section 2 are concerned with the immediate outcome of interactive events. There is little to say, in logical terms, about such events in isolation. The challenge is to describe the dynamics of systems driven by sequences of interactive decisions. We focus on discrete systems that switch between states via transitions arising from the interplay of two competing players. In this section, we introduce extensive game structures as a generic model of such systems. After briefly describing syntax and semantics of Game Logic and Alternating Time Logics, we proceed to comparing the two logics. The key step is to show that GL and ATL* are both invariant under an equivalence which relates game structures of the same effectivity.

Extensive game structures. Extensive game structures generalise Kripke structures by replacing the accessibility relation with transition relations associated to effectivity functions. (Our model is close to the one proposed in [10] for plain ATL.)

Let ACT be a set of atomic game types closed under dual and let PROP be a set of atomic propositions. An *extensive game structure* for ACT and PROP is a structure $\mathcal{G} = (V, \Gamma, (V_p)_{p \in \text{PROP}})$ where V is a set of *positions*, Γ is a function that associates to every position v an untyped game form $\Gamma(v)$ for the domain ACT with outcomes in V , and V_p designates those positions where p holds. We will usually consider rooted structures with a designated initial position. Intuitively, taking a transition of type $\alpha \in \text{ACT}$ in state v of \mathcal{G} amounts to switching into the state resulting as an outcome of an (atomic) play between Ego and Alter in the typed game form $\Gamma(v)_\alpha$. By taking a sequence of such transitions, the players Ego and Alter form a path of infinite length to which we refer as a *global play*.

3.1 Parikh's Game Logic

The Propositional Logic of Games GL, introduced by Parikh in 1983 ([16, 17]), was the first logical formalism dedicated to reasoning about games. It proposes

a way of describing the dynamics of interaction in a way similar to the one in which PDL describes the dynamics of program execution.

The syntax of GL allows to compose interactive scenarios for two players. Starting from a set PROP of atomic propositions and a set ACT of atomic game types (or action names), the expressions of GL are of two sorts, formulae and game expressions. Formulae φ are constructed from PROP by Boolean operations and modalities $\langle \gamma \rangle \varphi$ associated to game expressions γ that are generated by the grammar: $\gamma := a \mid \varphi? \mid \gamma; \gamma \mid \gamma \cup \gamma \mid \gamma^* \mid \gamma^d$, for $a \in \text{ACT}$.

Informally, game expressions specify a schedule for a game between the two players Ego and Alter. The sequential composition $\gamma_1; \gamma_2$ means: play γ_1 first, then γ_2 . The nondeterministic choice operator $\gamma_1 \cup \gamma_2$ lets the player in turn decide which of γ_1 or γ_2 to play. The iteration operator γ^* allows to play γ repeatedly, for a finite number of times, whereby the player in turn can decide before each round whether a new round is to be played. Finally, the test operator ($\varphi?$) invokes a referee to verify whether φ holds; if so, the play just continues, otherwise it breaks and the player in turn loses. Within atomic game forms, the plays proceed sequentially: first, the player in turn chooses his part of the action, and then the other player responds with his part. At the beginning of a play, Ego is in turn to move.

The game-specific essence of Game Logic resides in the dualisation operator. Informally, this operator corresponds to a player-swapping rule which reverses the order of play and the set of strategies available to a player. At the atomic level, it thus corresponds to dualising the type of a game form.

The semantics of GL-expressions is defined on neighbourhood structures, i.e., extensive game structures where the game forms $\Gamma(v)$ are given by untyped neighbourhood forms. Statements about the models are constructed by associating these game expressions with modalities. A typical statement $\langle \gamma \rangle \varphi$ expresses that, at the current state, Ego has a strategy to play according to γ in such a way that either φ is true when the play ends, or Alter breaks a rule and loses. For a formal definition we refer the reader to [18].

3.2 Alternating-Time Logics

The framework of temporal logics, founded in the work of Pnueli and Manna [14] represents a way of adding recursion mechanisms to basic modal logic that is conceptually different from dynamic logics such as PDL and GL. While the latter assume an internal perspective, referring to the execution of a program or a protocol, temporal logics are geared towards analysing the behaviour of systems in the flow of time, referring to sequences of states in a run by isolating them from their originating context.

The formalisms of Alternating-Time Logics proposed by Alur, Henzinger, and Kupferman [2] adapts the temporal quantification pattern for the purpose of analysing interactive systems, typically multi-agent systems. The main representative of this logic ATL* is defined as an extension of branching-time logic CTL* by adding a game quantifier which allows to refer to a play formed by two strictly competing players in an underlying game structure.

The native models of Alternating-Time Logics are concurrent game structures, i.e., extensive game structures where the transitions are given by agent forms. For a set of atomic propositions PROP and a number n of agents, the formulae of ATL^* are of two sorts, state formulae φ and path formulae η , generated by the following grammars:

$$\varphi := \perp \mid p \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle\langle C \rangle\rangle\eta \quad \text{and} \quad \eta := \varphi \mid \eta \vee \eta \mid \neg\eta \mid \mathbf{X}\eta \mid \eta \mathbf{U} \eta$$

where $p \in \text{PROP}$, $C \subseteq [n]$.

Plain ATL is the fragment of ATL^* obtained by restricting the application of the operator $\langle\langle C \rangle\rangle$ to path formulae of type $\mathbf{X}\eta$ and $\eta \mathbf{U} \eta$. While not very expressive, this fragment is relevant because it is computationally tractable.

The meaning of ATL^* -formulae in an extensive game structure \mathcal{G} is defined by mutual induction over path and state formulae. Path formulae are interpreted over traces of plays in \mathcal{G} according to the rules for linear temporal logic LTL with the constructors $\mathbf{X}\eta$ and $\eta \mathbf{U} \eta$ corresponding to the LTL-operators next and until, respectively. The quantifier $\langle\langle C \rangle\rangle$ transforms any path formula η into a state formula $\langle\langle C \rangle\rangle\eta$ which holds at those positions v from which, player Ego acting in capacity of coalition C has a strategy to force an infinite play which satisfies η .

It is important to remark that strategies of Ego are functions that associate to every initial segment π of a play, an action in the game form of type C reached in the play. In the extensive game over \mathcal{G} with η describing the winning outcomes, Ego can force a win if, and only if, he can force a win while playing such that in every atomic game form, he moves first and makes his choice visible to Alter.

3.3 Comparing GL and ATL^* .

A priori, GL and ATL^* are interpreted on different kinds of extensive game structures. To relate the two logics, we need to establish a correspondence between concurrent game structures and neighbourhood structures that is compatible with the logic. Minimal requirements on such a model correspondence would be (1) to relate two formulae $\varphi \in \text{ATL}^*$ and $\varphi' \in \text{GL}$ if, for all concurrent game structures \mathcal{G} and all corresponding neighbourhood structures \mathcal{G}' , we have $\mathcal{G} \models \varphi$ if and only if $\tilde{\mathcal{G}}' \models \varphi'$, and (2) to respect the Boolean and modal operators common to the two logics.

In the following, we characterise a much stronger model correspondence. Towards this, we introduce an equivalence between general extensive game structures under which both GL and ATL^* are invariant, and we show that each class of equivalent extensive structures has a representative among neighbourhood structures.

The idea is to identify each concurrent game structure $\mathcal{G} = (V, \Gamma, (V_p)_{p \in \text{Prop}})$ for n agents with the neighbourhood structure $\tilde{\mathcal{G}}$ obtained by replacing every (untyped atomic) game form $\Gamma(v)$ with the neighbourhood form corresponding to the effectivity of $\Gamma(v)$. We justify this identification by showing that ATL^* -formulae cannot distinguish between \mathcal{G} and $\tilde{\mathcal{G}}$. This allows us to reduce the

interpretation domain of ATL* without loss to neighbourhood domains —the interpretation domain of GL— over the set of types $\text{ACT} = 2^{[n]}$ corresponding to coalitions of agents. Over this restricted domain, we can compare the expressiveness of ATL* speaking about coalitions of n agents with the expressiveness of GL* speaking about a set of 2^n atomic game actions associated to agent coalitions.

The difficulty consists in defining the effectivity of an untyped game form (where it is not yet known which actions belong to a player) in such a way that the meaning of all its typed instantiations (where actions are readily assigned to players) are preserved. Our approach involves the notion of sequentialisation of a game form, which captures the situation where the players perform their choice in a given order.

Sequentialisation. Any game form Γ naturally gives rise to two *sequential* game forms Γ_E and Γ_A . The game form Γ_E correspond to the scenario where Ego chooses his action first, and then Alter chooses his action being informed about Ego’s choice. Conversely, in Γ_A , Alter chooses first and then Ego follows. We are interested, more generally, in the set of all sequential scenarios that may arise from an untyped game form $\Gamma = (S, Z, \pi)$, where strategies are not yet associated to a particular player. To capture the flow of information from the (yet unknown) first to the second mover we extend the set of available strategies to include all perfect-information strategies over choices from S . Formally, we consider the untyped game form $\hat{\Gamma} = (\hat{S}, Z, \hat{\pi})$ with strategies $\hat{S} = S \cup S^S$, where S^S denotes all functions from S onto S . The play function $\hat{\pi}$ is derived from π by setting $\hat{\pi}(s, t)$ to $\pi(s, t(s))$ if $(s, t) \in S \times S^S$, or to $\pi(s(t), t)$ if $(s, t) \in S^S \times S$; otherwise the value is left undefined. Each type α for Γ induces two types for the new game form, $E : \alpha$ and $A : \alpha$, which correspond to the scenarios in which Ego or Alter moves first, respectively. Thus, the new types assign to Ego the strategy sets $\hat{S}_{E:\alpha} := S_\alpha$ and $\hat{S}_{A:\alpha} := (S_\alpha)^{S-\alpha}$, respectively. We will call these types *sequential types*, and refer to $\hat{\Gamma}_{E:\alpha}$ and $\hat{\Gamma}_{A:\alpha}$, simply denoted $\Gamma_{E:\alpha}$ and $\Gamma_{A:\alpha}$, as *sequentialisations* of Γ . Observe that the dual of a sequential type $E : \alpha$ is $A : -\alpha$ which swaps both the sets of available actions and the play order of Ego and Alter.

By definition, effectivity functions do not distinguish between a game form Γ_α and its sequentialisation $\Gamma_{E:\alpha}$ with Ego as first mover, that is, $f(E:\alpha) = f(\alpha)$. Moreover, the effectivity of sequential types exhibits the following duality.

Lemma 1. *For any game form Γ and every appropriate type α ,*

$$\begin{aligned} f(-E:\alpha) &= f(A:-\alpha) = \{ X \subseteq Z \mid (\forall t \in S_\alpha^A)(\exists s \in S_{-\alpha}^E) s \hat{\sim} t \in X \} \\ &= \{ Z \setminus X \mid X \notin f(E:\alpha) \}. \end{aligned}$$

Consequently, the set of sequentialisations of an untyped game form is characterised by the effectivities of the scenarios where Ego moves first.

Neighbourhood representation. In Section 2 we illustrated that effectivity functions correspond to (typed) sequential game forms. Conversely, Lemma 1

points out that sequential game forms can be represented by a set of effectivity functions. In the following, we introduce untyped game forms that embed such a set of effectivity functions into one representation.

For an untyped game form Γ over a set of types ACT , let us consider the graph G_α representing the sequential form associated to the effectivity of Γ_α where we label all arcs emanating from the root by α and the remaining arcs by $-\alpha$. Now, we merge all the graphs G_α for $\alpha \in \text{ACT}$, by joining their roots and the terminal nodes that correspond to the same outcome. The resulting graph can again be viewed as an untyped sequential game form $\tilde{\Gamma}$, which we call the *neighbourhood representation* of Γ . The meaning of types for $\tilde{\Gamma}$ is determined by the arc labels; every type α corresponds to the set of strategies that select an α -successor for each node. The play function $\tilde{\pi}(s^E, s^A)$ for $\tilde{\Gamma}_\alpha$ returns the terminal node reached by moving first to the α -successor selected by s^E and then to the $(-\alpha)$ -successor selected by s^A . The construction is illustrated in Figure 1.

As the following lemma points out, the neighbourhood representation $\tilde{\Gamma}$ preserves the effectivity of all types for the original game form Γ .

Lemma 2. *Let Γ be an untyped game form and let $\tilde{\Gamma}$ be its neighbourhood representation. Then, $f(\Gamma_\alpha) = f(\tilde{\Gamma}_\alpha)$ for all types α .*

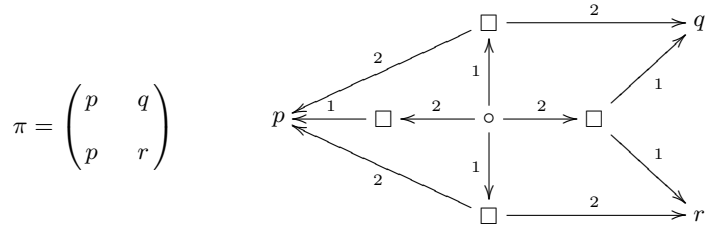


Fig. 1. A game form and its neighbourhood representation (only minimal effectivity sets are shown and the trivial types \emptyset and $\{1, 2\}$ are omitted)

Effectivity equivalence. Lemma 2 suggests a canonical representation of game forms in terms of neighbourhood forms. To make this idea precise, let us fix a modelling domain with a set ACT of types. We say that two games forms Γ and Γ' with the same set of outcomes are *effectivity-equivalent* if their effectivities $f(\Gamma)$ and $f(\Gamma')$ coincide. Likewise, two untyped game forms Γ and Γ' are effectivity equivalent if the game forms Γ_α and Γ'_α are so, for all types $\alpha \in \text{ACT}$.

Due to the fact that effectivity functions preserve the duality of sequential types $A: \alpha$ and $E: -\alpha$, it follows that the effectivity equivalence between untyped game forms extends to their sequentialisations.

Lemma 3. *If two untyped game forms Γ and Γ' are effectivity equivalent, then so are their sequentialisations i.e., $f(\Gamma_{i:\alpha}) = f(\Gamma'_{i:\alpha})$, for every type $\alpha \in \text{ACT}$ and each player $i \in \{E, A\}$.*

In particular it follows that, no matter whether a sequentialisation is applied to a game form or to its neighbourhood representation, the resulting sequential forms are equivalent.

Finally, we lift the notion of effectivity-equivalence to extensive game structures. We say that two extensive game structures $\mathcal{G} = (V, \Gamma, (V_p)_{p \in \text{PROP}})$ and $\mathcal{G}' = (V, \Gamma', (V_p)_{p \in \text{PROP}})$ over the same sets of positions V and with the same valuations V_p are effectivity-equivalent, if for any state v the game forms $\Gamma(v)$ and $\Gamma'(v)$ are effectivity equivalent. Notice that this is the same as requiring that \mathcal{G} and \mathcal{G}' have the same neighbourhood representation.

Theorem 4. *The logics GL and ATL* are invariant under effectivity equivalence: For any pair of effectivity-equivalent extensive game structures \mathcal{G} and \mathcal{G}' , we have $\mathcal{G} \models \varphi$ iff $\mathcal{G}' \models \varphi$, for any formula φ of GL or ATL*.*

Proof. The proof is by induction over the structure of formulae. The critical case regards the modal next-step operators $\langle\langle C \rangle\rangle X$ and γ of ATL* and GL, respectively. (When speaking about modal operators of ATL*, we tacitly mean the modal operators of the Alternating-Time μ -Calculus in which ATL* is embedded [2].)

Towards an operational characterisation of effectivity equivalence, we define *simulation* relations that capture the ability of a player to transfer his strategy from one game to another one in a way that maintains the same outcome on both sides. We say that, for Ego, the game form $\Gamma = (S^E, S^A, Z, \pi)$ is simulated by the game form $\Gamma' = (S'^E, S'^A, Z, \pi')$, and we write $\Gamma \preceq^E \Gamma'$, if for every $s \in S^E$ there exists $s' \in S'^E$ such that for every $t' \in S'^A$ there exists $t \in S^A$ for which $s \hat{=} t = s' \hat{=} t'$. We write $\Gamma \sim^E \Gamma'$, if $\Gamma \preceq^E \Gamma'$ and $\Gamma' \preceq^E \Gamma$. For Alter, the notions are defined analogously.

Then, for any pair Γ, Γ' of untyped game forms over the same set of types ACT and with the same sets of outcomes, we have:

- (i) For any type α , the forms Γ_α and Γ'_α are effectivity-equivalent if, and only if, $\Gamma_\alpha \sim^E \Gamma'_\alpha$.
- (ii) As untyped game forms, Γ and Γ' are effectivity-equivalent if, and only if, $\Gamma_\alpha \sim^E \Gamma'_\alpha$ and $\Gamma_\alpha \sim^A \Gamma'_\alpha$, for all types α .

Accordingly, if two extensive game structures \mathcal{G} and \mathcal{G}' over the same set of positions V are effectivity-equivalent, the simulation relation between atomic games $\Gamma(v)$ and $\Gamma'(v)$, for all $v \in V$ extends naturally to a simulation relation between the structures. Essentially, every game composed via operators of ATL* or GL can be played on \mathcal{G} in the same way as it can be played on \mathcal{G}' . \square

Since every game structure is effectivity-equivalent to its neighbourhood representation, we obtain the following corollary.

Corollary 5. *A formula of GL or ATL* holds in an extensive game structure if, and only if, it holds in its neighbourhood representation.*

We can associate to any concurrent game structure \mathcal{G} its neighbourhood representation $\tilde{\mathcal{G}}$ to define an appropriate correspondence \rightsquigarrow between formulae

$\varphi \in \text{ATL}^*$ and $\psi \in \text{GL}$ by setting $\varphi \rightsquigarrow \psi$ whenever $\mathcal{G} \models \varphi$ iff $\tilde{\mathcal{G}} \models \psi$. Beyond respecting Boolean operations, this correspondence has the property that $\varphi \rightsquigarrow \psi$ and $\varphi' \rightsquigarrow \psi$ implies $\varphi \equiv \varphi'$. This allows us to extend the interpretation of Game Logic to concurrent game structures \mathcal{G} by assigning to any formula $\varphi \in \text{GL}$ its meaning over the neighbourhood representation $\tilde{\mathcal{G}}$ which will finally enable us to compare the two logics.

4 Recursion mechanisms

In this last part of the paper, we sketch a direction for investigating the equivalence between formulae of GL and ATL^* in terms of automata. In the previous section, we have seen that the basic modal operators of ATL and GL are essentially equivalent on extensive game structures with a set of types that are adequate for agent forms. To analyse the recursion mechanisms of GL and ATL^* we now translate both logics into automata that run over extensive game structures. We call these *game automata*, because they operate with transitions determined by atomic game forms.

Game Automata. A game automaton for a set PROP of propositions and a set ACT of types is a tuple

$$\mathcal{A} = (Q := Q^E \dot{\cup} Q^A, \text{PROP}, \text{ACT}, q_{\text{I}}, \delta, \Omega),$$

where Q is a finite state set with partitions Q^E and Q^A controlled by Ego and Alter, respectively, $q_{\text{I}} \in Q$ is an initial state, $\delta : Q \times 2^{\text{PROP}} \rightarrow Q \times Q \cup \text{ACT} \times Q$ is a transition function, and $\Omega : Q \rightarrow \mathbb{N}$ is a priority function describing a parity acceptance condition. Intuitively, the run of the automaton on an input structure \mathcal{G} corresponds to a play of possibly infinite duration between Ego and Alter, starting from state q_{I} and the initial position v_0 of \mathcal{G} . From a state q and a position v , a transition $\delta(q, P)$ is enabled if the predicates in $P \subseteq \text{PROP}$ match those that hold at v ; the player who controls the current state is in charge of the transition: if $\delta(q, P) = (q', q'')$, he has to choose between switching the automaton into state q' or into state q'' ; otherwise, if $\delta(q, P) = (\alpha, q')$, the player who controls q first performs an action of type α in the game form $\Gamma(v)$, and then the other player performs an action of the dual type $-\alpha$. The outcome of this local play determines the new position in \mathcal{G} , while the automaton is switched into q' . Finally, the game structure \mathcal{G} is accepted, if Ego has a strategy to ensure that the sequence of states visited during the play satisfies the following parity property: the least priority occurring infinitely often is even.

Formally, acceptance is defined in terms of a graph game between Ego and Alter on the synchronised product between \mathcal{A} and \mathcal{G} . The only non-standard element of this definition regards the intermediary configuration reached after an α -action has been executed by one player (and before the dual action is executed by the opponent) which does not correspond to any state-position pair. This intermediary state can be represented, by the set of all possible outcomes

of the action (of type $-\alpha$) that the second mover has to take. The acceptance game is thus a classical graph game [11].

From Game Logic to automata. To translate a GL-formula into a game automaton, we first transform it into a pure game expression $\langle \gamma \rangle \text{true}$ from which we also eliminate all non-atomic test operations; next, we put the game expression into a normal form in which every operation is associated explicitly to a player $i \in \{\text{Ego}, \text{Alter}\}$ (see [3] for details). Notice that these operations only amount to relabellings on the syntax graph of the original formula which leaves its structure essentially unchanged.

Now, we build an automaton $\mathcal{A}(\gamma)$ inductively as illustrated in Figure 2; the states drawn in dotted frames are coalesced. Note that each component in this construction has a single entry (marked \bullet) and a single exit (marked \circ). Entry states are assigned to the player i in control of the corresponding subexpression. Significant priorities are assigned to states corresponding to \star -iteration operators. According to whether the iteration is controlled by Ego or Alter, the priority is even or odd, respectively, and the priority of a \star -expression is lower than that of all its subexpressions.

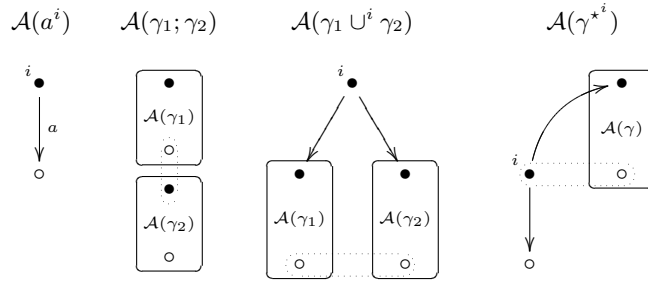


Fig. 2. Translating Game Logic into automata

The construction shows that GL-formulae translate into game automata where each component has a single entry and a single exit. In terms of programming-language theory, the interactive program constructions featured in GL are well structured. It is easy to show that every automaton with a transition graph that is well structured can be conversely translated into GL.

Proposition 6. *A class of extensive-form game models can be defined in Game Logic if, and only if, it can be described by a game automaton with a single-entry single-exit transition graph.*

To summarise, the higher-level quantification pattern of Game Logic corresponds to well-structured transition graphs. This structural restriction witnesses an expressive weakness of GL. It shows, for instance, that it is impossible to describe in GL extensive game models that embed a clique of size at least 3 ([4]). Thus, Game Logic is less expressive than the Alternating-Time μ -calculus.

From ATL to automata. To translate a typical ATL*-formula $\langle\langle C \rangle\rangle\eta$ into a game automaton, we construct first the automata \mathcal{A}_φ corresponding to the direct state subformulae φ of η . Next, we substitute all these state subformulae in η with fresh propositional variables X_φ as placeholders. The formula η' obtained in this way can be regarded as a linear-time expression over these variables; now, we consider a deterministic word automaton $\mathcal{A}_{\eta'}$ recognising the language of η' . This automaton we transform into a game automaton, by replacing each forward transition on the word model with an atomic modality corresponding to a game form of type C . Finally, we replace the tests for variables X_φ with a transition into the corresponding automaton \mathcal{A}_φ .

Analysing the structure of the automata obtained when translating plain ATL, the restricted variant of ATL where path formulae cannot be nested, it turns out that one obtains single-entry single-exit transition graphs. As a consequence of this translation and of Proposition 6, it thus follows that GL subsumes ATL.

Corollary 7. *Every formula φ of plain Alternating-Time Logic ATL can be translated into an equivalent Game Logic formula of size $\mathcal{O}(|\varphi|)$.*

The translation makes several expressive restrictions of ATL* apparent. For instance, every strongly connected component of the automaton obtained for an ATL* refers only to one kind of atomic types. Thus, one cannot express for instance, that agent 1 has a strategy to reach a state with property p in a play where he may form coalitions either with agent 2 or with agent 3, which is expressible in GL by $\langle\langle (1, 2) \cup (1, 3) \rangle\rangle^* p$.

On the other hand, we conjecture that GL cannot express all properties expressible in ATL*. A promising source of inspiration towards settling this issue is the research on non-ambiguous regular expressions (see, e.g. [6]). Intuitively, a regular expression is non-ambiguous if every word can be matched in at most one way to expression symbols while it is read. An example of an inherently ambiguous property over the set set of predicates $\{0, 1, 2\}$ is that infinitely often the symbol 2 is seen 2 steps before the symbol 0 occurred. Whether Ego is able to enforce a path with this property seems unlikely to be expressible in GL, whereas it is clearly expressible in ATL*.

5 Conclusion

We set out to compare two prominent formalisms for reasoning about games that are historically remote and emerged from different operational paradigms. Parikh's Game Logic purports an internal perspective on the execution of an interactive program, whereas the family of Alternating Time Logic of Alur, Henzinger, and Kupferman reflect an external perspective on computations in a concurrent multi-agent systems.

By rephrasing the semantics of the two formalisms in unified framework, we point out that they show remarkable similarities: at the atomic level, the differences are limited to representation aspects, whereas at the global level, both formalisms have limitations due to recursion mechanisms which can be explained

in terms of structural properties of game automata. Through our analysis, we reduce the question about how the two logics differ in their expressive power to questions about automata with a restricted transition structure.

References

1. T. ÅGOTNES, V. GORANKO, AND W. JAMROGA, *Alternating-time temporal logics with irrevocable strategies*, in TARK '07: Proc. 11th Conference on Theoretical Aspects of Rationality and Knowledge, 2007, ACM, pp. 15–24.
2. R. ALUR, T. A. HENZINGER, AND O. KUPFERMAN, *Alternating-time temporal logic*, J. ACM, 49 (2002), pp. 672–713.
3. D. BERWANGER, *Game Logic is Strong Enough for Parity Games*, Studia Logica, 75 (2003), pp. 205–219.
4. D. BERWANGER, E. GRÄDEL, AND G. LENZI, *The variable hierarchy of the μ -calculus is strict*, Theory of Computing Systems, 40 (2007), p. 437466.
5. TH. BRIHAYE, A. DA COSTA, F. LAROUSSINIE, AND N. MARKEY, *ATL with strategy contexts and bounded memory*, Research Report LSV-08-14, Laboratoire Spécification et Vérification, ENS Cachan, 2008.
6. A. BRUGGEMANN-KLEIN AND D. WOOD, *One-unambiguous regular languages*, Information and computation, 142 (1998), pp. 182–206.
7. D. C. DENNETT, *The intensional stance*, MIT Press, 1989.
8. E. A. EMERSON AND J. Y. HALPERN, *“Sometimes” and “not never” revisited: On branching versus linear time*, in Conference Record of the 10th Annual ACM Symposium on Principles of Programming Languages, POPL '83, ACM, 1983, pp. 127–140.
9. M. FISCHER AND R. LADNER, *Propositional dynamic logic of regular programs*, Journal of Computer and System Sciences, 18 (1979), pp. 194–211.
10. V. GORANKO AND G. VAN DRIMMELEN, *Complete axiomatization and decidability of alternating-time temporal logic*, Theor. Comput. Sci., 353 (2006), pp. 93–117.
11. E. GRÄDEL, W. THOMAS, AND T. WILKE, eds., *Automata, Logics, and Infinite Games*, vol. 2500 of LNCS, Springer, 2002.
12. J. HINTIKKA AND G. SANDU, *Game-theoretical semantics*, in Handbook of Logic and Language, Elsevier and MIT Press, 1997, pp. 361–340.
13. N. HOWARD, *Paradoxes of Rationality: Theory of Metagames and Political Behavior*, MIT Press, 1971.
14. Z. MANNA AND A. PNUELI, *Temporal Verification of Reactive Systems: Specification*, Springer-Verlag, 1992.
15. H. MOULIN AND B. PELEG, *Cores of effectivity functions and implementation theory*, Journal of Mathematical Economics, 10 (1982), pp. 115–145.
16. R. PARIKH, *Propositional Game Logic*, in IEEE Symposium on Foundations of Computer Science, IEEE, 1983, pp. 195–200.
17. R. PARIKH, *The logic of games and its applications*, Annals of Discrete Mathematics, 24 (1985), pp. 111–140.
18. M. PAULY, *From Programs to Games: Invariance and Safety for Bisimulation*, in Proc. 14th Annual Conference of the European Association for Computer Science Logic CSL 2000, vol. 1862 of LNCS, Springer-Verlag, 2000, pp. 586–496.
19. J. VAN BENTHEM, *Modal frame correspondences and fixed-points*, Studia Logica, 83 (2006), pp. 133–155.
20. W. VAN DER HOEK AND M. PAULY, *Modal logic for games and information*, in Handbook of Modal Logic, Elsevier, Amsterdam, 2007, pp. 1078–1143.