Little Tricky Logic: Misconceptions in the Understanding of LTL

Ben Greenman[a], Sam Saarinen[a], Tim Nelson[a], and Shriram Krishnamurthi[a]
a Brown University, Providence, RI, USA

Abstract

Context Linear Temporal Logic (LTL) has been used widely in verification. Its importance and popularity have only grown with the revival of temporal logic synthesis, and with new uses of LTL in robotics and planning activities. All these uses demand that the user have a clear understanding of what an LTL specification means.

Inquiry Despite the growing use of LTL, no studies have investigated the misconceptions users actually have in understanding LTL formulas. This paper addresses the gap with a first study of LTL misconceptions.

Approach We study researchers' and learners' understanding of LTL in four rounds (three written surveys, one talk-aloud) spread across a two-year timeframe. Concretely, we decompose "understanding LTL" into three questions. A person reading a spec needs to understand what it is saying, so we study the mapping from LTL to English. A person writing a spec needs to go in the other direction, so we study English to LTL. However, misconceptions could arise from two sources: a misunderstanding of LTL's syntax or of its underlying semantics. Therefore, we also study the relationship between formulas and specific traces.

Knowledge We find several misconceptions that have consequences for learners, tool builders, and designers of new property languages. These findings are already resulting in changes to the Alloy modeling language. Our study instruments are useful for training learners (whether academic or industrial) who are getting acquainted with LTL, and we provide a code book to assist in the analysis of responses to similar-style questions.

Grounding Our findings are grounded in the responses to our survey rounds. Round 1 used Quizius to identify misconceptions among learners in a way that reduces the threat of expert blind spots. Rounds 2 and 3 confirm that both other learners and researchers (who work in formal methods, robotics, and related fields) make similar errors. Round 4 adds deep support for our misconceptions via talk-aloud surveys.

Importance This work provides useful answers to two critical but unexplored questions: in what ways is LTL tricky and what can be done about it? Our survey instruments can serve as a starting point for futher studies.

ACM CCS 2012

- **Human-centered computing** → **User studies**;
- **Software and its engineering** → *Formal methods*;

Keywords LTL, misconceptions, user studies, property language design

The Art, Science, and Engineering of Programming

Perspective The Empirical Science of Programming

Area of Submission General-purpose programming

1 Introduction

Linear temporal logic (LTL) has long been a standard for writing property specifications in computer-aided verification. The language can express a variety of real-world phenomena while supporting good decision procedures [73]. It is also a small language, thereby presumably making it easy to learn and understand.

In recent years, LTL has been increasingly used for much more than verification. The old dream of temporal-logic synthesis [48, 61] has seen a revival [2, 3, 12]. LTL has been adapted to enable property-based testing of interactive web applications [56]. Even more intriguingly, roboticists have found intimate connections between LTL and robot planning [4, 27, 45], as a result of which numerous robotics systems now use LTL, e.g., [5, 11, 33, 40, 43, 68, 79]. Indeed, it is sufficiently pervasive that there are now even robotics classes that teach LTL to learners who have no prior experience with formal methods [49].

All these efforts are predicated on a central belief: that users of the logic actually understand it. The quality of verification, synthesis, or planning is only as good as the property statement. If a user *misunderstands* what the property is saying, there are no safeguards: the tool will blindly apply this property and check or generate the requested behavior, whether or not it was the desired one. It is therefore critical to know whether users accurately understand LTL, which is the focus of this paper.

As a case in point, this project was born of a worrisome incident. Two authors attended a research colloquium about using LTL in robot planning. The speaker, a roboticist, began with a brief introduction to LTL. However, this tutorial contained a mistake, but neither the speaker nor the somewhat LTL-aware audience spotted it. This prompted the authors to wonder whether this phenomenon is wider-spread, even amongst people trained in formal methods.

Concretely, this paper focuses on three directions of LTL understanding:

LTL to English: Given an LTL formula, can a reader accurately translate it into English? This is similar to what a person does when reading a specification, e.g., when code-reviewing work or studying a paper.

English to LTL: Given an English statement, can a reader accurately express it in LTL? This skill is essential for specification and verification.

Trace satisfaction: Given an LTL formula and a trace (sequence of states), can a reader accurately label the trace as satisfying or violating? Such questions directly test knowledge of LTL semantics.

We know of virtually no work that examines human factors in this setting (section 12).

Outline This paper begins by explaining the long-term goals of our work and how researchers in related areas have approached similar goals (section 2). It then presents the design of our multi-year study (section 3), the formative component of the study (section 4), and our method for confirming formative findings (section 5) before shifting focus to the main results and their implications. The paper concludes with related work (section 12) and a discussion of next steps (section 13).

Contributions Our work makes three main contributions:

- We find errors in all three question categories (sections 6, 7, and 8).
- We provide a code book of misconceptions (section 5).
- We provide three instruments to test for misconceptions (appendix A).

These contributions have implications for four classes of LTL users: learners, educators, tool builders, and designers of new property languages. The code book and instruments are of immediate value to learners and educators, whether in academic settings or industrial ones (e.g., [76]). Knowledge about misconceptions can be used by tool builders to create new learning tools or to issue alerts in existing ones. Lastly, our work is of use to designers of logics and is resulting in a change to Alloy 6.

In short: it may be folk knowledge that LTL is tricky, but *is it really*, *in what ways*, and *what can we do about it*? We believe this paper offers useful initial answers.

2 Background on Misconceptions

In educational psychology and other fields, there is an important difference between a *mistake* and a *misconception*. A mistake is simply an error; it could have occurred for any number of reasons. A misconception, in contrast, refers to having the wrong idea about a topic.

Misconceptions usually reveal themselves through mistakes, but not every mistake is a misconception. For instance, if subjects provide the wrong answer in response to a question, there are many possible explanations: they may have a genuine misconception; they may have misunderstood the question; they may have been tired; their hand may have slipped while checking boxes; and so on. In general, we can only discern a misconception by connecting to an intent.

In the education literature, seminal work by Hestenes [35, 36] introduced the idea of a *concept inventory* (CI). A concept inventory is a multiple-choice questionnaire where each question presents one correct answer and several wrong answers. The wrong answers are not chosen arbitrarily, however, but rather are carefully designed to (hopefully) be bijective with specific misconceptions. Thus, when a subject picks a particular wrong answer, there is a very high likelihood they have the corresponding misconception. For instance, if a study of children finds that they often misinterpret $\times$ to mean addition, a wrong answer for 4×3 would be 7.

Because a CI maps mistakes to misconceptions without requiring detailed responses, it is lightweight to deploy and effective for recognizing misconceptions with minimal effort. This paper takes concrete steps towards eventually creating a CI for LTL.

Creating a Concept Inventory Unfortunately, creating a CI is extremely labor and cost intensive. It often requires several rounds of interviews with subjects using Delphi processes and other resource-heavy methods [1, 22, 24, 32, 34, 70]. In response, Saarinen et al. [66] created a system called Quizius in which the subjects themselves generate questions and answer the questions that other subjects generated (i.e., a type of crowdsourcing).

Table 1 Survey rounds, questions, and paper outline

	Round 1	Rounds 2, 3, and 4
Trace Sat.	N/A	section 6
LTL ▷ Eng	section 4	section 7
Eng ▷ LTL	section 4	section 8

A central question for the Quizius system is how to decide which question to show at any given moment. Questions that are already generating disagreement may produce more of it, exposing more misconceptions; but new questions (on which there is not yet enough data) may produce interesting responses as well. This generates a choice between exploration of new questions and exploitation of existing ones. The choice corresponds to a multi-armed bandit process [7], which is what Quizius uses to decide which questions to present.

The Quizius paper showed (in a different domain: introductory Java programming) that Quizius produces results comparable to intensive expert work at vastly less cost. An additional feature of Quizius is that it helps reduce expert blind spots [52, 53]. However, we do not use Quizius alone; rather, we combine it with judicious use of expert effort to seed the system and to study responses and assemble the instruments described in this paper (i.e., "experts on the outside, Quizius on the inside").

3 Research Study Design

Our work spans two dimensions: three kinds of questions and four survey rounds involving different populations. These are the rows and columns of table 1. There are thus several logical ways to report our findings in a linear order. One possibility is to proceed column-by-column, following the surveys in chronological order. Another is to proceed row-by-row, focusing on the questions.

We feel a hybrid presentation is best. First, we describe Round 1 in full (section 4) because it was formative for the later rounds. Thereafter we focus on the question types (which are semantically coherent), presenting data from Rounds 2, 3, and 4 combined (sections 6 to 8), rather than focusing on chronology. Between these two major parts, section 5 provides context for the latter rounds.

The rest of this section explains the questions, the survey rounds, and their context.

3.1 Study Questions

Our studies focus on two translation questions. These questions relate to practical uses of LTL (table 2), namely, the reading and writing of specifications:

1. LTL to English (abbreviated: LTL ▷ Eng) questions present a short formula and ask for a paraphrase. For example, one question and an acceptable answer follow:

   Q. Translate to English: !F(!x1)

   A. "x1 is always true"

2. English to LTL (abbreviated: Eng ▷ LTL) questions present an English specification and ask for an LTL formula. For example:

   Q. Translate to LTL: "x1 holds after one or more steps"

   A. X(F(x1))

As we explain in section 5, we realized after Round 1 (hence the "N/A" in table 1) the need for a third question type that directly asks about LTL semantics. These questions do not correspond to a task that an LTL user explicitly performs, though it is implicit in translating to LTL:

3. Trace satisfaction (abbreviated: Trace Sat.) questions present two items: an LTL formula and a trace (a sequence of states) representing some concrete domain. They ask whether the trace satisfies the formula. All of our traces consist of a sequence of four states followed by a fifth "lasso" state that repeats forever. For example:

   Q. Is the formula X(x0 or x1) satisfied by this trace?

   {x0} {x0} {} {x0} {x0}

   A. Yes, because x0 holds in the second state.

3.2 Survey Rounds

We collected anonymized data via four survey rounds spread across a two-year span:

- **Round 1** took place in Spring 2020. It used 90 students enrolled in an upper-level, tool-based, applied logic course taught by an author at a private US university.
- **Round 2** took place in Spring 2021. It was the same course, institution, and instructor. The course had only 57 students. We attribute the low enrollment to COVID-19.
- **Round 3** took place in Summer 2021 and used 29 anonymous researchers who had prior exposure to LTL.
- **Round 4** took place in Spring 2022 after a third iteration of the applied logic course (same institution, same instructor) and recruited 11 students.

Round 1 used Quizius with expert seed questions to generate preliminary instruments for the LTL ▷ Eng and Eng ▷ LTL questions. Rounds 2 and 3 used Qualtrics and assessed the Round 1 instruments. Round 4 used Qualtrics and Zoom to record participants' talk-aloud reasoning while they took a survey. More details follow.

3.2.1 Round 1 Details

We administered Round 1 as a required assignment. Prior to it, students received two lectures that covered both LTL and the SPIN model checker [37]. One lecture was in-person; the second was remote. Students also modeled the dining philosophers problem in SPIN during one lab session. Unfortunately, students did not do any additional LTL assignments or exams because the first COVID-19 pandemic shutdowns occurred at this time.

Students completed two Quizius quizzes in Round 1, one for each of LTL ▷ Eng and Eng ▷ LTL. They responded to up to ten examples already in the system and had to submit one more question that they deemed interesting to the pool. The assignment

Q. Translate to LTL: Whenever x1 is true, x2 will be true at some point in the future.

A. LTL: ____
Rationale: ____

Q. Translate to English: G(x1 -> F(x2))

A. English: ____
Rationale: ____

Figure 1 Round 1 example questions

instructions promised full credit for all honest attempts, regardless of correctness. Students received credit through an anonymous email address.

To make sure the first few students would have questions to answer, the instructor seeded Quizius with LTL ▷ Eng and Eng ▷ LTL questions. Each direction began with only eight seeds to avoid biasing the question pool toward these expert-generated questions. The seeds were created as pairs; each Eng ▷ LTL seed is a valid translation of one LTL ▷ Eng seed and vice-versa. Figure 1 presents an example pair and shows that both question types asked for a translation and a plain-text rationale.

Students did not receive questions as pairs in the manner suggested by figure 1. If a student happened to receive a matching pair of questions, these questions would have appeared on separate web pages.

3.2.2 Round 2 Details

We administered Round 2 as a required assignment using Qualtrics. Prior to the survey, students received four remote lectures that covered LTL and the Electrum language [47]. (Electrum adds LTL to the Alloy modeling tool [21, 39].) Among other topics, the lectures modeled locking algorithms. Students additionally completed one Electrum lab and two Electrum assignments.

For Round 2, we curated the most interesting questions from Round 1 and designed a survey to take no more than 90 minutes of students' time, excluding breaks. (The median completion time was ultimately 93 minutes, including time when the browser tab was idle.) The survey had 19 questions, divided as follows:

- 5 LTL ▷ Eng questions,
- 5 Eng ▷ LTL questions, and
- 9 trace satisfaction questions.

After each part, the survey gave students the opportunity to submit one new question and then encouraged them to take a short break before moving on. As in Round 1, the assignment instructions promised full credit for honest attempts and delivered this credit via anonymous email addresses.

To keep student effort manageable, we made two simplifications. First, we did not ask for rationales (which many students had spent a great deal of time writing in Round 1). We instead asked for their confidence and, optionally, a *near miss*: an incorrect response that another learner might submit. Second, we concretized all questions to ask about a panel with three lights: red, green, and blue.

Q. Is the formula `Red` satisfied by this trace? (the final state loops)

{GB} {RGB} {RGB} {RGB} {RGB}

A. Answer: Yes / No

What about the trace made you give that answer?

Figure 2 Round 2 example trace satisfaction question

The Eng ▷ LTL section encouraged students to check the syntax (but not correctness) of their formulas and gave students the option of saying that a specification was inexpressible in LTL. The trace satisfaction questions presented a formula and a trace (figure 2); they asked for a yes/no judgment about whether the trace satisfied the formula and a plain-text explanation. The instructions for these questions explained that the final state in each trace was a "lasso" that repeated forever.

3.2.3 Round 3 Details

To see whether our results extended beyond the students, we posed similar questions to a more experienced population. We requested research colleagues who work in formal methods and robotics to share a survey amongst researchers in their group. To avoid subjects feeling self-conscious, we did not collect email addresses, IP addresses, or any other personal information (and said so up front). The subjects were not paid for their time. All answers were nevertheless vetted by the authors for "seriousness," and none were excluded on these grounds.

To maximize participation, because we were using the uncompensated time of experts, we wanted to minimize the effort we demanded of them. Therefore, each subject was asked only nine questions from the Round 2 question pool (Qualtrics sampled questions uniformly at random without replacement):

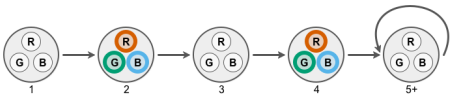
- 4 trace satisfaction questions,
- 3 LTL ▷ Eng questions, and
- 2 Eng ▷ LTL questions—in which we accepted any style of LTL syntax.

The number of questions in each category was inversely proportional to our sense of how much effort each one takes. The final page of this Qualtrics survey collected subjects' research areas and prior exposure to LTL. Most respondents worked (at least partly) in formal methods / verification (N=22), almost half worked in AI / machine learning (N=14), three worked in robotics, one worked in programming languages, and one worked in HCI. All respondents had some prior exposure to LTL.

The survey intentionally did not ask subjects to rate their LTL expertise. Doing so at the beginning could have reduced their confidence. Doing so at the end may merely reflect their impression of how they did. Also, respondents may not have shared a frame of reference to answer such a question uniformly.

The Round 3 questions and prompts were the same as in Round 2 with minor changes: we improved the formatting of the LTL ▷ Eng questions, clarified the wording in a few Eng ▷ LTL questions, significantly improved the formatting of the trace satisfaction

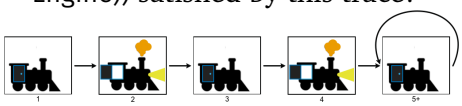
Q. Is the formula (eventually (always Red)) satisfied by this trace?



A. Answer: Yes / No

(a) Round 3 example

Q. Is the formula (eventually (always Engine)) satisfied by this trace?



A. Answer: Yes / No

(b) Round 4 example

Figure 3 Rounds 3 and 4 example trace satisfaction questions

questions, and removed the rationale for trace satisfaction—to reduce work and thus, hopefully, increase participation. Figure 3 presents an example Round 3 trace. The states are images rather than text and the transitions are clearly labeled with arrows. The red, green, and blue colors are from a colorblind-friendly palette [78].

3.2.4 Round 4 Details

Unlike the other rounds, we administered Round 4 as a talk-aloud study. Its main purpose was to distinguish mistakes from misconceptions (section 2) by having subjects explain their reasoning. Its secondary purpose was to test our prior findings in a different domain; specifically, we formulated questions in terms of a train with three features rather than as a panel with three colors (figure 3).

We recruited subjects who were enrolled in the Spring 2022 offering of the logic course mentioned above. The material covered in the course did not differ substantially from the previous year (section 3.2.2). For the study, subjects met with one author for a 25-minute Zoom session and received $50 compensation as either an electronic Amazon gift card or a physical gift card to the university bookstore. Subjects completed a survey with 19 questions: 9 trace satisfaction, 5 LTL ▷ Eng, and 5 Eng ▷ LTL. During the study, the author asked subjects to explain their reasoning out loud and provided minor clarifications throughout. For example, if a subject was unsure about the semantics of an LTL operator, the author asked the subject to describe possible semantics, choose one, and continue with that choice.

3.3 Additional Details

On the Use of Students As Kitchenham et al. [41] point out, the use of students as subjects is not a major threat for studies interested in novice software engineers; after all, "students are the next generation of software professionals." Many of our students had prior work experience and quite a few had job offers by the end of the semester (internship or full-time), making them suitable candidates for our study. All students had some classroom exposure to LTL. In Rounds 2 and 4, students had lab and homework experience modeling systems in Electrum.

English Language Fluency Because two of our tasks involve English, the fluency of the subjects matters. Rounds 1, 2, and 4 took place at a university that conducts all

Table 2 LTL and Electrum syntax

LTL	Electrum	English
G(a0)	always(a0)	a0 always holds
F(a0)	eventually(a0)	a0 holds at least once
X(a0)	after(a0)	
	next_state(a0)	a0 holds in the next state
a0 U a1	a0 until a1	a1 eventually holds, and a0 holds in all prior states

classes in English and expects a high degree of fluency. International applicants can demonstrate fluency through a TOEFL score of 100 or more (Internet-based) or 600 or more (paper-based), or an IELTS score of 8.0 or more. Most also had 1–3 years at the university by then. In general, we did not observe any notable issues in the English output. (However, we *did* make some changes to the prose of questions between rounds: see section 4.)

Round 3 participants were either leading researchers or members of their research groups; that is, professionals who read and write papers that use LTL. Furthermore, again, we did not notice any significant English problems in their written output.

Two Syntaxes: Classic and Electrum Although Rounds 1, 2, and 4 took place in instances of the same course, the courses employed different LTL tools and consequently used different syntax for formulas. Table 2 summarizes the syntax involved. Round 1 used the classic LTL syntax with single-letter names for temporal operators: G, F, X, and U. Round 2 switched to Electrum so that students could continue to use Alloy—which was introduced earlier in the course—to work with LTL formulas. Electrum uses English words for the LTL operators: always, eventually, after, and until.

During Round 2, we identified a vernacular misconception regarding Electrum's after operator (discussed in section 9.1.1). Round 4 and the next iteration of the course therefore used the name next_state for the X operator. Our data suggests that the change is an improvement. (Electrum cannot use the traditional name next [61] for X because next is a reserved word in Alloy.)

4 Formative Data and Analysis

In Round 1, using Quizius, subjects generated 90 LTL ▷ Eng questions and 87 Eng ▷ LTL questions, which received 901 and 886 answers, respectively. (Not all subjects finished all parts.) These answers were *not* equally spread across the questions because of the multi-armed bandit process in Quizius: questions received a median of six answers, but the highest-ranked questions received 30 or more answers each.

To analyze the data, one coder studied the high-ranking questions and categorized the incorrect answers. The use of a single coder is ameliorated because (a) the coder was working *in conjunction* with the Quizius algorithm; (b) the results were going to be tested in subsequent rounds; and, (c) for generating formal instruments, we intended

Table 3 Datasets and analysis methods

Question	Round(s)	Artifact	N	Method
Trace Sat.	2,3,4	Yes/No choices	728	auto
Trace Sat.	2	Eng text	513	manual
Trace Sat.	4	Eng speech	99	manual
LTL ▷ Eng	2,3,4	Eng text	427	manual
LTL ▷ Eng	4	Eng speech	55	manual
Eng ▷ LTL	2,4	Electrum code	340	auto, manual
Eng ▷ LTL	3	LTL text	58	manual
Eng ▷ LTL	4	Eng speech	55	manual

to and did use multiple coders with a proper inter-coder reliability measure (section 5). Thus, we felt this was a reasonable process for generating draft instruments.

The data from Round 1 led us to conjecture 15 preliminary LTL misconceptions, listed in the supplement. We intentionally do not present the list here because these preliminaries were mere "stepping stones" that helped us obtain a robust set of misconceptions. For example, one preliminary misconception that did not pan out was the idea that an X-wrapped term spans multiple states. Outside of Round 1, very few subjects made mistakes that supported this misconception. Another was about the meaning of the English word "therefore": does it include the current state or not? Rather than test this ambiguity, we chose to avoid it by rewording questions in Round 2. Further details on the preliminaries appear in chapter 7 of Saarinen's dissertation [65].

5 Confirming Formative Findings

Armed with preliminary misconceptions, our subsequent rounds set out to confirm that they persist. To do this, we extracted the questions that seemed most productive (in that they generated the most misconceptions), and curated them (e.g., cleaning up their presentation for those that were authored by learners). We used these questions for the subsequent rounds.

After reviewing the Round 1 data, we also realized that we were failing to check how well subjects understood the basic semantics of LTL. This is a subtle, but vital, shift in cognitive setting. *Evaluating* the truth of a formula on a trace and *synthesizing* a formula given an English prompt are related, but not equivalent, skills. This motivates adding trace satisfaction questions.

In fact, traces help in both directions. English answers to LTL ▷ Eng questions may be ambiguous or low-effort. Subjects might simply transliterate (`Red U Green`) into "`Red` is on until `Green` is on," which does not provide much insight. Adding a few trace satisfaction questions therefore lets us check for low-level issues we might have otherwise missed. One such example is the belief that (`Red U Green`) requires `Red` to become false when `Green` becomes true (which we label "`ExclusiveU`").

5.1 Dataset

Table 3 presents a high-level view of the data that we collected in later rounds. Trace satisfaction questions received yes/no answers in all rounds, and additionally received written explanations in Round 2 and spoken explanations in Round 4. LTL ▷ Eng questions received English translations in all rounds and spoken explanations in Round 4. Eng ▷ LTL questions received Electrum code in Rounds 2 and 4, free-form LTL text in Round 3, and spoken explanations in Round 4.

The last column in table 3 tells us what methods we can use for analyzing the data. Some of these are amenable to automated analysis: e.g., checking the yes/no answers and the correctness of the Electrum formulas (which we checked for *semantic*, not syntactic, equality to the correct answer). The written and spoken English answers require manual analysis. In Round 3, because we did not ask subjects to pre-check their LTL answers (to reduce the time burden), it was simpler to manually classify the responses.

The one remaining entry to explain is the manual analysis of Electrum answers in Round 2. Automated checking lets us decide if a formula is semantically correct. When it is wrong, however, we require human judgment to determine in what way it went wrong. In particular, semantic equality can create bins of equivalence classes, but answers in the same bin may correspond to different misconceptions. This kind of overlap appeared in our dataset; the first "caution" in section 8 presents an example.

5.2 A Code Book for Manual Evaluation

The manual aspects of our analysis need a code book to reliably map responses to misconceptions. Because we have three question types that require manual analysis (table 3), we developed three code books. Our coding began with the Eng ▷ LTL responses because we found these were the most difficult to comprehend. Using the misconceptions from Round 1 as a starting point, two authors applied the methods from grounded theory [31] to identify a core set of misconceptions and develop a code book. The coders worked through eight rounds of categorization, independently labeling formulas and then meeting to improve the code book. Each round included at least five and at most 15 formulas. The code book received 18 significant revisions overall, all of which are included in the supplement. After the eight development rounds, the two coders labeled 28 formulas each as a final test for agreement. The Cohen Kappa score [17] on this final test was 0.91, indicating extremely high agreement. In light of the high agreement, one coder labeled the remaining formulas and the second coder merely spot-checked the results.

Next, the two coders scanned the LTL ▷ Eng and trace satisfaction responses for errors that were not covered by the Eng ▷ LTL codes. Although one additional code arose from this scan (ExclusiveU), the coders found that most errors were of a similar nature to the Eng ▷ LTL errors. They therefore adapted the Eng ▷ LTL code book with minor changes to label the LTL ▷ Eng and trace satisfaction responses.

Figure 4 presents a generalized version of the code books. It consists of eleven tags, each of which comes with a description of the erroneous responses to which it applies

Abstract Code Book

The following tags describe semantic errors that a learner can make when responding to a survey question. If an incorrect response matches a tag description, we say that the tag *applies* to the response. For Eng▷LTL responses, consider the first three tags in order and then consider the remaining tags as a set. For LTL▷Eng and trace satisfaction responses, start with the third tag (Unlabeled); if it does not apply, then consider the rest as a set.

1. **Precedence** (Eng▷LTL only): Applies to LTL formulas that are correct up to missing parentheses. For other misparenthesized formulas, apply no labels (Unlabeled) instead of guessing about intent.
   - Expected (a0 and a1) => a2 but subject wrote a0 and a1 => a2
2. **ReasonableVariant** (Eng▷LTL only): Applies to LTL formulas that are correct for an unintended reading of the question. The example here is based on the phrase "Blue will turn on" (Rajhans et al. [63] call this a *rising edge* issue).
   - Expected F(Blue) but subject wrote F(!Blue and X(Blue))
3. **Unlabeled**: If an answer is convoluted and/or ambiguous, then apply this tag and no others.
4. **BadProp**: Applies to responses that mis-use a logical operator or an atomic symbol.
   - Expected "a0 implies a1" but subject wrote "a0 and a1"
   - Expected X(a0) but subject wrote X(a1)
5. **BadStateIndex**: Applies to responses that use a correct term at an incorrect state index. Does not apply when a fan-out operator (F, G, U) is missing or included erroneously.
   - Expected "a0 holds three states from now" but subject wrote "a0 holds now"
   - Expected a0 U (a1 and F(a2)) but subject wrote (a0 U a1) and F(a2)
6. **BadStateQuantification**: Applies to responses that mis-use or swap a fan-out operator (F, G, U).
   - Expected "a0 eventually holds" but subject wrote "a0 always holds"
   - Expected a0 U a1 but subject wrote G(a0) U a1
7. **ExclusiveU**: Applies to responses that assume an until is satisfied only when both the right subterm and the negation of the left subterm hold.
   - Expected a0 U (!a0 and a1) but subject wrote a0 U a1
8. **ImplicitF**: Applies to responses that either ignore or introduce an F quantifier.
   - Expected "whenever a0, then a1 in the next state" but subject wrote "a0 and a1 alternate forever"
   - Expected F(a0) but subject wrote a0
9. **ImplicitG**: Applies to responses that either ignore or introduce a G quantifier.
   - Expected "a0 always holds" but subject wrote "a0 holds now"
   - Expected G(a0 => G(a1)) but subject wrote G(a0 => a1)
10. **OtherImplicit**: Applies to underconstrained responses that are not covered by the ImplicitF and ImplicitG tags.
    - Expected "whenever a0 holds then a1 also holds" but subject wrote "a0 holds now and a1 holds"
    - Expected !a0 U a0 but subject wrote F(a0)
    - Expected a0 U G(!a0) but subject wrote a0 and F(G(!a0))
    - Expected F(a0) and G(a0 => XG(!a0)) but subject wrote F(a0 and XG(!a0))
11. **WeakU**: Applies to responses that confuse the U operator with the weak variant W, which does not guarantee that its second subterm eventually holds.
    - Expected "a0 holds for a finite number of states" but subject wrote "a0 always holds"
    - Expected a0 U a1 but subject wrote F(a1) and a0 U a1

Figure 4 Abstract code book of semantic errors

and one or more examples showing an expected response and an incorrect response. The descriptions use a high-level wording that may be specialized to LTL formulas and English text, such as the formulas and texts in the examples.

The first three tags deal with high-level issues in a particular order: Precedence is for LTL formulas (not English responses) that are correct up to missing parentheses, ReasonableVariant is for LTL formulas that are correct for a slightly different reading of the English question than intended, and Unlabeled is for complex or ambiguous responses. All remaining tags can be studied for any response in any order.

For readers interested in using this code book (either on our instrument questions or on related questions), we offer the following tips and experiences:

- The ImplicitG and ImplicitF responses were of two kinds: they either ignored or added a quantifier. In the LTL ▷ Eng direction, added quantifiers were more common than ignored ones. In the Eng ▷ LTL direction, ignored quantifiers were more common.
- The ExclusiveU tag did not apply to any Eng ▷ LTL responses. Indeed, we recognized the need for this tag only after labeling all LTL responses and moving on to the English responses.
- Neither the BadProp nor the BadStateQuantification tags applied to any LTL ▷ Eng responses. We attribute this our choice of questions; different questions would likely show a need for both tags.

6 Trace Satisfaction

Table 4 tabulates the evidence that we found for various misconceptions among the trace satisfaction questions. The leftmost column lists misconceptions and the following columns count evidence from answers in Rounds 2, 3, and 4.[1] The first six rows correspond to misconceptions that we intentionally tested with the trace satisfaction questions. The seventh row shows that four responses gave (unanticipated) evidence of ImplicitG, even though we did not explicitly test for it.

The responses support several misconceptions:

- WeakU (N=29 R2, N=2 R3, N=9 R4), ImplicitF (N=12 R2, N=2 R4), and ExclusiveU (N=9 R2, N=1 R3, N=1 R4) are especially problematic.
- Most of the ImplicitG errors (3 of 4) arose from formulas that constrain a single state (e.g., X(Red)). The other error is from an until formula; it suggests an implicit always on the right side.
- Every BadStateIndex error is due to a mixup about which single state an X-wrapped term refers to.

[1] Note that one answer can increase the counts in multiple rows, provided the response contains more than one mistake. There are no such trace satisfaction responses, but several among Eng ▷ LTL and LTL ▷ Eng data.

- The OtherImplicit errors express a desire for F to prevent flickering; specifically, for F(G(Red)) to be satisfied only by traces in which the first Red state is the beginning of an always-Red suffix.

There are far fewer errors among the Round 3 answers. Nevertheless, both WeakU and ExclusiveU may be issues for even these experienced LTL users. Recall also that Round 3 subjects received a random subset of the question pool (section 3.2.3) rather than every question.

Cautions The findings need to be interpreted in the following context:

- A trace satisfaction question pairs a formula with a *specific* trace. It is quite possible that small changes to the trace may affect the responses.
- Round 2 presented traces using text rather than images; refer to figure 2 for an example. The notation led to identifiable confusion in thirteen responses (23% R2), but may have misled others in subtle ways.
- Some of the ImplicitF errors may in fact be due to confusion about the start state. Subjects might have assumed that trace matching is allowed to skip a prefix of the trace. However, none of the Round 4 talk alouds contain evidence of this confusion.
- The Round 3 question pool accidentally *omitted* a question that asked whether X constrains two states, the current and next state, rather than the next alone. But given that Rounds 2 and 4 subjects did well on this question (95% R2, 100% R4 correct), it seems unlikely that it would have gathered any mistakes.

Key Takeaway Subjects had trouble with the basic semantics of F (OtherImplicit), U (WeakU), and unqualified formulas (ImplicitF, ImplicitG).

7 LTL to English

Table 5a shows evidence of misconceptions in translating LTL to English. Overall, 23% of the Round 2 responses, 15% of the Round 3 responses, and 24% of the Round 4 responses were incorrect.

- Half of the Round 2 population and most (9 of 11) of the Round 4 population exhibited the WeakU misconception. Even in Round 3, it caught two people.
- The BadStateIndex errors are of three types, with some overlap: applying the right side of an implication to the next state (N=4 R2, N=1 R4), misinterpreting the scope of an F (N=6 R2, N=1 R3), and misinterpreting the state that an X-wrapped term refers to (N=12 R2, N=2 R3, N=2 R4).
- ImplicitG is common in all populations. In Rounds 3 and 4, an unqualified formula prompted most (N=4 R3, N=4 R4) of the incorrect responses: R => X(X(X(R))). Eight of the ImplicitG mistakes in Round 2 were for the same formula. The others were from a missing G on the right of an until (N=2) and a missing top-level G (N=3).
- The eight OtherImplicit responses are due to one question: G(Red => X(!Red) and X(X(Red))). These responses said that Red blinks forever, which is true only if Red holds in the first or second state.

Table 4 Trace satisfaction errors

Misconception	R2	R3	R4
BadProp	1	-	-
BadStateIndex	7	1	-
ImplicitF	12	-	2
ExclusiveU	9	1	1
OtherImplicit	3	-	1
WeakU	29	2	9
ImplicitG	3	-	1

- One LTL ▷ Eng question tested whether X(ao) entails !ao in the current state, because several Round 1 responses had that mistake. None of the subjects in Rounds 2, 3, and 4 made this error.

Cautions Our findings need some care in interpretation:

- Because the primary output here is English, which is not a formal language, our two-coder method may have mislabeled some written responses. Though in general we found a high level of articulation in the English, there are issues, e.g., some of it was written by subjects who are not native speakers and some of it lacks punctuation.
- The "blinker" errors (OtherImplicit) suggest a kind of confirmation bias among subjects. If the questions had asked for one satisfying and one non-satisfying trace in addition to a translation, subjects might not have made these errors.

Key Takeaway In general, subjects did well at this task. The most common error was that subjects expressed a correct constraint at an incorrect time (BadStateIndex).

8 English to LTL

Table 5b shows the evidence of misconceptions in translating LTL to English. This corresponds to the direction of authoring LTL based on requirements. The error rates are much higher than for the previous questions: 56% of the Round 2 responses, 28% of the Round 3 responses, and 47% of the Round 4 responses were incorrect. We note that the mistakes are spread across most of the tags.

- ImplicitG is the most common error. In all rounds, every question received at least one response that was missing either a toplevel G or one around a subterm.
- OtherImplicit arose in at least two forms. In Round 3, the OtherImplicit all assumed an "eager" semantics for F. Section 9.2.1 discusses this point further. In Rounds 2 and 4, some responses assumed that variables did not change unless specified. Almost all of these were misuses of the F operator. Because variables represented two different things in these rounds—namely, lights on a panel in Round 2 and features on a robot train in Round 4—this may be a general issue.

Table 5 LTL to English and English to LTL errors

(a) LTL to English errors

Misconception	R2	R3	R4
BadProp	-	-	-
BadStateIndex	18	3	3
BadStateQuantification	-	1	-
ExclusiveU	15	1	5
ImplicitF	4	-	-
ImplicitG	13	5	4
OtherImplicit	7	1	-
WeakU	26	2	6
Unlabeled	1	-	-

(b) English to LTL errors

Misconception	R2	R3	R4
BadProp	8	2	6
BadStateIndex	11	3	3
BadStateQuantification	22	1	2
ExclusiveU	-	-	-
ImplicitF	10	-	3
ImplicitG	47	7	10
OtherImplicit	30	4	8
WeakU	2	-	-
Unlabeled	3	1	1
Precedence	2	-	-
ReasonableVariant	18	1	-

- BadStateQuantification formulas typically contained an extra operator; they rarely swapped one operator for another. A common extra-operator mistake was to write the following formula, which is unsatisfiable: `G(Red) U !Red`.
- As noted in section 5, ReasonableVariant applied in cases where subjects used a different interpretation of the question than what we had in mind. We reworded questions between Round 2 and Round 3 in an effort to clarify, which may explain the drop in the presence of this tag.
- BadStateIndex often arose in connection with a binary logical operator. Subjects assumed that the right sides of conjunctions (`and`) and implications (`=>`) applied to the *next* state, rather than to the *current* one.
- Although there are few ExclusiveU and WeakU errors, the responses gave us little confidence that subjects can properly use the `U` operator. There were many misapplications that fell under the BadStateQuantification and BadStateIndex codes.
- LTL has limits on its expressive power [71]; not every property in English can be translated into it. Subjects were aware of this fact, and in Round 1 we saw 4 instances of subjects responding to English sentences with the claim that they were not expressible in LTL. All these responses were incorrect.

  Because the boundaries of LTL expressiveness were not a focus of our study, we presented only English sentences that were LTL-expressible. We did, however, carry over two of the Round 1 questions that prompted "inexpressible" responses. Three such responses arose in Round 2. None arose in Round 3 or Round 4.

Cautions Our results carry the following caveats:

- We focused on coding incorrect responses. However, there could be misconceptions lurking in correct responses also. For instance, consider `((a0 until a1) and F(a1))` and the same formula without the unnecessary `F`. The two are semantically equivalent, but the former suggests a WeakU misconception. Through spot-checks we found

some evidence for misconceptions lurking in correct answers, but we did not conduct a systematic study.

- Round 2 subjects were accustomed to writing LTL in the context of Electrum, but were asked to fill out the survey checking only for syntactic validity. Access to Electrum's semantic checks may have reduced errors. On the other hand, some subjects may have used their preferred tooling despite the instructions (as we have seen in other settings where we have deployed similar instruments).

Key Takeaway By contrast to the other question types, the Eng ▷ LTL direction was fraught with errors and provides evidence for a large number of misconceptions. Unfortunately, this task is perhaps the most important of the three. A user has to write correct LTL in order to apply the logic.

9 Implications for Tool Builders and Language Designers

Our findings motivate two concrete suggestions for tool builders and two general suggestions for the designers of LTL-based languages.

9.1 Implications For Tool Builders

9.1.1 "Binary After" Misconception

Upon close inspection, some of the incorrect formulas in Round 2 used the unary after operator (Electrum's version of X) as a binary operator. One example follows, with two red rules (▁) to mark important whitespace:

Q. Translate to LTL: Whenever the Red light is on, it turns off in the next state and on again in the state after that.

A. always (Red ▁ after (not Red) and not Red ▁ after (Red))

This is a perfectly natural use of the English word "after," which connects two clauses in the same manner as "until" does; indeed, a unary "after" makes no sense in English. But in Electrum, after behaves quite differently. Electrum reads this formula as having *three* and connectives, two of which implicitly appear at the marked whitespace. Thus, the formula is a syntactically-valid but semantically-incorrect answer.

A binary use of after should ideally be a syntax error. If such a change is not possible, we recommend using a different keyword (such as next_state) to avoid the confusion with English. This problem, known as a *vernacular misconception* [19], has also been found in programming languages [58, 59].

9.1.2 "Implicit-And" Restriction

As the example above demonstrates, Electrum implicitly puts an and between terms separated by whitespace. For the authors and maintainers of large formulas, this shorthand is very helpful. One useful and intentional use of implicit-and is the following predicate for a fully-lit traffic light:

```
pred allLightsAreLit {
  Red in Traffic.lit
  Yellow in Traffic.lit
  Green  in Traffic.lit
}
```

In other formulas, though, implicit-and leads to syntactically-valid formulas that are incorrect in subtle ways.

We raised this issue with the Alloy Board and they agreed to change Electrum. Henceforth, implicit-and will appear only at newline breaks, and not between space-separated or tab-separated terms. With this change, the traffic light formula remains valid and the formula from section 9.1.1 raises a syntax error.

9.2 Implications For Logic Designers

9.2.1 More Control over Backtracking

Four of the LTL responses from experts in Round 3 had OtherImplicit errors. All four suggest a desire for an F operator that does not backtrack. Twelve of the Round 2 and three of the Round 4 responses are similar. Here is one example from Round 3:

Q. Translate to LTL: The Red light is on in exactly one state, but not necessarily the first state.

A. eventually(Red and after(always not Red))

The answer would be correct if LTL required the first Red state in a trace to satisfy the entire term under the F operator. Authors of a new LTL language might consider adding such a variant of F to their toolbox, or perhaps a Prolog-like cut operator [16] to disallow backtracking at the first occurrence of a Red state. Alternatively, a language might introduce shorthand for the many applications of F that Menghi et al. [50] propose as core movement patterns; a *strict ordered visit* would suffice here.

A recent robotics paper [5] has the same no-backtracking assumption, and we have also seen it in colloquium talks in robotics. However, this assumption does not seem to be limited to the robotics community: formal methods researchers in Round 3 are also responsible for OtherImplicit errors.

9.2.2 Explicit State Index

The BadStateIndex errors in our data indicate significant confusion about when an LTL term goes into effect. A language might address these errors by providing more control over the state index, or perhaps an explicit representation of the index.

One common form of BadStateIndex error is the "and then" problem: writing (a0 and a1) instead of (a0 and X(a1)). Subjects in both Round 2 (N=2) and Round 3 (N=2) had this specific problem; others (N=9 R2, N=1 R3, N=3 R4) made a similar mistake with implication. PSL includes a non-overlapping suffix-implication operator with exactly this semantics [26]. Our work provides support for this operator and suggests the need for a suffix-and as well.

A second kind of state index error involved the use of F to connect two toplevel terms (N=4 R2):

Q. Translate to LTL: The Red light is on in exactly one state, but not necessarily the first state.

A. (not Red until Red) and
eventually(Red => after(always not Red))

One could easily fix this particular formula by shuffling the after term into the right conjunct of the until. But, keeping in mind Electrum's implicit and, the human authors and maintainers of LTL specifications might benefit from a more direct representation perhaps using labels to bind and refer to state indices.

10 Implications for Educators

Both our instruments (appendix A) and our inventory of misconceptions (figure 4) are each independently of use. Educators may wish to apply these in a traditional classroom setting or as part of a workplace training program. It is worth noting that we cannot guarantee that incorrect answers are bijective with misconceptions, so some further interpretation of wrong answers is necessary.

Once we have found misconceptions, how can they be addressed? A natural tendency is to assume that learners can just be "taught right." While that may work in some cases—e.g., by incorporating these findings into how LTL is introduced—research suggests that that may not suffice.

There is an extensive literature on understanding misconceptions. The US National Research Council [19] refines them into five categories; some, like preconceived notions, are very unlikely to apply here, while others seem especially relevant: vernacular misconceptions are those caused by using words with other natural language meanings, and conceptual misunderstandings are those that arise when instruction permits learners to create faulty models.

Many techniques have been proposed to help learners overcome misconceptions. Researchers have found success using *concept maps* [54]. The work of Posner et al. [62] presents a theory of conceptual change, at the heart of which is the *refutation text* [77]. Studies show that these work in some cases or domains, but sometimes not in others. In general, ideas from physical science education do not necessarily carry over directly to LTL because learners form conceptual understandings of the physical world due to their daily interaction with it (e.g., gravitation), which seems unlikely for most dimensions of understanding LTL. In general, however, the literature makes clear [13, 46, 67] that direct instruction alone is unlikely to overcome misconceptions; activities that are more learner-centric are much more likely to be effective.

11 Threats to Validity

Internal Validity Coding inherently contains various biases. Our high inter-coder reliability score only indicates that the coders have aligned their biases, not that they

have eliminated them. Nevertheless, we believe the codes we have arrived at are reasonable. The supplement lets others review our coding.

Quizius has two threats that can cause it to overlook an interesting question: ordering and timing. If the first few recipients of a difficult question happen to get it correct, then the question receives less attention going forward. And if a question arrives late during the quiz period, it has few opportunities to gather answers.

External Validity Rounds 1, 2, and 4 are based on students at the same institution, in the same course, and with the same instructor. Relaxing each of these factors can cause different outcomes. For instance, the instructor's level of comfort with the natural language can be a factor. Even the amount of exposure could help: apparent misconceptions in Round 1 that did not replicate in Rounds 2 and 3 may be because of the limited prep time Round 1 subjects had due to COVID-19. Although Round 3 supports a large number of misconceptions and suggests that our observations generalize beyond the Round 1 population, repeating the process on a much broader population would be valuable and may reveal new misconceptions.

Ecological Validity Our approach definitely lacks ecological validity: subjects were answering questions about LTL absent a concrete use. Studying subjects as they use a tool (perhaps using talk-alouds), while much harder to run in a controlled fashion, would help us identify how these misconceptions are handled in practice.

It is also worth noting that using LTL in context, with a tool, will not automatically catch mistakes: if the system-under-test and formula are incorrect in the same way, no counter-example will help the user correct their misconception. For that reason, studies of our kind are still useful and provide concrete targets for more ecologically valid studies.

Construct Validity Our work suffers from some construct validity as well. Apparent misconceptions could be artifacts of the specific wording or presentation in our instruments (as we have already seen to some extent: section 5), or could be evidence of some other misconception that we had not identified, such as the confusion about rising edges [63]. The main way to mitigate these concerns is both to perform further validation steps and to tie our results to methods that improve ecological validity. One other construct validity concern would be if our work was misinterpreted as claiming to capture *all* misconceptions, but we make no such claim.

Conclusion Validity Our work is intentionally light on "conclusions," for two reasons. First, as the first work of its kind, it is inherently formative. Second, our main product is *instruments* that others can use; it is those uses that might arrive at conclusions, whose validity the users would have to justify. Our main conclusion here is that there are misconceptions in the understanding of LTL, and that we have instruments that can identify some of them with reasonable confidence. The former seems inherently likely: it is hard to imagine, given all the misconceptions about other formal artifacts, that LTL is a solitary exception. For the latter, we believe our multiple rounds and forms of validation lend some credibility to our instruments.

12 Related Work

Despite the long and distinguished history of literature on temporal logic (e.g., [15, 44, 48, 57, 60, 61, 74]), to our knowledge, there are no in-depth user studies on patterns of misconception. The only exception we know is a recent study that compares LTL to two similar logics [20]. Although that study has some information about how learners fare with LTL, the focus is on comparison *between* the three languages, not on why learners struggle with LTL specifically.

Of course, it is not news that properties might contain mistakes! Vacuity checking [10, 14, 28, 42, 51], for instance, guards against one kind of error. Other works have addressed other specific specification errors: e.g., [8, 38, 55]. However, all these focus on particular *mistakes*; they do not imply a misunderstanding of the logic itself. Furthermore, even a property that passes the above checks can be *wrong* due to a misunderstanding of the logic. Our focus is thus on understanding, which in turn can generate new checkers.

Several authors have created temporal logic formula templates by examining families of examples. Dwyer et al. [25] identify a taxonomy of specification patterns based on industrial and academic examples. Menghi et al. [50] introduce patterns for robotics missions drawn from hundreds of natural-language specifications. Rajhans et al. [63] identify several template formulas, developed through conversations with industry partners, presented through a graphical tool. Our work is different in its focus on providing fundamental insights into users' misconceptions (often expressed through mistakes), rather than patterns of normal use.

There is a longstanding debate on the relative merits of linear-time (e.g., LTL) versus branching-time (e.g., CTL [15]) logics. This debate is *emphatically not* the topic of this paper; we do not claim to address what is most "intuitive" [72]. If we had begun this process with a different logic, such as CTL, we might well have surfaced other errors not seen here. We stress that it would not suffice to reuse our prompts in a different setting: vital features of the logic might not be exercised and much insight would be missed.

The linear vs. branching debate led to new logics in the early 2000s. IBM introduced Sugar [9], which mixes linear- and branching-time semantics and later grew into PSA [26]. Likewise, Intel released the language ForSpec [6]. Both languages are motivated by the desire to improve the user experience, and both also make central claims about usability: e.g., Sugar claims that hardware engineers can "easily and intuitively specify their designs". However, these claims are asserted without rigorous justification, and neither is accompanied by any catalog of remaining difficulties of the form that we have found.[2]

In terms of designing new logics, there is a broad literature on misconceptions and design methods in programming languages, such as classic works by Pea [59], Pane

[2] From personal communication with Moshe Vardi, we learned that many design choices in ForSpec, such as using English words instead of mathematical symbols, were indeed based on extensive (but informal) discussions with different types of users.

and Myers [58], and others. The methods, if not the findings, of these works would be useful to the designers of new logics.

The Wason Selection Task [75] has been used to argue that humans may not reason using the rules of logic (though others have argued the result is contextual [18, 69]). In contrast, our work does not study *reasoning*, only on understanding of the logic itself. However, some results on how context impacts semantics [69] may eventually link the two efforts.

Finally, our English-to-LTL exercises resemble those in Iltis [30], a tool for teaching logic. It is possible that Iltis would provide a good framework for studies like ours, although the tool itself focuses on pedagogy, rather than studies of misconceptions.

13 Discussion

Other Question Types Our study and results suggest several more types of questions that would yield additional insight. For instance, we did not ask subjects to explicitly create traces corresponding to formulas; in particular, there would be value to asking for both correct and (near-miss) incorrect traces, to probe their understanding of the space described by a formula. One can similarly go from formulas to traces. We could also test their understanding of the (in)equality of pairs of LTL formulas. Other studies [20, 64] contain instruments that would also be useful to adapt.

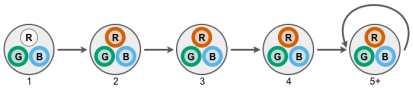
Other Logics Naturally, this style of work can be applied to other property languages as well, such as CTL. Writing correct specifications for complex systems is difficult in any formal language, especially because the requirements and the underlying system may change over time (see discussions in [23, 29]). It is therefore critical to identify specification errors and correct them. Our processes suggest a possible approach, and (for similar logics) our instruments provide a *starting* guide. Whatever the process used, we stress the importance of using mechanisms that mitigate expert blind spots such as Quizius. In fact, four questions in the final instruments were generated by subjects in Round 1 (A.2.1, A.3.2, A.3.3,and A.3.5).

Conclusion With the use of LTL on the rise, it is important to understand its ergonomics well. Our studies constitute a first, formative step in this direction, and have already had some language design impact. The errors that we found were consistent across two syntaxes, two domains, and a variety of subjects, which suggests that the misconceptions are robust. We also suggest some high-level consequences for the design of new property languages. Given that relatively simple formulas generated our findings, we suspect many other issues lurk in more complicated formulas.

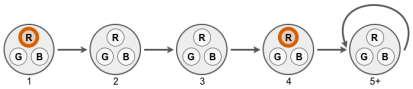
A Instrument

A.1 Trace Satisfaction

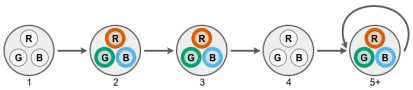
A.1.1 Is the formula Red satisfied by this trace?



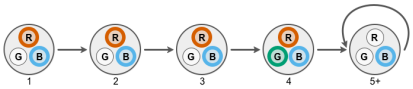
A.1.2 Is the formula after(after(after(Red))) satisfied by this trace?



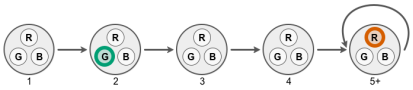
A.1.3 Is the formula always(Red => after(after(after(Red)))) satisfied by this trace?



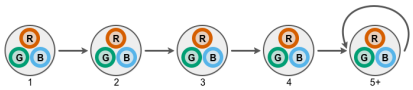
A.1.4 Is the formula ((after Red) until (after Green)) satisfied by this trace?



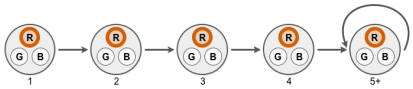
A.1.5 Is the formula ((eventually Red) and (eventually Green)) satisfied by this trace?



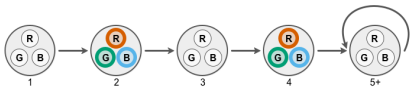
A.1.6 Is the formula after(after(eventually(Red))) satisfied by this trace?



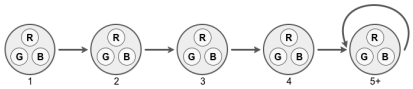
A.1.7 Is the formula (Red until Blue) satisfied by this trace?



A.1.8 Is the formula eventually(always(Red)) satisfied by this trace?



A.1.9 Is the formula always(Red => Green) satisfied by this trace?



A.2 LTL to English

A.2.1 Translate to English: Red => after(after(after(Red)))

A.2.2 Translate to English: after(after(eventually(after(Red))))

A.2.3 Translate to English: ((eventually Red) => (always Blue))

A.2.4 Translate to English: ((Red until Blue) and always(Red))

A.2.5 Translate to English: always(Red => (after(not Red) and after(after(Red))))

A.3 English to LTL

A.3.1 Translate to LTL: Whenever the Red light is on, it is off in the next state and on again in the state after that.

A.3.2 Translate to LTL: The Red light is on in exactly one state, but not necessarily the first state.

A.3.3 Translate to LTL: The Red light cannot stay on for three states in a row.

A.3.4 Translate to LTL: Whenever the Red light is on, the Blue light will be on then or at some point in the future.

A.3.5 Translate to LTL: The Red light is on for zero or more states, and then turns off and remains off in the future.

References

[1] Vicki L. Almstrum, Peter B. Henderson, Valerie J. Harvey, Cinda Heeren, William A. Marion, Charles Riedesel, Leen-Kiat Soh, and Allison Elliott Tew. "Concept Inventories in Computer Science for the Topic Discrete Mathematics". In: *ACM SIGCSE Bulletin* 38.4 (2006), pages 132–145.

[2] Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan. "A Framework for Transforming Specifications in Reinforcement Learning". In: *CoRR* abs/2111.00272 (2021).

[3] Gal Amram, Suguman Bansal, Dror Fried, Lucas Martinelli Tabajara, Moshe Y. Vardi, and Gera Weiss. "Adapting Behaviors via Reactive Synthesis". In: *CAV*. 2021, pages 870–893.

[4] Marco Antoniotti and Bud Mishra. "Discrete Events Models + Temporal Logic = Supervisory Controller: Automatic Synthesis of Locomotion Controllers". In: *ICRA*. 1995, pages 1441–1446. DOI: 10.1109/ROBOT.1995.525480.

[5] Brandon Araki, Xiao Li, Kiran Vodrahalli, Jonathan A. DeCastro, Micah J. Fry, and Daniela Rus. "The Logical Options Framework". In: *ICML*. Volume 139. 2021, pages 307–317.

[6] Roy Armoni, Limor Fix, Alon Flaisher, Rob Gerth, Boris Ginsburg, Tomer Kanza, Avner Landver, Sela Mador-Haim, Eli Singerman, Andreas Tiemeyer, Moshe Y. Vardi, and Yael Zbar. "The ForSpec Temporal Logic: A New Temporal Property-Specification Language". In: *TACAS*. 2002, pages 296–311.

[7] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. "The Nonstochastic Multiarmed Bandit Problem". In: *SIAM J. Comput.* 32.1 (2002), pages 48–77.

[8] Derek L. Beatty and Randal E. Bryant. "Formally Verifying a Microprocessor Using a Simulation Methodology". In: *Conf. Design Auto.* 1994, pages 596–602.

[9] Ilan Beer, Shoham Ben-David, Cindy Eisner, Dana Fisman, Anna Gringauze, and Yoav Rodeh. "The Temporal Logic Sugar". In: *CAV*. 2001, pages 363–367.

[10] Ilan Beer, Shoham Ben-David, Cindy Eisner, and Yoav Rodeh. "Efficient Detection of Vacuity in ACTL Formulas". In: *CAV*. Volume 1254. 1997, pages 279–290.

[11] Amit Bhatia, Lydia E. Kavraki, and Moshe Y. Vardi. "Sampling-based motion planning with temporal goals". In: *ICRA*. 2010, pages 2689–2696.

[12] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. "Synthesis of Reactive(1) designs". In: *Journal of Computer and System Sciences* 78.3 (2012), pages 911–938.

[13] M. Cakir. "Constructivist Approaches to Learning in Science and Their Implications for Science Pedagogy: A Literature Review". In: *Intl. J. Env. & Sci. Ed.* 3.4 (2008), pages 193–206.

[14] Hana Chockler and Ofer Strichman. "Easier and More Informative Vacuity Checks". In: *MEMOCODE*. 2007, pages 189–198.

[15] Edmund M. Clarke and E. Allen Emerson. “Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic”. In: *Logics of Programs*. Volume 131. 1981, pages 52–71.

[16] W. F. Clocksin and C. S. Mellish. *Programming in Prolog*. 2nd edition. Springer–Verlag, 1984.

[17] Jacob Cohen. “A Coefficient of Agreement for Nominal Scales”. In: *Educational and Psychological Measurement* 20 (1960), pages 37–46.

[18] Leda Cosmides and John Tooby. “Cognitive Adaptations for Social Exchange”. In: *The Adapted Mind: Evolutionary Psychology and the Generation of Culture*. 1992, pages 163–228.

[19] National Research Council. *Science Teaching Reconsidered: A Handbook*. National Academies Press, 1997.

[20] Christoph Czepa and Uwe Zdun. “On the Understandability of Temporal Properties Formalized in Linear Temporal Logic, Property Specification Patterns and Event Processing Language”. In: *IEEE Trans. Soft. Eng.* 46.1 (2020), pages 100–112.

[21] Daniel Jackson. *Alloy: a language & tool for relational models*. http://alloy.mit.edu/alloy/. Accessed November 1, 2016. 2016.

[22] Holger Danielsiek, Wolfgang Paul, and Jan Vahrenhold. “Detecting and Understanding Students’ Misconceptions Related to Algorithms and Data Structures”. In: *Special Interest Group on Computer Science Education*. 2012, pages 21–26.

[23] Richard A. DeMillo, Richard J. Lipton, and Alan J. Perlis. “Social Processes and Proofs of Theorems and Programs”. In: *Comm. ACM* 22.5 (1979), pages 271–280.

[24] Paul Denny, John Hamer, Andrew Luxton-Reilly, and Helen C. Purchase. “PeerWise: Students Sharing their Multiple Choice Questions”. In: *ICER*. 2008, pages 51–58.

[25] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. “Patterns in Property Specifications for Finite-State Verification”. In: *ICSE*. 1999, pages 411–420.

[26] Cindy Eisner and Dana Fisman. *A Practical Introduction to PSL*. Springer, 2006.

[27] Georgios E. Fainekos, Hadas Kress-Gazit, and George J. Pappas. “Temporal Logic Motion Planning for Mobile Robots”. In: *ICRA*. IEEE, 2005, pages 2020–2025.

[28] Dana Fisman, Orna Kupferman, Sarai Sheinvald-Faragy, and Moshe Y. Vardi. “A Framework for Inherent Vacuity”. In: *HVC*. Volume 5394. 2008, pages 7–22.

[29] William Gasarch. *I went to the “debate” about Program Verif and the Lipton-Demillo-Perlis paper*. Accessed 2022-03-24. URL: https://blog.computationalcomplexity.org/2021/06/i-went-to-debate-about-program-verif.html.

[30] Gaetano Geck, Artur Ljulin, Sebastian Peter, Jonas Schmidt, Fabian Vehlken, and Thomas Zeume. “Introduction to Iltis: an interactive, web-based system for teaching logic”. In: *ITiCSE*. 2018, pages 141–146.

[31] B. Glaser and A. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Sociology Press, 1967.

[32] Kenneth J. Goldman, Paul Gross, Cinda Heeren, Geoffrey L. Herman, Lisa C. Kaczmarczyk, Michael C. Loui, and Craig B. Zilles. "Identifying Important and Difficult Concepts in Introductory Computing Courses using a Delphi Process". In: *Special Interest Group on Computer Science Education*. 2008, pages 256–260.

[33] David Gundana and Hadas Kress-Gazit. "Event-Based Signal Temporal Logic Synthesis for Single and Multi-Robot Tasks". In: *IEEE Robotics Autom. Lett.* 6.2 (2021), pages 3687–3694.

[34] Geoffrey L. Herman, Michael C. Loui, and Craig B. Zilles. "Creating the Digital Logic Concept Inventory". In: *Special Interest Group on Computer Science Education*. 2010, pages 102–106.

[35] David Hestenes. "Toward a modeling theory of physics instruction". In: *Am. J. Phys.* 55.5 (1987), pages 440–454.

[36] David Hestenes, Malcolm Wells, and Gregg Swackhamer. "Force concept inventory". In: *The physics teacher* 30.3 (1992), pages 141–158.

[37] Gerard J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, 2003.

[38] Yatin Vasant Hoskote, Timothy Kam, Pei-Hsin Ho, and Xudong Zhao. "Coverage Estimation for Symbolic Model Checking". In: *Conf. Design Auto.* 1999, pages 300–305.

[39] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. 2nd edition. MIT Press, 2012.

[40] Yiannis Kantaros and Michael M. Zavlanos. "STyLuS*: A Temporal Logic Optimal Control Synthesis Algorithm for Large-Scale Multi-Robot Systems". In: *Int. J. Robotics Res.* 39.7 (2020), pages 812–836.

[41] Barbara A. Kitchenham, Shari Lawrence Pfleeger, Lesley Pickard, Peter W. Jones, David C. Hoaglin, Khaled El Emam, and Jarrett Rosenberg. "Preliminary Guidelines for Empirical Research in Software Engineering". In: *IEEE Trans. Software Eng.* 28.8 (2002), pages 721–734.

[42] Orna Kupferman and Moshe Y. Vardi. "Vacuity detection in temporal model checking". In: *Int. J. Softw. Tools Tech. Transf.* 4.2 (2003), pages 224–233.

[43] Morteza Lahijanian, Shaull Almagor, Dror Fried, Lydia Kavraki, and Moshe Vardi. "This Time the Robot Settles for a Cost: A Quantitative Approach to Temporal Logic Planning with Partial Satisfaction". In: *AAAI*. 2015, pages 3664–3671.

[44] Leslie Lamport. "What Good is Temporal Logic?" In: *Inf. Proc.* 1983, pages 657–668.

[45] S.G. Loizou and K.J. Kyriakopoulos. "Automatic synthesis of multi-agent motion tasks based on LTL specifications". In: *CDC*. Volume 1. 2004, pages 153–158.

[46] J. Longfield. “Discrepant Teaching Events: Using an Inquiry Stance to Address Students’ Misconceptions”. In: *Intl. J. Teach. and Learn. in Higher Ed.* 21.2 (2009), pages 266–271.

[47] Nuno Macedo, Julien Brunel, David Chemouil, Alcino Cunha, and Denis Kuperberg. “Lightweight Specification and Analysis of Dynamic Systems with Rich Configurations”. In: *FSE*. 2016, pages 373–383.

[48] Zohar Manna and Pierre Wolper. “Synthesis of Communicating Processes from Temporal Logic Specifications”. In: *TOPLAS* 6.1 (1984), pages 68–93.

[49] Shahar Maoz and Jan Oliver Ringert. “Reactive Synthesis with Spectra: A Tutorial”. In: *ICSE*. 2021, pages 320–321.

[50] Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi, and Thorsten Berger. “Specification Patterns for Robotic Missions”. In: *IEEE Trans. Software Eng.* 47.10 (2021), pages 2208–2224.

[51] Kedar S. Namjoshi. “An Efficiently Checkable, Proof-Based Formulation of Vacuity in Model Checking”. In: *CAV*. Volume 3114. 2004, pages 57–69.

[52] Mitchell J. Nathan and Anthony Petrosino. “Expert Blind Spot Among Preservice Teachers”. In: *Amer. Educational Research J.* 40.4 (2003), pages 905–928.

[53] Mitchell J Nathan, Kenneth R Koedinger, Martha W Alibali, et al. “Expert blind spot: When content knowledge eclipses pedagogical content knowledge”. In: *Intl. Conf. Cog. Sci.* 2001, pages 644–648.

[54] J. D. Novak and D. B. Gowin. *Learning how to learn*. Cambridge University Press, 1984.

[55] Martin Oberkönig, Martin Schickel, and Hans Eveking. “A Quantitative Completeness Analysis for Property-Sets”. In: *FMCAD*. 2007, pages 158–161.

[56] Liam O’Connor and Oscar Wickström. “Quickstrom: Property-based Acceptance Testing with LTL Specifications”. In: *PLDI*. 2022, To appear.

[57] Susan Owicki and Leslie Lamport. “Proving Liveness Properties of Concurrent Programs”. In: *TOPLAS* 4.3 (1982), pages 455–495.

[58] John F. Pane and Brad A. Myers. *Usability Issues in the Design of Novice Programming Systems*. Technical report CMU-CS-96-132. Carnegie Mellon University, 1996.

[59] Roy D. Pea. “Language-Independent Conceptual “Bugs” in Novice Programming”. In: *J. Ed. Comp. Research* 2.1 (1986), pages 25–36.

[60] Amir Pnueli. “The Temporal Logic of Programs”. In: *Foundations of Computer Science*. 1977, pages 46–57.

[61] Amir Pnueli and Roni Rosner. “On the Synthesis of a Reactive Module”. In: *POPL*. 1989, pages 179–190.

[62] G. J. Posner, K. A. Strike, P. W. Hewson, and W. A. Gertzog. “Accommodation of a Scientific Conception: Toward a Theory of Conceptual Change”. In: *Sci. Edu.* 66.2 (1982), pages 211–227.

[63] Akshay Rajhans, Anastasia Mavrommati, Pieter J. Mosterman, and Roberto G. Valenti. “Specification and Runtime Verification of Temporal Assessments in Simulink”. In: *Runtime Verification*. 2021, pages 288–296.

[64] Phyllis Reisner. “Human Factors Studies of Database Query Languages: A Survey and Assessment”. In: *ACM Comput. Surv.* 13.1 (1981), pages 13–31.

[65] Sam Saarinen. “Query Strategies for Directed Graphical Models and their Application to Adaptive Testing”. PhD thesis. Brown University, 2021.

[66] Sam Saarinen, Shriram Krishnamurthi, Kathi Fisler, and Preston Tunnell Wilson. “Harnessing the Wisdom of the Classes: Classsourcing and Machine Learning for Assessment Instrument Generation”. In: *Special Interest Group on Computer Science Education*. 2019, pages 606–612.

[67] L. Savion. “Clinging to discredited beliefs: The larger cognitive story”. In: *J. Schol. of Teach. and Learn.* 9.1 (2009), pages 81–92.

[68] Ankit Shah, Pritish Kamath, Julie A. Shah, and Shen Li. “Bayesian Inference of Temporal Task Specifications from Demonstrations”. In: *NeurIPS*. 2018, pages 3808–3817.

[69] Keith Stenning and Michiel van Lambalgen. *Human Reasoning and Cognitive Science*. MIT Press, 2008.

[70] Allison Elliott Tew and Mark Guzdial. “Developing a Validated Assessment of Fundamental CS1 Concepts”. In: *Special Interest Group on Computer Science Education*. 2010, pages 97–101.

[71] Moshe Y. Vardi. “An Automata-Theoretic Approach to Linear Temporal Logic”. In: *Banff Workshop*. Volume 1043. 1995, pages 238–266.

[72] Moshe Y. Vardi. “Branching vs. Linear Time: Final Showdown”. In: *TACAS*. 2001, pages 1–22.

[73] Moshe Y. Vardi and Pierre Wolper. “An Automata-Theoretic Approach to Automatic Program Verification (Preliminary Report)”. In: *LICS*. 1986, pages 332–344.

[74] Moshe Y. Vardi and Pierre Wolper. “Reasoning About Infinite Computations”. In: *Inf. Comput.* 115.1 (1994), pages 1–37.

[75] Peter Cathcart Wason. “Reasoning”. In: *New Horizons in Psychology I*. Penguin, 1966.

[76] Hillel Wayne. *Consulting*. Accessed 2022-03-19. URL: https://www.hillelwayne.com/consulting.

[77] Kristin M. Weingartner and Amy M. Masnick. “Refutation texts: Implying the refutation of a scientific misconception can facilitate knowledge revision”. In: *Contemp. Edu. Psych.* 58 (2019), pages 138–148.

[78] Bang Wong. “Color blindness”. In: *Nature Methods* 8.6 (2011), pages 441–442.

[79] Tichakorn Wongpiromsarn, Alphan Ulusoy, Calin Belta, Emilio Frazzoli, and Daniela Rus. “Incremental temporal logic synthesis of control policies for robots interacting with dynamic agents”. In: *IROS*. IEEE, 2012, pages 229–236.

About the authors

Ben Greenman (benjamin.l.greenman@gmail.com)

Sam Saarinen (sam_saarinen@alumni.brown.edu)

Tim Nelson (timothy_nelson@brown.edu)

Shriram Krishnamurthi (shriram@brown.edu)