Using Locales to Define a Rely-Guarantee Temporal Logic

William Mansky, Elsa L. Gunter

Department of Computer Science, University of Illinois at Urbana-Champaign,
Thomas M. Siebel Center, 201 N. Goodwin, Urbana, IL 61801-2302
{mansky1,egunter}@illinois.edu

Abstract. In this paper, we present an agent-based logic called Rely-Guarantee Temporal Logic (RGTL), developed using the Isabelle theorem prover. RGTL provides a formalism for expressing complex temporal-logic specifications of multi-agent systems, as well as a compositional method of reasoning about the dependencies between components in such a system. Taking advantage of Isabelle's locale functionality, we are able to express various choices about the notion of "strategy" used in the logic (e.g., memoryless/memory-based) as parameters to the semantics, whereas previously these choices were considered to define semantics for distinct variants of agent-based logics. We can then state and formally verify various aspects of RGTL, including its reasoning principles and its expressiveness relative to Alternating-time Temporal Logic (ATL), independently of the type of underlying strategies, by using locales to axiomatize the necessary requirements on strategies.

Keywords: logics for agency, temporal logic, reasoning about strategies, modular specification, Isabelle proof assistant

1 Introduction

Alternating-time temporal logic (ATL) [**?**] is an extension of temporal logic to a system with multiple *players*, agents whose choices influence the evolution of a system. By introducing quantification over the *strategies* defining the future actions of some set of players, ATL provides a mechanism for formulating properties of the form "A can guarantee φ" or "A must allow φ". However, when dealing with systems containing multiple components working in concert, "can guarantee" and "must allow" are of less interest than "will guarantee" or "does not guarantee". These properties can be expressed, for instance, using the ATL-STIT language proposed by Broersen et al. [**?**], where STIT is an acronym for "sees to it that". Rely-Guarantee Temporal Logic (RGTL) expands on this approach, providing a formalism for expressing temporal-logic properties of and dependencies between components, as well as generalizing the notion of "strategy" from a deterministic function on states and agents to a range of potentially nondeterministic, progressively refined objects. RGTL is a logic designed to support the concepts of rely-guarantee reasoning and agency as first-class concepts, providing a flexible logic for specifying and checking requirements on complex multi-component systems.

The design of the semantics of RGTL was done using the Isabelle proof assistant, and in particular takes advantage of Isabelle's *locale* facility, which provides a mechanism for collecting and manipulating the assumptions required by various theories [**?**]. Through successive layers of locales, we build up the necessary framework for defining RGTL, including the underlying automata (concurrent game structures), a fundamental notion of strategies, and various axioms and operations on strategies. By minimizing the assumptions made in any given locale, we can give a general statement of the logic, independent of various distinctions that in past work have been considered to define different logics. For example, ATL with irrevocable strategies [**?**] has been defined in two variants, IATL (in which strategies are memoryless) and MIATL (in which strategies have unbounded memory). By abstracting away from the details of strategy computations, we are able to give a single definition of RGTL for both memoryless and memory-based strategies (as well as various other potential distinctions), which can be specialized to either case by plugging in the corresponding sublocale. Using the same approach in our analysis of expressiveness, we are able to prove that RGTL is more expressive than ATL* regardless of the type of strategies used, as long as the type of strategies is consistent across the two logics.

2 Example: A First Look

To understand the extra flexibility afforded to us by RGTL over ATL, let us consider a simple example with two agents, A and B, and a system. The system offers to each agent a toggle, which, at each instant, the agent associated with the toggle may either push or leave alone. The toggles jointly control whether a light is on or off. If, in a given instant, just one agent pushes their toggle, the light will change state: if it was on it will go off, and if it was off it will go on. If both agents either leave their toggles alone, or simultaneously push them, the light will not change state: if it was on, it will stay on, and if it was off, it will stay off.

Now let us consider the property P that at some point the light will be on and remain on from that point forward. In Linear Temporal Logic (LTL) [**?**], this can be stated as $\Diamond\,\Box\,\mathsf{light\_on}$, i.e., "eventually always the light is on". Obviously, if the two agents are free at each instant to choose whether to push the toggle or not, the system will display some traces that satisfy this property, but also many that do not. If we want to know whether the two players can collaborate to assure P, then we are effectively asking if there exists a trace satisfying P, which is a property that can be expressed in branching-time temporal logics such as CTL* [**?**]. However, if we wish to focus on what one agent can control without joint collaboration with the other, we are unable to prove any meaningful results. In particular, speaking in ATL terms, it should be clear that a single agent cannot guarantee P, and indeed must allow $\neg P$ (written as $[\![A]\!]\Box\Diamond\neg\mathsf{light\_on}$). No matter what strategy agent A pursues, there is a way for agent B to mess things up.

However, were agent A able to make use of certain properties of the behavior of agent B, then it might be possible for A to craft a strategy to always guarantee P, even if A did not know exactly what B would do at any given instant. For example, if B could definitely be relied upon to eventually stop toggling, then the strategy for A to always push the toggle on when the light is off will guarantee that eventually the light will be on and stay on. It is this kind of conditional component-wise reasoning that we aim to express and support in RGTL.

3 RGTL Syntax

Intuitively, the ATL path quantifier $\langle\langle A\rangle\rangle$ allows us to express "can-guarantee" properties; $\langle\langle A\rangle\rangle\varphi$ holds of a system when there is *some* strategy for A that ensures φ (despite the actions of the remaining agents). The dual operator, $[\![A]\!]\varphi \equiv \neg\langle\langle A\rangle\rangle\neg\varphi$, holds when for *any* strategy for A, the remaining agents can ensure φ; this can be intuitively understood as a "must-allow" property. In the case in which we have an existing strategy on which we want to check properties, neither of these operators provides the correct formalism.

Instead, we would like to say that a program *does* satisfy a property, and more generally that agent a satisfies some property P_a as long as agent b satisfies its own property P_b, which may be thought of as the *protection envelope* for agent b. The concept of the protection envelope appears in the work of Gunter et al. [?]. While it may be possible to show that a particular workflow for b satisfies a desired property, minor variations in the workflow for b may violate the property. The protection envelope is a more general property that may be satisfied by variations on b's expected workflow, while providing enough information to ensure safety of the overall system. The $\overset{A}{\Rrightarrow}$ operator is designed to facilitate this style of system specification: the left-hand side of the implication is the protection envelope for A, and the right-hand side is the property enabled by this envelope. Because of its similarity to the rely-guarantee approach originally proposed by Jones [?], we refer to this operator as the "rely-guarantee arrow".

Following CTL* and ATL*, an RGTL formula is either a *state formula* or a *path formula*; the semantics of a state formula depends only on information about the current state, while the semantics of a path formula includes assertions on possible future states. The path and state formulae of RGTL defined as:

$$\varphi ::= \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \psi$$

$$\psi ::= \mathsf{true} \mid \pi \mid \psi \wedge \psi \mid \neg\psi \mid \psi \overset{A}{\Rrightarrow} \psi \mid \mathit{\Lambda}\varphi$$

where $\pi \in \mathit{\Pi}$ is an atomic proposition and $A \subseteq \mathcal{A}$ is a set of agents. As in LTL, φ (read "next φ") asserts that φ holds in the next state along a path, while $\varphi_1\,\mathcal{U}\,\varphi_2$ (read "φ_1 until φ_2") asserts that φ_1 holds at every point along the path until the first point at which φ_2 holds. RGTL also includes the rely-guarantee operator $\overset{A}{\Rrightarrow}$, and the $\mathit{\Lambda}$ operator, which quantifies over the outcomes of a strategy. The semantics of these operators is given in the following section.

4 Semantics

4.1 Concurrent Game Structures and Strategies

The semantics of RGTL in Isabelle is built up through a series of nested locales. Each locale introduces a set of objects and axioms defining one of the concepts needed to give semantics to RGTL formulae, and provides useful constructs and lemmas for working with that concept. Ideally, each locale introduces exactly the assumptions needed for the definitions and proofs it provides. Through this approach, our semantics remains agnostic of the underlying implementation of various features of the semantics, in particular that of the type of *strategies* introduced below.

The first locale, `CGS`, introduces the concept of a concurrent game structure, a type of automata that moves from state to state according to the actions of a set of agents. The semantics of ATL and related logics, including RGTL, are evaluated with concurrent game structures as their underlying automata.

Locale Definition 1 *A concurrent game structure (abbreviated CGS) is a tuple* $(\mathcal{A}, Q, \Pi, \pi, \Sigma, e, \delta)$*, where* $\mathcal{A}$ *is a finite and non-empty set of agents (also called players),* Q *is a finite set of states,* Π *is a finite set of atomic propositions,* π *is a labeling function from each state* $q \in Q$ *to the set of atomic propositions that hold in* q*,* Σ *is a finite set of actions available to the agents,* $e : \mathcal{A} \times Q \to 2^{\Sigma}$ *is a function that gives the (non-empty) set of enabled actions for each combination of agent and state, and* $\delta : Q \times \Sigma^{\mathcal{A}} \to Q$ *defines the transitions between states based on the actions of each agent.*

As in other agent-based logics, satisfaction of an RGTL formula is defined in terms of *strategies* for sets of agents and the *outcomes* of those strategies. Various definitions of strategies have been presented for ATL; for instance, strategies may have no memory, bounded memory, or unbounded memory [**?**], and may be deterministic or nondeterministic [**?**]. In the strategy locale for RGTL, `CGS_strategies`, we give an extremely general definition of strategies, and require only the operations needed to define the semantics of strategy-based logics.

Locale Definition 2 *Let* $(\mathcal{A}, Q, \Pi, \pi, \Sigma, e, \delta)$ *be a CGS.*

Let R *be the type of* state information*, which supports the operations* $\mathsf{current\_state}(\rho)$*,* $\mathsf{init}(q)$ *(creation of initial state information), and* $\rho \cdot q$ *(update with a new state).*

A strategy *is an object supporting the function* $[\![\_]\!] : \mathcal{A} \times R \to 2^{\Sigma}$ *such that for all agents* a *and state information* ρ*,* $[\![S]\!](a, \rho)$ *is a non-empty subset of* $e(a, \mathsf{current\_state}(\rho))$*.*

Intuitively, $[\![S]\!](a, \rho)$ is the set of actions allowed by S for agent a, given knowledge ρ of past states. In a concrete instance, ρ may be a state, finite history, or infinite history; for example, we can obtain memoryless strategies by taking $R = Q$ and letting $\mathsf{current\_state}(\mathsf{q}) = q$, $\mathsf{init}(q) = q$, and $q \cdot q' = q'$.

Given these axioms, we can define the following constructs on strategies.

Definition 1. *A strategy S is* deterministic *if for each agent a, either $|[\![S]\!](a,\rho)| = 1$ for all ρ or else $[\![S]\!](a,\rho) = e(a, \mathsf{current\_state}(\rho))$ for all ρ.*

A deterministic strategy is one that either completely determines the actions of an agent, or else places no restrictions on it. This is the type of strategies used in the original definition of ATL [**?**]; in general, RGTL strategies may offer any number of choices for each agent.

Definition 2. *The outcomes* out *of a strategy S and state information ρ are defined as $\mathsf{out}(S,\rho) = \{\lambda.\ \lambda_0 = \mathsf{current\_state}(\rho) \wedge \forall i.\ \exists \overline{\sigma}.\ (\forall a.\ \sigma_a \in [\![S]\!](a, \rho \cdot \lambda_{[1,i]})) \wedge \lambda_{i+1} = \delta(\lambda_i, \overline{\sigma})\}$, where $\overline{\sigma}$ is a vector of actions, one for each agent. We write λ_i for the i^{th} element of the sequence λ, $\lambda_{[i,j]}$ for the subsequence of λ starting at the i^{th} element and ending at the j^{th} element (or the empty sequence when $j < i$), and $\rho \cdot \lambda$ for ρ updated with the elements of λ.*

An infinite path λ through the underlying CGS is an *outcome* of a strategy S given state information ρ if λ starts in the current state $\mathsf{current\_state}(\rho)$ and there is a way to proceed from each state in λ to the next that is allowed by S. Note that in the case where S is deterministic and ρ is a single state q, this corresponds exactly to the original ATL definition of outcomes. As outcomes are infinite sequences of states, we make use of the theory of infinite lists from the Archive of Formal Proofs [**?**] to help us reason about them in Isabelle.

Definition 3. *We say that a strategy T is a* refinement *of a strategy S, written $T \sqsubseteq S$, when $[\![T]\!](a,\rho) \subseteq [\![S]\!](a,\rho)$ for each agent a and state information ρ.*

We also refer to a strategy T such that $T \sqsubseteq S$ as a *sub-strategy* of S. As one might expect, reducing the nondeterminism of a strategy reduces the set of outcomes; this result follows directly from the relevant definitions.

Lemma 1. *If $T \sqsubseteq S$, then $\mathsf{out}(T,\rho) \subseteq \mathsf{out}(S,\rho)$ for any ρ.*

As part of the `CGS_strategies` locale, we also assume several methods of deriving strategies from existing strategies, forming an implementation-agnostic algebra of strategies. The first such axiom allows us to derive strategies that ignore or presume particular state information. This will be of particular use in showing the relationship between RGTL and ATL (Section **??**).

Locale Axiom 1 *In `CGS_strategies`, for any strategy S and state information ρ, there is a strategy T such that $\forall a\ \lambda.\ [\![S]\!](a, \rho \cdot \lambda) = [\![T]\!](a, \mathsf{init}(\mathsf{current\_state}(\rho)) \cdot \lambda)$, and a strategy R such that $\forall a\ \lambda.\ [\![S]\!](a, \mathsf{init}(\mathsf{current\_state}(\rho)) \cdot \lambda) = [\![T]\!](a, \rho \cdot \lambda)$.*

Second, we assume that given a potentially nondeterministic strategy with a range of possible outcomes, we can pick out a sub-strategy that produces any particular outcome.

Locale Axiom 2 *In `CGS_strategies`, for any outcome $\lambda \in \mathsf{out}(S,\rho)$, there is a strategy T such that $T \sqsubseteq S$ and $\mathsf{out}(T,\rho) = \{\lambda\}$.*

4.2 Strategies in RGTL

While these definitions are sufficient to allow us to talk about strategies and satisfaction for ATL and its variants, RGTL requires several additional strategy operators. We axiomatize these operators in the `RGTL_semantics` locale.

Locale Definition 3 $\top$ *is a strategy such that for any agent a and state information ρ, $[\![\top]\!](a, \rho) = e(a, \mathsf{current\_state}(\rho))$.*

Given a strategy S for the system and a set of agents $A \subseteq \mathcal{A}$, we define the restriction *of S to A by*

$$[\![S|_A]\!](a, \rho) = \begin{cases} [\![S]\!](a, \rho) & a \in A \\ e(a, \mathsf{current\_state}(\rho)) & a \notin A \end{cases}$$

We can use restriction to talk about strategies for individual agents or groups of agents, which place no restrictions on the behavior of the rest of the system. Note that $\top$ has the same semantics as $S|_\emptyset$ for any S.

In order to determine satisfaction of the rely-guarantee operator, we also need a mechanism for combining multiple strategies.

Locale Definition 4 *We say that two strategies S and T are* consistent *if for all input we have $[\![S]\!](a, \rho) \cap [\![T]\!](a, \rho) \neq \emptyset$. We define the* join *of two consistent strategies, written $S \sqcap T$, by $[\![S \sqcap T]\!](a, \rho) = [\![S]\!](a, \rho) \cap [\![T]\!](a, \rho)$.*

In other words, $S \sqcap T$ allows only actions that are allowed by both S and T. When S and T are inconsistent, the output of $S \sqcap T$ is ill-defined, since all strategies must allow at least one action for any input. We take care to ensure that this case does not arise in the evaluation of the satisfaction of RGTL formulae. We also assume that the $\sqcap$ operator is associative, commutative, idempotent, and has $\top$ as an identity; in other words, $\sqcap$ induces a semilattice on strategies, with $\top$ as the top element.

The $\sqcap$ operator can be shown to have the following properties with respect to refinement:

Lemma 2. *If $T \sqsubseteq S$, then $T \sqcap S' \sqsubseteq S \sqcap S'$ for any S' consistent with T.*

Lemma 3. *$(S \sqcap T)|_A \sqsubseteq S|_A$ and $(S \sqcap T)|_A \sqsubseteq T|_A$ for any consistent S and T.*

These properties are useful in establishing a framework for component-wise reasoning in RGTL (see Section **??**).

4.3 RGTL Semantics

The satisfaction of a RGTL state formula is defined with respect to a CGS C, a strategy S, and state information ρ, as follows:

- $C, S, \rho \models \mathsf{true}$
- $C, S, \rho \models p$ iff $p \in \pi(\mathsf{current\_state}(\rho))$ where $p \in \mathit{\Pi}$ is an atomic proposition

- $C, S, \rho \models \psi_1 \wedge \psi_2$ iff $C, S, \rho \models \psi_1$ and $C, S, \rho \models \psi_2$
- $C, S, \rho \models \neg\psi$ iff $C, S, \rho \not\models \psi$
- $C, S, \rho \models \psi_1 \overset{A}{\Rrightarrow} \psi_2$ iff $\forall T.\ T|_A \sqsubseteq S|_A \wedge (\forall R.\ C, T|_A \sqcap R|_{\overline{A}}, \rho \models \psi_1) \Rightarrow$ $C, S \sqcap T|_A, \rho \models \psi_2$
- $C, S, \rho \models \Lambda\varphi$ iff $\forall \lambda \in \mathsf{out}(C, S, \rho).\ C, S, \rho, \lambda \models \varphi$

Of particular note is the semantics of the rely-guarantee arrow, which formally expresses the core of the rely-guarantee reasoning principle: $\psi_1 \overset{A}{\Rrightarrow} \psi_2$ holds when, for any strategy $T|_A$ for A that guarantees ψ_1 regardless of the behavior of the rest of the agents ($R|_{\overline{A}}$), that strategy combined with the current strategy S guarantees ψ_2. In other words, as long as $T|_A$ can be relied on to provide ψ_1, S and $T|_A$ together guarantee ψ_2.

The satisfaction of a path formula also depends on a future path λ, which in general is provided by the strategy S through evaluation of the Λ operator.

- $C, S, \rho, \lambda \models \varphi_1 \wedge \varphi_2$ iff $C, S, \rho, \lambda \models \varphi_1$ and $C, S, \rho, \lambda \models \varphi_2$
- $C, S, \rho, \lambda \models \neg\varphi$ iff $C, S, \rho, \lambda \not\models \varphi$
- $C, S, \rho, \lambda \models \varphi$ iff $C, S, \rho \cdot \lambda_1, \lambda_{[1,\infty)} \models \varphi$
- $C, S, \rho, \lambda \models \varphi_1\, \mathcal{U}\, \varphi_2$ iff $\exists i.\ C, S, \rho \cdot \lambda_{[1,i]}, \lambda_{[i,\infty)} \models \varphi_2 \wedge$ $\forall j < i.\ C, S, \rho \cdot \lambda_{[1,j]}, \lambda_{[j,\infty)} \models \varphi_1$
- $C, S, \rho, \lambda \models \psi$ iff $C, S, \rho \models \psi$

This satisfaction relation is defined as a primitive recursive function in the `RGTL_semantics` locale, and forms the basis for all of the following theorems and proofs.

5 Example: Verifying Rely-Guarantee Properties

Recall the simple light-switch system of Section ??. In this section, we will use RGTL to formally state and verify the rely-guarantee property described previously.

We begin by describing the concurrent game structure C that we will use to model the system. We have two agents, A and B, each in charge of a toggle. We will model the state of our system with a collection of boolean variables: $\mathsf{light\_on}$, which is true when the light is on in the current state; pushed_A, which is true when A's toggle was pushed in the previous state; and pushed_B, which is true when B's toggle was pushed in the previous state. Thus, our system has a total of eight states. Our variables will also act as our atomic propositions: each holds on a state exactly if it is true in the state. The actions available to each agent are either to push their toggle, or to do nothing (a τ action). In every state, both actions are enabled for each agent. Then our transition function can be described as:

$$\delta(q, (\sigma_A, \sigma_B)) = \left\{ \begin{array}{l} \mathsf{light\_on} \ = \text{if } \sigma_A = \sigma_B \text{ then } \mathsf{light\_on}\ q \text{ else } \neg\mathsf{light\_on}\ q \\ \mathsf{pushed}_A = (\sigma_A = \mathsf{push}) \\ \mathsf{pushed}_B = (\sigma_B = \mathsf{push}) \end{array} \right\}$$

For this example, we will take our state information to be histories, that is, finite sequences of states already seen. We can take the set of strategies to be all functions mapping elements of $\{A, B\}$ and sequences of states to non-empty subsets of $\{\tau, \mathsf{push}\}$. The strategy $\top$ is the function that assigns to each agent the full set $\{\tau, \mathsf{push}\}$ in each state. The join of two strategies is the component-wise intersection of the outputs of the strategies. We will show in Section **??** that the axioms of the `RGTL_semantics` locale are satisfied by this model.

Finally, recall the system property we wish to verify: that eventually the light will always be on. As in LTL, this property may be expressed in RGTL as $A\Diamond\Box\mathsf{light\_on}$ (where $\Diamond$ ("eventually") and $\Box$ ("always") can be defined in terms of the $\mathcal{U}$ operator). As stated earlier, this is not a property that one agent alone can guarantee. However, if we assume that agent B will eventually stop pushing their toggle ($A\Diamond\Box\neg\mathsf{pushed}_B$), then there is a strategy for A to pursue, namely:

$$S(a, \{\mathsf{light\_on},\ \mathsf{pushed}_A,\ \mathsf{pushed}_B\} = \left\{ \begin{array}{ll} \{\tau\} & \text{if } a = A\ \wedge\ \mathsf{light\_on} \\ \{\mathsf{push}\} & \text{if } a = A\ \wedge\ \neg\mathsf{light\_on} \\ \{\tau, \mathsf{push}\} & \text{if } a = B \end{array} \right\}$$

S is a strategy for A in the sense that $S|_A = S$, that is, S only restricts the behavior of A.

Using these pieces, we can prove the following for any history ρ:

Lemma 4. $C, S, \rho \models (A\Diamond\Box\neg\mathsf{pushed}_B) \overset{B}{\Rrightarrow} (A\Diamond\Box\mathsf{light\_on})$.

Proof. By the semantics of RGTL, we can prove this by fixing a strategy T such that $T|_B \sqsubseteq S_B$, assuming that $\forall R.\ C, T|_B \sqcap R|_A, \rho \models A\Diamond\Box\neg\mathsf{pushed}_B$, and showing that $C, S \sqcap T|_B, \rho \models A\Diamond\Box\mathsf{light\_on}$. In particular, since $S|_A = S$, we may assume that $C, T|_B \sqcap S, \rho \models A\Diamond\Box\neg\mathsf{pushed}_B$. By the semantics of A, this means that along every outcome $\lambda \in \mathsf{out}(T|_B \sqcap S, \rho)$, there is some point i such that for any $j \geq i$, $\neg\mathsf{pushed}_B\ \lambda_j$. Given our labeling, this is true only if $\sigma_B = \tau$ from point i onwards, that is, if B only performs τ after point i. Now, either $\mathsf{light\_on}\ \lambda_i$, or $\neg\mathsf{light\_on}\ \lambda_i$. In the former case, the action prescribed by S for A is τ, and since B must also perform τ, the light will remain on indefinitely. In the latter case, the action prescribed by S for A is push, and since B must perform τ, the light will go on in the next state. In either case, by the above logic, once the light is on it will remain on indefinitely, as both agents continue to perform τ. Thus $C, S \sqcap T|_B, \rho \models A\Diamond\Box\mathsf{light\_on}$, and the proof is complete. □

In this manner, we can use RGTL to state and verify properties on a single agent given some assumptions on the remainder of the system. In Section **??**, we will state a theorem that allows us to compose properties of this form to construct general specifications for a larger system.

6 Logical Properties of RGTL

Here we present various theorems that facilitate reasoning about specifications in RGTL, all of which have been formally proved in Isabelle. All theorems are proved in the context of the `RGTL_semantics` locale, and so can be generalized to any interpretation of concurrent game structures, strategies, and strategy operators.

6.1 Properties of the $\Rrightarrow$ Operator

First, we examine the behavior of the rely-guarantee operator $\overset{A}{\Rrightarrow}$ at the extremes, that is, when A is either the full set of agents $\mathcal{A}$ or the empty set.

Lemma 5. *$C, S, \rho \models \psi_1 \overset{\mathcal{A}}{\Rrightarrow} \psi_2$ iff $C, T, \rho \models \psi_1$ implies $C, T, \rho \models \psi_2$ for all $T \sqsubseteq S$.*

In other words, $\overset{\mathcal{A}}{\Rrightarrow}$ is a stronger form of implication that holds not only for the current strategy S but for all sub-strategies of S as well.

Lemma 6. *$C, S, \rho \models \psi_1 \overset{\emptyset}{\Rrightarrow} \psi_2$ iff $C, S, \rho \models \psi_2$ only if $C, T, \rho \models \psi_1$ for all T.*

This lemma shows that $\psi_1 \overset{\emptyset}{\Rrightarrow} \psi_2$ states the rather unintuitive property that ψ_2 holds under the current strategy only if ψ_1 is true under *any* strategy, i.e., ψ_1 is a constant that holds regardless of strategy. While at first this property may seem too restrictive to be of use, we can in fact use it to construct several defined operators that provide general quantification over strategies.

Definition 4. *Let $\mathsf{exS}\ \psi \equiv (\neg\psi) \overset{\emptyset}{\Rrightarrow} \mathsf{false}$ and $\mathsf{allS}\ \psi \equiv \neg\mathsf{exS}\ \neg\psi$.*

Lemma 7. *$C, S, \rho \models \mathsf{exS}\ \psi$ iff $\exists S'.\ C, S', \rho \models \psi$.*

Lemma 8. *$C, S, \rho \models \mathsf{allS}\ \psi$ iff $\forall S'.\ C, S', \rho \models \psi$.*

These operations help us bridge the gap between RGTL, in which established strategies are carried throughout a formula, and ATL, in which strategies are reselected at each strategy quantifier.

6.2 Reasoning in RGTL

The core of component-wise reasoning in RGTL is the following theorem, modeled on the rule given by Xu et al. for parallel composition in concurrent programs [?].

Theorem 1. *Suppose we have a CGS C, a strategy S, and disjoint sets of agents A and B such that $C, S|_A, \rho \models \mathit{rely}_A \overset{\overline{A}}{\Rrightarrow} \mathit{guar}_A$ and $C, S|_B, \rho \models \mathit{rely}_B \overset{\overline{B}}{\Rrightarrow} \mathit{guar}_B$. Furthermore, suppose that for all T, $C, T, \rho \models (\mathit{rely}_A \wedge \mathit{guar}_A) \Rightarrow \mathit{rely}_B$ and $C, T, \rho \models (\mathit{rely}_B \wedge \mathit{guar}_B) \Rightarrow \mathit{rely}_A$. Then $C, S|_{A \cup B}, \rho \models \mathit{rely}_A \overset{\overline{A \cup B}}{\Rrightarrow} \mathit{guar}_A \wedge \mathit{guar}_B$.*

Proof. We show that $C, S|_{A\cup B}, \rho \models \mathit{rely}_A \overset{\overline{A\cup B}}{\Rrightarrow} \mathit{guar}_A \wedge \mathit{guar}_B$ by fixing a strategy $T|_{\overline{A\cup B}}$ for agents not in $A \cup B$, assuming that $T|_{\overline{A\cup B}}$ guarantees rely_A for any behavior of A and B, and showing that therefore $U = S|_{A\cup B} \sqcap T|_{\overline{A\cup B}}$ satisfies $\mathit{guar}_A \wedge \mathit{guar}_B$. In particular, we may assume that $T|_{\overline{A\cup B}} \sqcap S|_B$ guarantees rely_A. Then since $C, S|_A, \rho \models \mathit{rely}_A \overset{\overline{A}}{\Rrightarrow} \mathit{guar}_A$, we know that $S|_A \sqcap T|_{\overline{A\cup B}} \sqcap S|_B = U$ guarantees guar_A.

Similarly, we may assume that $T|_{\overline{A\cup B}}$ guarantees rely_A, and thus show that $S|_A \sqcap T|_{\overline{A\cup B}}$ guarantees guar_A. Using our assumption once more, we have that $S|_A \sqcap T|_{\overline{A\cup B}}$ also satisfies rely_A, and so satisfies rely_B. Then, since $C, S|_B, \rho \models \mathit{rely}_B \overset{\overline{B}}{\Rrightarrow} \mathit{guar}_B$, we can conclude that $S|_B \sqcap S|_A \sqcap T|_{\overline{A\cup B}} = U$ satisfies guar_B as well, and the proof is complete. □

This theorem connects RGTL to the method of rely-guarantee reasoning for which it is named [**?**]. The pre- and post-conditions used by Xu et al. are absent, since RGTL deals with properties on infinite executions rather than terminating processes, but otherwise the rely-guarantee method of reasoning fits neatly with the $\overset{A}{\Rrightarrow}$ operator, justifying our intuitive understanding of it as the "rely-guarantee arrow". While the language of Xu et al. uses an interleaved model of concurrency, the CGS model provides true synchronization, so the disjunctive requirements on the rely- and guarantee-formulae can be replaced with stronger conjunctive conditions. Using this rule, if we prove that each component of a system satisfies its specification (its guarantee) given the protection envelope of the rest of the system, we can then conclude that the combined system satisfies the combination of each component specification.

6.3 Expressiveness

With the help of the exS operator defined in Section **??**, we can construct an embedding of ATL* in RGTL. In particular, we can define a syntactic transformation h from a formula in ATL* to an RGTL formula as follows:

- $h_{state}(p) = p$ where $p \in \Pi$ is an atomic proposition
- $h_{state}(\neg\psi)_{state} = \neg h_{state}(\psi)$, and $h_{state}(\psi_1 \wedge \psi_2) = h_{state}(\psi_1) \wedge h_{state}(\psi_2)$
- $h_{state}(\langle\langle A\rangle\rangle\varphi) = \mathsf{exS}\ \neg(A(h_{path}(\varphi)) \overset{A}{\Rrightarrow} \mathsf{false})$
- $h_{state}(\neg\varphi)_{path} = \neg h_{path}(\varphi)$, and $h_{path}(\varphi_1 \wedge \varphi_2) = h_{path}(\varphi_1) \wedge h_{path}(\varphi_2)$
- $h_{path}(\psi) = h_{state}(\psi)$ where ψ is a state formula
- $h_{path}(\varphi) = h_{path}(\varphi)$
- $h_{path}(\varphi_1\, \mathcal{U}\, \varphi_2) = h_{path}(\varphi_1)\, \mathcal{U}\, h_{path}(\varphi_2)$

In order to show that this translation preserves the semantics of ATL*, we first must address two major disparities between RGTL and ATL. The first is revocability of strategies: while in ATL all strategies are cleared from the context at each quantification operator, RGTL may in general retain its strategies indefinitely once chosen. The use of the exS operator allows us to simulate the revocable behavior of ATL:

Lemma 9. *For any strategy S, $C, S, \rho \models h_{state}(\psi)$ iff $C, \top, \rho \models h_{state}(\psi)$, and $C, S, \rho, \lambda \models h_{path}(\varphi)$ iff $C, \top, \rho, \lambda \models h_{path}(\varphi)$.*

The second point of disparity is in the treatment of state information. In ATL, the information available to a strategy begins at the point the strategy is chosen; while it may build up knowledge of the past over its lifetime, it has no access to the states visited before reaching the strategy quantifier where it was invoked. In RGTL, by contrast, a strategy may have available the entire history built up over the course of evaluation of a formula. This gap is bridged by use of Locale Axiom **??** from Section **??**; given that there exists a strategy that produces certain outcomes given some state information, we can provide one that produces the same outcomes given only the current state, and vice versa. (Note that, since the type of state information is a parameter to the `CGS_strategies` locale, the state information used may not necessarily be a history; in the case where the type of state information is instantiated to be simply the current state, the following lemma is trivial.)

Lemma 10. *For any strategy S, $C, S, \rho \models h_{state}(\psi)$ iff the single-state state information $C, S, \mathsf{init}(\mathsf{current\_state}(\rho)) \models h_{state}(\psi)$, and $C, S, \rho, \lambda \models h_{path}(\varphi)$ iff $C, S, \mathsf{init}(\mathsf{current\_state}(\rho)), \lambda \models h_{path}(\varphi)$.*

With these two differences reconciled, we can then prove the following theorem.

Theorem 2. *For any ATL* state formula ψ, path formula φ, and state information ρ, $C, \rho \models_{\mathrm{ATL}} \psi$ if and only if $C, \top, \rho \models_{\mathrm{RGTL}} h_{state}(\psi)$, and $C, \lambda \models_{\mathrm{ATL}} \varphi$ if and only if $C, \top, \mathsf{init}(\lambda_0), \lambda \models_{\mathrm{RGTL}} h_{path}(\varphi)$.*

Proof. By simultaneous induction on the structure of ψ and φ.

While most cases of the translation are straightforward, the translation of the strategy quantifier $\langle\langle A\rangle\rangle$ is of particular interest. To understand the correctness of the embedding, we must unfold the semantics of our translation for $\langle\langle A\rangle\rangle\psi$. By Lemma **??**, $\mathsf{exS}\ \neg(\Lambda(\varphi) \overset{A}{\Rrightarrow} \mathsf{false})$ is true given state information ρ iff $\exists S.\ C, S, \rho \models \neg(\Lambda(\varphi) \overset{A}{\Rrightarrow} \mathsf{false})$. In general, for any formula ψ, we have that $C, S, \rho \models \neg(\psi \overset{A}{\Rrightarrow} \mathsf{false})$ iff $\exists T.\ T|_A \sqsubseteq S|_A \wedge (\forall R.\ T|_A \sqcap R|_{\overline{A}}, \rho \models \psi)$. Choosing S to be equal to T, we then have that $C, S, \rho \models \mathsf{exS}\ \neg(\Lambda(\varphi) \overset{A}{\Rrightarrow} \mathsf{false})$ iff $\exists T.\ \forall R.\ C, T|_A \sqcap R|_{\overline{A}}, \rho \models \Lambda(\varphi)$, that is, there is some strategy T for A such that for all strategies R for the remaining agents, in all outcomes, φ holds. This is precisely the definition of the strategy quantification operator $\langle\langle A\rangle\rangle$. □

Thus, h is a semantics-preserving embedding of ATL* in RGTL, and we can conclude that RGTL is at least as expressive as ATL*. This proof is completed in the `RGTL_semantics` locale, and so is independent of implementation details, and in particular of the type of strategies used. In other words, we have shown that RGTL is at least as expressive as ATL* for all variants of strategies – whether memory-based, memoryless, deterministic, or nondeterministic – as long as RGTL and ATL* use the same type of strategies.

As Alur et al. have shown that model-checking for ATL* is 2EXPTIME-complete (for memory-based deterministic strategies), model-checking for RGTL with these strategies is at least 2EXPTIME-complete. Model-checking for memoryless strategies may be more tractable, since the space of memoryless strategies is sharply constrained by the size of the CGS; on the other hand, model-checking for fully nondeterministic strategies is likely to be more complex.

6.4 Concrete Interpretation

Thus far, all our reasoning has taken place inside the `RGTL_semantics` locale, under the assumption of a type of strategies supporting the various operations used in the semantics of RGTL. In order to show that these properties hold for any actual logical system, we must show that there exists an *interpretation* of the locale, i.e., a concrete instantiation of the various required types and sets that satisfies the locale's axioms. In this section, we present a model of nondeterministic strategies with unbounded memory, and use it to construct an interpretation of the `RGTL_semantics` locale.

Definition 5. *A* nondeterministic strategy with unbounded memory *on a CGS* $(\mathcal{A}, Q, \Pi, \pi, \Sigma, e, \delta)$ *is a function* $S : \mathcal{A} \times Q^+ \to 2^\Sigma$ *such that for any agent* a *and history* ρ, $S(a, \rho) \subseteq e(a, \mathsf{last}(\rho))$ *and* $S(a, \rho)$ *is non-empty.*

Using this definition, we can construct an interpretation of `RGTL_semantics` in two steps. More precisely, we will construct a proof that the `CGS` locale, extended with this notion of strategies, is a sublocale of `RGTL_semantics`; that is, we will provide concrete interpretations for strategies and strategy operators, but continue to axiomatize the definition of a concurrent game structure. Since `RGTL_semantics` is built on top of the `CGS_strategies` locale, we begin by showing that `CGS` extended with this notion of strategies is a sublocale of `CGS_strategies`.

Lemma 11. *A `CGS` along with its nondeterministic strategies with unbounded memory is an instance of `CGS_strategies`.*

Proof. To prove this, we must show that the type of state information supports the operations $\mathsf{current\_state}(\rho)$, $\mathsf{init}(\rho)$, and $\rho \cdot q$, and that the type of strategies supports the operation $[\![S]\!]$. Our type of state information is Q^+, the set of finite non-empty sequences of states in Q, and so we can define $\mathsf{current\_state}(\rho) = \mathsf{last}(\rho)$, $\mathsf{init}(q) = q$, and $\rho \cdot q = \rho \cdot q$ in the sense of concatenation of sequences. Similarly, our strategies are already functions from $\mathcal{A} \times Q^+$ to 2^Σ that are non-empty and consistent with e, so we may define $[\![\_]\!]$ to be simply the identity function.

In addition, we must show that the two strategy-creation axioms are satisfied by our interpretation. Given a strategy S with certain behavior on a sequence ρ, the strategy $\lambda a\ \rho'.\ S(a, \rho_{[0,|\rho|-1)} \cdot \rho')$ produces the same behavior on $\mathsf{current\_state}(\rho)$; similarly, for a given sequence ρ, if S has some behavior on $\mathsf{current\_state}(\rho)$, the strategy $\lambda a\ \rho'.\ S(a, \rho'_{[|\rho|-1,|\rho'|)})$ has the same behavior on

ρ. Finally, we must show that for any outcome $\lambda \in \mathsf{out}(S, \rho)$, there exists a sub-strategy $T \sqsubseteq S$ such that $\mathsf{out}(T, \rho) = \{\lambda\}$. This is the most complex part of the proof of interpretation; putting aside the technical details, the intuition is to provide the strategy that, for each history of the form $\rho \cdot \lambda_{[1,i]}$, produces a vector of actions $\overline{\sigma}$ such that $\delta(\lambda_i, \overline{\sigma}) = \lambda_{i+1}$. We know that such a vector exists at each point i because S has produced λ as an outcome through precisely such a vector, and so can create a strategy that at each step mirrors the behavior of S in producing λ. □

Since our strategy interpretation function is the identity function, it follows naturally that the strategy operators are given by their axiomatized semantics.

Definition 6. *Let* $\top = \lambda a\ \rho.\ e(a, \mathsf{current\_state}(\rho))$*,* $S|_A = \lambda a\ \rho.$ if $a \in A$ then $S(a, \rho)$ else $e(a, \mathsf{current\_state}(\rho))$*, and* $S \sqcap T = \lambda a\ \rho.\ S(a, \rho) \cap T(a, \rho)$*.*

Lemma 12. *Under these definitions, a `CGS` with nondeterministic strategies with unbounded memory is an instance of `RGTL_semantics`.*

Proof. We must simply show that each strategy operator satisfies its axioms, which follows directly from the definitions of the operators. □

7 Conclusion

In this paper, we have presented the rely-guarantee-based temporal logic RGTL, and demonstrated its use as a compositional method for verifying properties of multi-agent concurrent systems. We have shown a semantics-preserving embedding of ATL* into RGTL parameterized by the type of strategies used by the agents, so that we can be assured of the relative expressiveness of RGTL as long as certain assumptions hold on the type of strategies. We also have presented an instantiation of the generic type of strategies, and demonstrated that one common notion of strategy satisfies the necessary assumptions. All theorems have been formally verified in the Isabelle theorem prover, giving us a strong assurance of their correctness.

While we believe RGTL has appeal as a logic in its own right, it has also benefited considerably from the use of Isabelle in its development. By building RGTL on top of Isabelle's locale system, we are able to define several variants of the logic – deterministic, nondeterministic, memoryless, memory-based – with a single semantic function. The use of locales on the one hand allows us to state our theorems and write our proofs in their full generality, and on the other hand forces us to explicitly state our assumptions and demonstrate that they are satisfied by the intended models. The building blocks of our proofs, including the locales `CGS` and `CGS_strategies`, may be reused in the Isabelle development of other strategy-based logics, allowing for strategy-agnostic expressiveness results such as ours with respect to ATL*. The strategy operators defined in our locales may also have uses beyond the semantics of RGTL; for instance, our join operation $\sqcap$ on strategies is related to the † operation used in IATL to update a CGS with a strategy [?]. Recent research in agent-based formalisms has given rise to a plethora of ATL-related logics (see for instance Brihaye et al.'s taxonomy of ATL variants [?]); we believe that movement towards a general, strategy-agnostic framework for defining the semantics of these logics will considerably simplify the process of formally stating, verifying, and comparing them.

The Isabelle development described in this paper can be found online at `https://netfiles.uiuc.edu/mansky1/www`.

8 Future Work

While RGTL represents a step forward in expressing properties of multi-agent systems, there are still various features of real-world programs that are not reflected in the logic. For instance, using the ordinary temporal logic connectives of LTL/CTL/ATL, it is difficult to compare values across states in a path; a property such as "the value of x always increases" is non-intuitive to state and prove. The Temporal Logic of Actions (TLA) [?] is a variant of LTL that addresses this problem by expanding the atomic propositions to relations over pairs of states ("actions"). Work is in progress to extend RGTL to the setting of actions, allowing a more concise intuitive description of software- and workflow-like properties.

One clear area for further work is the problem of *incomplete information*; in practice, not every player in a game may have full access to the current state. There has been extensive work on this problem as it relates to ATL and similar logics; see for instance the work of Dima et al. [?] Through approaches such as imposing equivalence classes on histories, thus limiting each player's knowledge of the environment, we can more accurately model partial-knowledge scenarios, for instance parallel programs in which each thread can only access certain variables.

Strategy Logic [?] is a logic related to ATL, with facilities for more general quantification over strategies. We are currently exploring an extension of strategy logic with strategy satisfaction and refinement, which we believe to be strictly more expressive than RGTL. Since strategy logic is known to be decidable, this may provide a method of proving the decidability of model-checking for RGTL.

While the complexity of model-checking full RGTL is as yet undetermined, in practice, the full expressiveness of RGTL may not be required. For instance, note that in the case study, arbitrary nesting of the $\overset{A}{\Rrightarrow}$ operator is not required to express the protection-envelope properties. If we restrict our language to a Horn-clause-like fragment of RGTL, in which the rely-guarantee operator is used in a strictly "positive" manner, the model-checking problem may become more tractable.

9 Related Work

In previous work by Nieto [?], the original Owicki-Gries method (as refined by Jones [?]) was formalized in Isabelle/HOL. This approach relies on axiomatic semantics (Hoare-style reasoning) rather than temporal logic for program verification, but provides a similar principle of modular reasoning.

Our notion of strategy satisfaction is similar to and borrows concepts from the STIT-extension of ATL proposed by Broersen et al. [?]; however, the STIT extension is restricted to the case of deterministic strategies, and does not address the subtleties of strategy combination and refinement. We also build on the work of Yasmeen on varieties of ATL and logics with strategies [?].

Mogavero et al. [?] define a semantics for ATL* that, like RGTL, retains knowledge of the execution complete history ρ rather than only the current state and future execution. In Mogavero's "relentful" approach, the temporal operators are evaluated on the combination of history and future execution, so that for instance $\diamond\varphi$ is satisfied if φ held sometime in the past. In our semantics, players may take history into account when making their strategic decisions, but each temporal formula is still evaluated beginning at the moment its strategy comes into effect, with the history serving only to increase the range of possible strategies.

GL (game logic) is a generalization of ATL/ATL* proposed by Alur et al. [?], which allows arbitrary quantification over sets of strategies. In combination with strategy satisfaction and refinement, this provides a mechanism similar to (but not equivalent to) the rely-guarantee operator of RGTL. Strategy logic, mentioned above, can also be seen as an extension of GL with more flexible strategy quantification.

ATL with Strategy Contexts and Bounded Memory [?] is another step towards greater control over the strategy quantification of ATL, providing several quantification operators that allow switching between revocable (ATL-style) and irrevocable (IATL-style) use of strategies. We have not yet determined the expressiveness of RGTL with respect to ATL$^*_{sc}$, which would be an interesting area for future work.

10 Acknowledgements

This material is based upon work supported in part by NASA Contract NNA10DE79C and NSF Grant 0917218. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NASA or NSF.