Misconceptions in Finite-Trace and Infinite-Trace Linear Temporal Logic

Ben Greenman[1,2]([✉]) [ID], Siddhartha Prasad[2] [ID], Antonio Di Stasio[3] [ID],
Shufang Zhu[3] [ID], Giuseppe De Giacomo[3] [ID], Shriram Krishnamurthi[2] [ID],
Marco Montali[4] [ID], Tim Nelson[2] [ID], and Milda Zizyte[2] [ID]

[1] University of Utah, Salt Lake City, USA
benjaminlgreenman@gmail.com
[2] Brown University, Providence, USA
[3] University of Oxford, Oxford, UK
[4] Free University of Bozen–Bolzano, Bolzano, Italy

Abstract. With the growing use of temporal logics in areas ranging from robot planning to runtime verification, it is critical that users have a clear understanding of what a specification means. Toward this end, we have been developing a catalog of semantic errors and a suite of test instruments targeting various user-groups. The catalog is of interest to educators, to logic designers, to formula authors, and to tool builders, e.g., to identify mistakes. The test instruments are suitable for classroom teaching or self-study.

This paper reports on five sets of survey data collected over a three-year span. We study misconceptions about finite-trace LTL_f in three LTL-aware audiences, and misconceptions about standard LTL in novices. We find several mistakes, even among experts. In addition, the data supports several categories of errors in both LTL_f and LTL that have not been identified in prior work. These findings, based on data from actual users, offer insights into what *specific ways* temporal logics are tricky and provide a groundwork for future interventions.

Keywords: LTL · LTLf · misconceptions · user studies

1 Introduction

Temporal logics are indispensable for specifying and verifying the behavior of complex systems. Linear temporal logic (LTL) and its restriction to finite traces (LTL_f) are two especially useful members of the family. LTL, for example, has been widely adopted by the robotics community [4,5,10,29,37,42,45,48,60,70]. LTL_f has applications to runtime verification [64], web-page testing [54], business process modeling [20,22], process mining [16], planning [13,24,25], reinforcement learning [21], and image processing [65]. Furthermore, both logics support good decision procedures [67] and enable program synthesis [2,3,7,11,49,56,62,71].

These successes all depend, however, on a crucial assumption: that users of the logics can actually write correct specifications. Given a well-formed but

Globally / Always

$$\sigma \ \models G(x) \iff \forall j, \qquad \sigma(j) \ \models x$$
$$\sigma_N \models G(x) \iff \forall j : \mathbf{j \le N}, \sigma_N(j) \models x$$

Finally / Eventually

$$\sigma \ \models F(x) \iff \exists j, \qquad \sigma(j) \ \models x$$
$$\sigma_N \models F(x) \iff \exists j : \mathbf{j \le N}, \sigma_N(j) \models x$$

Next

$$\sigma \ \models X(x) \iff \qquad\qquad\qquad \sigma(1) \ \models x$$
$$\sigma_N \models X(x) \iff \mathbf{1 \le N} \ \wedge \ \sigma_N(1) \models x$$

Until

$$\sigma \ \models x\,U\,y \iff \exists j, \qquad \sigma(j) \ \models y$$
$$\wedge \ \forall i : i < j, \sigma(i) \ \models x$$
$$\sigma_N \models x\,U\,y \iff \exists j : \mathbf{j \le N}, \sigma_N(j) \models y$$
$$\wedge \ \forall i : i < j, \sigma_N(i) \models x$$

Fig. 1. Semantics of four LTL and LTL$_f$ operators: G, F, X, U

incorrect formula, synthesis will output a system that behaves as specified—whether or not that is the desired behavior. It is therefore critical to know the *specific* misunderstandings that lead to incorrect formulas in order to correct them via tools, logic design, and teaching. That is the focus of this paper.

Contributions and Outline. After a brief introduction to LTL, LTL$_f$, and our pedagogy (Sect. 2), we proceed with the following contributions:

- We introduce two test instruments (Sect. 3):
  - a *finite trace* instrument that tests respondents' understanding of the delta between LTL and LTL$_f$, and
  - an *introductory* instrument that promotes active learning of LTL.
- We present a dataset of over 3,000 responses collected from dozens of respondents over the past three years (Sect. 4). The data contains mistakes from beginning, knowledgeable, and expert respondents (Sect. 6).
- We present a catalog of LTL and LTL$_f$ misconceptions (Sect. 5) that is thoroughly grounded in the data (Sect. 7).

The main results are in Sects. 6 and 7. The paper concludes with threats to validity (Sect. 8), related work (Sect. 9), and a brief discussion (Sect. 10).

2   Background

LTL formulas are interpreted over infinite traces, $\sigma = s_0 s_1 s_2 \cdots$, where each s_i is a state that provides valuations for a set of atomic propositions [55]. LTL$_f$ formulas are interpreted over finite traces, $\sigma_N = s_0 s_1 \cdots s_N$ [69]. While LTL and LTL$_f$ share the same syntax, their semantics differ as shown by the **highlighted constraints** in Fig. 1. This figure uses the notation $\sigma(j)$ to select a suffix of σ starting from position j. For example, $\sigma(2)$ is equal to $s_2 \cdots$. An *always* (G) operator quantifies over all remaining states in the trace, an *eventually* (F) must find a satisfying suffix before the trace ends, a *next* (X, aka *strong next*) constrains the suffix after the current state, and an *until* (U) must find a satisfying suffix for its right operand and ensure that its left operand holds beforehand. Not pictured is the LTL$_f$ weak next (X_W, omitted to save space), which does not require that a next state exists.

2.1 LTL$_f$ Example: Concision via Finiteness

Finite prefixes can be expressed within an infinite LTL trace, but doing so may require intricate formulas. To illustrate, consider a busy philosopher sitting in front of a bowl of ice cream. She has a lot of thinking to do, but if she decides to eat ice cream, she needs to do so before the ice cream melts. In LTL$_f$, traces are finite. The end of a trace might correspond, e.g., to the termination of a program or the end of a data stream. Ending the trace at the point where the ice cream melts allows for a simple framing of this property:

$G(w \implies F(e))$ # where w means "wants to eat" and e means "is eating"

By contrast, LTL requires a larger formula with a new variable (m: ice cream has melted) and a gadget to encode a prefix of an infinite trace.

$!m \land F(m) \land$ # ice cream eventually melts
$G(m \implies G(m)) \land$ # once melted, ice cream stays melted
$G(m \lor$ # either ice cream is melted, or
    $(w \implies F(e \land !m)))$ # philosopher who wants to eat eventually does

2.2 Toward a Concept Inventory

This paper is part of a larger effort to create a set of *concept inventory* test instruments for LTL, LTL$_f$, and related logics. Our guiding example is the Force Concept Inventory for teaching physics [39,40], a multiple choice test in which every incorrect choice is carefully designed to match *one* specific misconception. Unless test-takers select the wrong choice by mistake, their results strongly suggest which concepts they need to review. We are developing test instruments that use a variety of question types to identify the misconceptions that a temporal logic concept inventory should cover.

In a perfect world, every course subject would come with a concept inventory. However, developing an inventory takes several rounds of careful study (e.g., via think-aloud interviews) to identify misconceptions and reliably pinpoint them among test-takers [1,63]. One impediment to development is the expert blind spot [51,52]; namely, that test designers overlook concepts that learners struggle with. Our Spreading X misconception (Sect. 7.6), for example, is an issue that we were blind to.

This paper builds on prior LTL instruments [35,58] that employed a learner-driven tool called Quizius [59] to reduce the up-front cost of discovering misconceptions. Prior work [35] refined the instruments through three post-Quizius surveys, finding support for some potential misconceptions and discarding others. This paper represents a significant step forward in the iterative development of concept inventories with four additional studies that find misconceptions in LTL and in the unexplored domain of LTL$_f$.

3 Instrument Design

This section describes the design of our study instruments. Complete instruments are in the artifact for this paper [34]. We contribute two instruments: a *finite-trace* instrument that contrasts LTL$_f$ with LTL and an *introductory* instrument

Q. Describe the formula $G(X(\text{red}))$ for LTL and LTL$_f$.

LTL:

LTL$_f$:

(a) Describe Formulas

Q. Write a formula for Red is on exactly once in LTL and LTL$_f$.

LTL:

LTL$_f$:

(b) Write Formulas

Q. Is the formula red $\wedge\ G(X_W(\text{blue}))$ satisfied by this trace?

Answer: Yes   /   No

Rationale:

(c) Trace Matching

Q. Why does the formula $F(\text{red})$ reject this trace?

Answer:

(d) Explain Mismatches

Q. Is $G(!a)\ =!F(a)$ valid for any term a in LTL$_f$?
   This equation is valid in LTL.

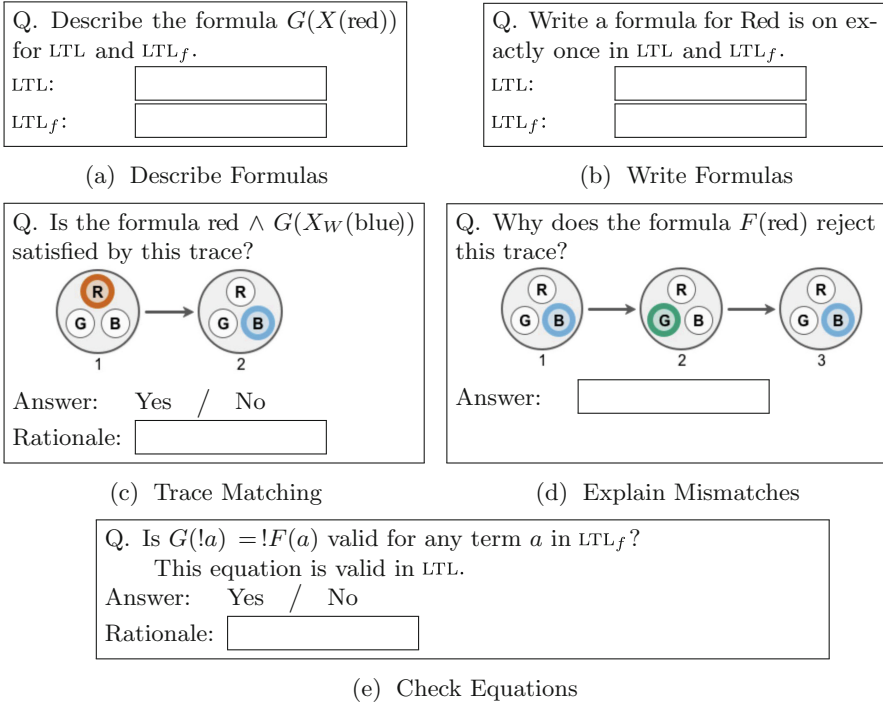Answer: Yes   /   No

Rationale:

(e) Check Equations

Fig. 2. Example questions

that assumes only minimal knowledge of LTL. The instruments are based on prior LTL work [35], reusing questions and question types that have proven effective in the past. The questions use simple state spaces with three on/off features such as the 3-color panel in Fig. 2.

The central question types ask about informal-to-formal translations:

Describe Formulas (Fig. 2a)**:** Given an LTL or LTL$_f$ formula, translate it to an English-language description. This task is similar to what a person does when reading a specification and deciding whether it is correct.

Write Formulas (Fig. 2b)**:** Given an English statement, translate it to LTL and/or LTL$_f$ or say that it is inexpressible. This is *the key skill* for doing formal verification. ("there must be a [informal-to-formal] transition" [26]).

Three other question types address specific goals. One type, Trace Matching, is from prior work [35]. The other two expose differences between LTL and LTL$_f$.

Trace Matching (Fig. 2c)**:** Given a formula and a trace, mark the trace as either satisfying or violating. These questions test for specific, semantic misunderstandings. All traces were either finite or repeated the final state.

Explain Mismatches (Fig. 2d)**:** Given an LTL$_f$ formula and a finite trace that violates the formula, explain the reason for the mismatch. The instructions

suggest four potential explanations: (1) only an infinite trace can satisfy the formula; (2) the trace is too long, i.e., the formula accepts no traces of this length; (3) the trace is too short; or (4) trace content mismatch, i.e., the wrong lights are on/off in some states. These questions serve as a tutorial on the mismatches that can arise in a finite-trace setting.

Check Equations (Fig. 2e)**:** Given an equation and a statement of its validity in LTL, determine whether it is valid in LTL$_f$ for non-empty traces. These questions test general ways in which LTL and LTL$_f$ formulas differ.

3.1   LTL$_f$ Instrument

The finite trace instrument is designed for an LTL-aware audience. This instrument has five parts, corresponding to the five question types above but arranged in order of difficulty rather than importance:

1. Explain Mismatches   2. Trace Matching   3. Describe Formulas
4. Write Formulas   5. Check Equations

Part 1 functions as an LTL$_f$ primer. It presents five mismatched formulas and traces and asks respondents to think critically about why the two disagree. For example, the trace in Fig. 2d is rejected by the formula $F(red)$ because it has no red states. Respondents who expect F to accept an empty trace (similar to weak next) may be able to use this example to correct their misconception.

Parts 2, 3, and 4 appear in order of increasing difficulty so that respondents can build confidence as they approach the harder questions. There are six Trace Matching questions, four Describe Formulas questions, and five Write Formulas questions. The translation questions each ask about LTL and LTL$_f$: respondents must provide two formulas (or two descriptions), or write "same" if the second would be identical. One question presents a formula that is insensitive to infiniteness [23], for which "same" is the correct response.

Part 5 presents three equations that are valid in LTL, such as $!X(a) = X(!a)$, and one that is invalid in LTL: $G(F(a)) = F(G(a))$. Respondents must decide whether the equations are valid in LTL$_f$.

3.2   LTL INSTRUMENTS

We used two instruments with students: a new *introductory* instrument, and the LTL instrument from prior work [35]. Both instruments have three parts:

1. Trace Matching   2. Describe Formulas   3. Write Formulas

Part 1 uses lasso traces where the last shown state repeats indefinitely. The state space is a locomotive with three features: engine smoke, a door, and a headlight. Parts 2 and 3 ask for translations to and from LTL.

$$(1)$$
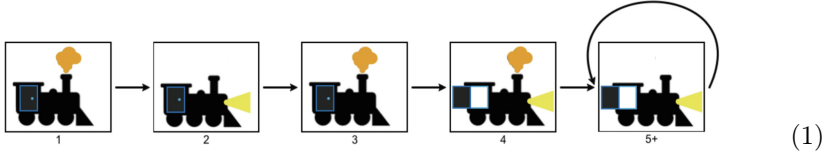
The first instrument is intended for students who have no knowledge of temporal logic. It presents nine of the easy-to-answer Trace Matching questions, and only two Describe questions and two Write questions. Some of the trace questions match the same formula with different traces to hone in on misconceptions. The translation questions intentionally do not ask about the until operator.

The second instrument is from prior work [35] with minor enhancements. It asks nine Match questions, five Describe questions, and five Write questions.

Table 1. Study contexts, number of respondents, and number of responses

Context	Instrument	Respondents	Total Responses
α'23	finite-trace	22	1132
α'24	finite-trace	18	693
FTAI	finite-trace	24	455
β_1	introductory	31	403
β_2	LTL [35]	24	456

4   Data

We deployed our instruments to four populations over three years. The finite-trace instrument went out to two semesters of students at a public UK university (α'23, α'24) and to the attendees of a symposium on LTL_f in artificial intelligence (FTAI—anonymized acronym). The introductory instrument was used in an embedded systems course at a private US university (β_1, β_2). Between 18 and 24 respondents completed each instrument, and each participant contributed dozens of individual responses to the overall dataset. Table 1 provides the details. We hosted each instrument on Qualtrics.

4.1   Student α: 2023 and 2024

Populations α'23 and α'24 consisted of students enrolled in an elective course on self-programming agents, which is dedicated to various forms of LTL_f reactive synthesis and planning in the context of autonomous agents. Students can take this course in the final year of a BSc in computer science or during an MSc on Advanced CS. Both α populations are similar and received comparable instruction, though we remark that the instructor joined the university in 2023. Early in

the term, students received a lecture on LTL and completed the LTL instrument from prior work as a homework exercise. Shortly afterward, students received a lecture on LTL_f and completed the finite-trace instrument as homework. The LTL responses were of very high quality (92% correct in α'23), so we analyze only the LTL_f responses in this paper.

The α'23 instrument differs from the final, α'24 instrument in two ways: the Explain Mismatches questions are multiple choice and there are three additional Check Equations questions (which did not lead to interesting incorrect responses). Free response is better for Explain Mismatches because it is less constraining. Respondents struggled when two choices might reasonably apply, and forcing them to choose was not helpful in our search for misconceptions.

4.2   FTAI: 2023

FTAI is our anonymized name for a symposium on finite-trace temporal logics for AI that was held in 2023. The event brought together world-class researchers with deep expertise in temporal logics including LTL_f. Seventeen attendees (74%) self-reported AI as among their primary research areas, nine (39%) selected formal methods, and five (21%) selected machine learning. Eleven claimed to be knowledgeable in LTL_f specifically. All but a few attendees were in-person.

On the first day of the symposium, we presented (via Zoom) a brief introduction to our work on logic misconceptions and gave respondents 15 min to fill out the instrument. This introduction did not explain LTL_f semantics and it did not explain our question types; all instructions were in the instrument itself. Ten respondents completed the instrument in the allotted time. Eight respondents finished by the end of the conference. Six others finished later in Spring 2023; these may have been colleagues of symposium attendees, as we encouraged attendees to share the instrument link with their research groups.

Respondents in this study received only a subset of the α'23 instrument to maximize the completion rate, which explains the relatively low number of responses in Table 1. They completed 3 out of 5 Explain Mismatches questions, 3 of 6 Trace Matching questions, 2 of 4 Describe Formulas questions, 2 of 5 Write Formulas questions, and 5 of 7 Check Equations questions—all selected uniformly at random by Qualtrics.

4.3   Student β: 2022

Population β completed two instruments, β_1 and β_2, in the context of an elective undergraduate course on embedded systems taught at a private US university. The course has limited time to cover LTL-based model checking, making it critical to teach LTL quickly to students unfamiliar with temporal logic. In 2022, near the end of the semester, we assigned the introductory instrument as homework (β_1) without teaching LTL in lecture. Students had several days to read the course textbook [47] and submit. The next lecture featured LTL and assigned the full LTL instrument [35] as homework due the following week (β_2).

All homework in embedded systems was graded by participation. Furthermore, students were allowed to drop three homeworks during the term. We know from survey comments that at least two students were planning to drop an LTL homework, but since responses are anonymous and these comments appeared only in complete surveys, there is no reliable way to determine which of these students, if any, actually dropped an LTL homework.

5   Catalog Design

The catalog, or "code book" (in the qualitative analysis sense), is our rubric for temporal logic misconceptions. Figure 3 presents a short overview of the core semantic errors. Its aim is to provide just enough background for readers to understand our results in Sects. 6 and 7. The full catalog in our artifact comes with instructions showing how to apply the labels to new responses [34].

Length : too many/few states (LTL$_f$ only)
Last : attempt $X_W(\mathit{false})$ in LTL
Bad Prop : wrong $\land/\lor/\Rightarrow$/atom
Bad State Index : too many/few X
Bad State Quantification : wrong $F/G/U$
Cycle G : underspecified G term
Implicit F : missing/assumed F

Implicit G : missing/assumed G
Implicit Prefix : F missing context
Other Implicit : other underconstraint
Weak U : mistake U for W
Exclusive U : expect disjoint U terms
Trace-Split U : overspecified U
Spreading X : mistake XX for $X \land XX$

Fig. 3. Brief summary of misconceptions

In addition to the labels in Fig. 3, there are three meta labels: Precedence, RV, and Unlabeled. Precedence applies to responses that are ambiguous due to missing parentheses. RV stands for "Reasonable Variant," and applies to written formulas that support an unintended reading of an English prompt. Unlabeled is for responses that contain several mistakes or otherwise defy categorization.

The **highlighted** labels are new to this work. **Length** and **Last** apply only to LTL$_f$. **Cycle G**, **Implicit Prefix**, **Trace-Split U**, and **Spreading X** apply to both LTL$_f$ and LTL. The other labels originate in prior work [35]. We developed the new labels by starting from the prior catalog and applying techniques from grounded theory [33] to discover categories of mistakes. Two authors worked as labelers. First, the labelers independently assessed sample responses using the baseline catalog. Coding happened in small sessions to minimize labeler fatigue. Second, the labelers met to identify patterns among responses that did not fit the current rubric. Third, the labelers used the standard Cohen's κ score [17] to check agreement. This measure typically ranges from 0 to 1, where a score above 0.8 is considered excellent [61]. The coders quickly reached a high score, perhaps due to the well-tested baseline catalog. Further details on instrument development follow:

Finite Trace: $\kappa = 0.79$ after labeling 26 responses: 14 Write Formulas, 8 Describe Formulas, and 6 Check Equations.

Introductory: $\kappa = 0.83$ after labeling 13 responses: 9 Write Formulas and 4 Describe Formulas.

6   Results: Incorrect Responses, Specific Errors

Our instruments collected a variety of errors across the four populations. Table 2 presents the totals at a high-level. Table rows correspond to question types (with abbreviated names, such as Explain for Explain Mismatches), and table columns name the instrument deployments. Each cell counts the number of incorrect responses (*not* the number of respondents who contributed these responses) and reports it as a percentage of the total responses for that particular instrument and question type. Be advised that percentages are not comparable across columns because the number of questions in each part may have changed; for example, Check Equations has 7 questions in α'23 and 4 in α'24.

Table 2. Total incorrect responses

	α'23	α'24	FTAI	β_1	β_2
Explain	18 (20.00%)	4 (4.44%)	16 (22.22%)	N/A	N/A
Match	2 (1.85%)	3 (2.78%)	6 (8.33%)	76 (27.24%)	43 (19.91%)
Describe	23 (15.97%)	23 (15.97%)	19 (19.79%)	30 (48.39%)	47 (39.17%)
Write	38 (21.11%)	45 (25.00%)	32 (33.68%)	41 (66.13%)	76 (63.33%)
Check	9 (7.14%)	5 (6.94%)	25 (20.83%)	N/A	N/A

The main takeaway from Table 2 is that every question type attracted some incorrect responses, and some attracted quite a few (over 20%). Trace Matching was the easiest question across the board and Write Formulas was the hardest; even the FTAI respondents submitted a fair number of incorrect formulas. Students in β_1 submitted many incorrect responses. At a glance, it would seem that the β_2 responses are only marginally better percentage-wise, but there were nearly twice as many translation questions in the β_2 instrument and they were more difficult; the small percentage improvement is encouraging.

Each incorrect response may correspond to zero or more misconceptions in our catalog, depending on why it is incorrect. Table 3 presents the catalog classification of the incorrect responses. The columns are grouped by three question types: Trace Matching, Describe Formulas, and Write Formulas. We discuss the other question types in prose below. Within each question type, columns correspond to deployments. The rows are labels from the catalog. Each cell counts the number of incorrect responses; we use a dash (-) rather than a zero to make the nonzero numbers easier to see.

Table 3. Errors in incorrect responses (one response may match several labels)

(a) Finite trace instrument

Code	Match α'23	Match α'24	Match FTAI	Describe α'23	Describe α'24	Describe FTAI	Write α'23	Write α'24	Write FTAI	Total
Length	-	2	3	7	2	2	10	3	3	31
Last	-	-	-	-	-	-	1	5	-	6
Bad Prop	-	-	-	-	2	4	-	6	2	14
Bad State Index	-	-	-	-	-	-	2	9	5	17
Bad State Quantification	-	-	-	-	-	-	-	1	2	3
Cycle G	-	-	-	-	-	-	-	2	2	4
Implicit F	-	-	-	7	6	5	3	7	2	30
Implicit G	-	-	-	1	-	-	4	8	2	15
Implicit Prefix	-	-	-	-	-	-	8	4	8	20
Other Implicit	-	-	-	-	-	3	1	-	-	4
Weak U	2	1	2	-	1	1	-	-	-	7
Exclusive U	-	-	1	-	3	2	-	-	-	6
Trace-Split U	-	-	-	-	-	-	-	-	3	3
Spreading X	-	-	-	-	-	-	-	-	-	-
Precedence	-	-	-	-	-	-	-	1	1	2
RV	-	-	-	-	-	-	2	-	-	2
Unlabeled	-	-	-	8	10	4	13	2	9	46

(b) Introductory and LTL [35] instruments

Code	Match β_1	Match β_2	Describe β_1	Describe β_2	Write β_1	Write β_2	Total
Bad Prop	9	7	3	8	16	14	57
Bad State Index	1	8	15	7	3	10	44
Bad State Quantification	7	3	5	9	4	4	32
Cycle G	-	-	-	-	-	-	-
Implicit F	11	11	1	1	-	4	28
Implicit G	13	1	1	7	23	23	68
Implicit Prefix	-	-	-	-	-	8	8
Other Implicit	-	-	-	-	-	5	5
Weak U	15	9	-	2	-	-	26
Exclusive U	8	5	-	4	-	-	17
Trace-Split U	-	-	-	-	-	2	2
Spreading X	6	-	1	3	10	3	23
Precedence	2	-	-	4	-	3	9
RV	-	-	-	-	-	-	-
Unlabeled	6	-	2	16	7	19	50

Every core label has at least some support from the responses, with Bad State Index, Implicit F, and Implicit G being among the most popular. The Weak U label has low numbers, but these came primarily from a Trace Matching question that specifically tests this issue; the fact that even one FTAI participant made this mistake is noteworthy. Issues with trace length constraints (Length) are common in LTL$_f$; see Sect. 7 for examples. Lastly, the low numbers for generic labels (Bad State Quantification and Other Implicit) and for reasonable variants (RV) suggest that the revised catalog is better at pinpointing issues and that the revised instruments are clearer to respondents.

We report some negative findings as well. Two labels, Cycle G and Trace-Split U, have little support overall and warrant targeted testing in the future. Unlabeled is unfortunately common, which suggests a need for interviews to learn the reasoning behind any deeply-incorrect responses. Some unlabeled responses in Table 3b do, however, have explanations. These are from respondents who were confused about LTL syntax, or who did not attempt the question.

Remaining Question Formats. The finite trace instruments include two question types that are not in Table 3a: Explain Mismatches and Check Equations. The incorrect Explain Mismatches responses are all Unlabeled; most of these are due to the multiple-choice ambiguity noted in Sect. 3.1. The incorrect Check Equations responses cannot be labeled definitively because these questions did not ask respondents to explain their reasoning (Fig. 2e). We merely note that the data suggests issues with Length, OtherImplicit, and a weak notion of F. The weak-F responses incorrectly marked $F(a) = a \lor X(F(a))$ as invalid in LTL$_f$.

7   Results: Categories of Errors

We turn now to the actual survey responses that support the new categories of errors; namely, the two LTL$_f$ labels and four additional LTL labels. To ground the discussion, the subsections below present actual instrument questions ("Q") and representative sample responses ("WA" for "wrong answer"). We also discuss how tools might use our findings to provide feedback.

Certain questions appeared only in the finite-trace instruments and vice-versa. These are noted below. Also, to streamline the presentation, we have translated the introductory-instrument responses to use colors instead of loco-motive characteristics (compare Fig. 2 and Eq. (1)).

7.1   Length (LTL$_f$ only)

The Length label applies to responses that require too many or too few states. When writing an LTL$_f$ formula, this issue can arise from the use of strong next instead of weak next. Tools might help by reporting the trace length(s) that a formula accepts.

– **Q.** Describe the LTL$_f$ formula $red \land !X(blue)$.

– **WA.** "The first state must be red and the second state must not be blue."
This answer implies that a second state must exist, but the formula does not.
There are four responses of this sort in the dataset: two in α'23, one in α'24,
and one in FTAI.

– **Q.** Describe the LTL$_f$ formula $G(red \Rightarrow X(!red \wedge X(red)))$.
– **WA.** "For every state, if there is a red light on, the next state is with the
red light off, and the state afterward is with the red light on. The trace must
have at least have 3 states."
No finite trace with a red light can satisfy this formula, as every red light
demands another two states later. There are seven responses of this sort: five
in α'23 and one each in α'24 and FTAI.

– **Q.** Write an LTL$_f$ formula for: *Blue is on in the first state, off in the second
state, and alternates on/off for the remaining states.*
– **WA.** $blue \wedge G(blue \Rightarrow X_W(!blue \wedge X_W(blue)))$
The prompt requires at least two states, but the formula accepts traces with
only one blue state. Interestingly, this formula is correct in LTL using X
instead of X_W, which underscores the subtlety of LTL$_f$. Eight α'23, one α'24,
and zero FTAI responses made this error.

7.2   Last (LTL$_f$ only)

The Last label applies to responses that attempt to encode a final state in infinite-
trace LTL instead of saying that the prompt is inexpressible. All such responses
stem from one formula-writing question.

– **Q.** Write (if possible) an LTL formula for: *Green is on in the final state.*
– **WA.** $F(G(green))$
While this response is correct for LTL$_f$ and is syntactically-valid LTL, it is
trying to answer an impossible question. There are six responses of this sort:
one from α'23, five from α'24, and zero from FTAI.

7.3   Cycle G

In LTL and LTL$_f$, the G operator imposes a constraint on every state. Yet, some
responses expect G to constrain one state, skip a few states, and reapply later.
The skipped states are precisely those captured by occurrences of X within the
G operand. A tool might help by highlighting atom constraints at each time
index (in the following example, index 2 would show a contradiction).

– **Q.** Write an LTL formula for: *Blue is on in the first state, off in the second
state, and alternates on/off for the remaining states.*
– **WA.** $G(blue \wedge X(!blue))$
This formula is unsatisfiable because it requires blue to be both on and off
in the second state. There are four responses of this sort, two from α'24 and
two from FTAI. However, we must caution that these responses came from
only two people who made the mistake consistently in LTL and LTL$_f$.

7.4   Implicit Prefix

The baseline catalog contains a generic label Other Implicit for responses that accept too many traces but do not fall under a more precise category. One such response from FTAI describes $G(red \Rightarrow X(!red \wedge X(red)))$ as "whenever red holds, it also holds two steps later," leaving the middle state underconstrained.

The Implicit Prefix label narrows the scope of Other Implicit. It applies to responses that correctly describe the suffix of valid traces but leave the prefix underconstrained. It does not apply to the example in the previous paragraph. Tools might help by showing example traces; for instance, traces with early states that satisfy some but not all constraints under an F may be informative.

– **Q.** Write an LTL formula for: *Red is on exactly once*.
– **WA.** $F(red \wedge X(G(!red)))$
   This formula describes a suffix in which red is on at one state and turns off afterward, but it does not prevent red from turning on before this point. There are 24 responses of this sort: eight each from α'23 and FTAI, and four each from α'24 and β_2. The finite-trace respondents made this mistake consistently in LTL and LTL$_f$, so the total in terms of people is only 14.

– **Q.** Write an LTL formula for: *Green is on for zero or more states, then turns off and remains off in the future*.
– **WA.** $G(F(!green))$
   Whereas the specification asks for green to stay on until it turns off, the formula allows green to turn on and off before reaching a non-green suffix. There are four responses of this sort in β_2. This question is not in the finite-trace instruments because it does not contrast LTL and LTL$_f$.

7.5   Trace-Split U

Several responses use F and G in the left operand of an until, as in $G(red)\ U\ blue$. These responses are usually incorrect. Some of them would be correct, however, if the left and right operands were interpreted on different parts of the full trace: a prefix on the left and a suffix on the right. (Interpreting on a prefix makes no sense in LTL, but is sensible in LTL$_f$.) The Trace-Split U label captures these responses. Tools can help by reporting such nested operands as a U antipattern.

– **Q.** Write an LTL formula for: *Blue is on in at least two states*.
– **WA.** $F(blue)\ U\ F(blue)$
   Any trace with one blue state satisfies the formula. There are two responses of this sort from FTAI and zero elsewhere.

– **Q.** Write an LTL formula for: *Green is on for zero or more states, then turns off and remains off in the future*.
– **WA.** $G(green)\ U\ G(!green)$
   Although a natural-language reading of this formula sounds compelling

(*always green until always not green*), the left G would entail a green light in every state. There are two responses of this sort in β_2. This question is not in the finite-trace instruments because it does not contrast LTL and LTL$_f$.

7.6 Spreading X

The X operator targets one specific state whereas G, F, and U quantify over an unknown future. This difference is evidently confusing to beginners, as several of the β_1 and β_2 responses expect one X to constrain both the current state and the next state. With nesting, these responses expect a longer interval, e.g., three red states for $X(X(red))$. Prior work with novices observed this issue as well [58]. We did not find evidence for it in our earlier studies [35], so perhaps the misconception is easily corrected. Tools can help by reminding users that an n-fold composition of X constrains one state n steps ahead.

– **Q.** Describe the LTL formula *blue* $\Rightarrow X(X(X(blue)))$.
– **WA.** "When the blue light is on, it will stay on for the next 3 states."
  There are three such responses. This question is only in the β_2 instrument.

– **Q.** Write an LTL formula for: *Red cannot stay on for 3 states in a row*.
– **WA.** $G(!X(X(X(red))))$
  There are eight such responses in β_1, and three in β_2. The finite-trace instrument does not include this question.

8 Threats to Validity

Qualitative coding inherently comes with biases, and our high agreement scores do not prove that these have been excised. To mitigate this issue, our data is available for other researchers to audit. Another threat is that the sets over which we computed agreement are not large.

One author manually classified responses for correctness and may have mislabeled some, despite our auditing. Write Formulas responses in particular might have leveraged automation, but the survey did not enforce an LTL syntax in order to lower the burden on respondents. Thus, there are variations such as `or` versus `|` and `engine` versus `E` that we had to normalize manually. One response uses `next` (perhaps inspired by PSL weak next [28]) without specifying a strong or weak interpretation. This ambiguity is a threat; fortunately, the response in question is incorrect in the same way with X or X_W. Operator precedence is another avenue for miscommunication; we assume, e.g., weak precedence for implication, but respondents may have had a different meaning in mind.

Regarding external validity, the two α studies took place at the same institution with the same instructor. The β study used a different institution and student population, and although the results are comparable to α they may not carry over to other populations, such as learners in industry. FTAI respondents were under time pressure due to the conference, and may have rushed through the more difficult translation questions.

Two question types require fluency in English. Although we did not specifically check for fluency, our respondents seem to meet this bar. Both universities that we worked with conduct all classes in English and expect a high degree of fluency. The FTAI symposium used English as well for all papers and talks. There were no indications of severe language issues in the responses.

Our instruments are rather weak ecologically because they ask basic questions about a rudimentary state space. Practical uses of LTL would involve systems with interacting components, and users would have access to verification tools. Performing studies in a realistic setting is an important topic for future work.

9   Related Work

Design tools [15,57], alternative languages and logics [6,8,28,46,66], pattern languages [27,36,43,50,57], natural-language translators [12,18,30] and error checkers [9,14,41,44,54], all seek to improve the usability of temporal logics such as LTL. Yet, none of these works study the misunderstandings of humans; at best, they address mistakes that a person *might* make.

Prior work on the Declare modeling language used think-aloud interviews to discover and validate errors [38]. Our work can help separate general LTL issues from Declare-specific issues. Other related user studies include two comparisons of LTL to similar logics [15,19], and an interface design study [15]. While these studies target learners, the focus is not directly on logic misconceptions.

Our translation questions are similar to those from Iltis [31,32], a tool for teaching logic. Iltis might serve as a framework for future studies, though it is aimed toward pedagogy rather than studies of misconceptions.

With the introductory instruments, we considered providing a link to Wickström's LTL visualizer [54,68]. We did not, due to concerns that misconceptions about the tool, which has not been validated, would be a confounding factor.

10   Looking Forward

We conducted a first study of LTL$_f$ misconceptions in three populations with well-informed respondents, and studied LTL in two rounds with novices. The data offers insights into mis-specifications with two categories of LTL$_f$-specific mistakes, four new categories of LTL mistakes, and refined support for categories from prior work [35]. Given the very simple scenarios and formulas that we used, we suspect that many more issues lurk in more complicated settings.

Our work has obvious implications for learners and educators. We have already begun to employ its insights to create a new interactive learning environment called the LTL Tutor: https://www.ltl-tutor.xyz/. We have also had positive experiences in an undergraduate course on logical modeling [53] and in a graduate course on software verification. The instruments work well as an in-class activity followed by group discussion.

This work can also impact the design of future logics. Narrowly, it suggests different operator designs; broadly, it provides a methodology to identify misconceptions in the first place.

Finally, this work also has implications for tools that consume LTL or LTL$_f$. Currently, tools assume that a logical utterance precisely captures the user's intent, and verify, synthesize, or otherwise manifest exactly what was written. Our work can (and should!) be used to check for the presence of predictable errors, e.g., by checking that users really meant what they wrote (especially if they fall within a misconception category).

Data Availability Statement. The survey instruments, final catalog, and labeled responses are available in the artifact for this paper [34].

Disclosure of Interests. The authors have no competing interests to declare.

References

1. Almstrum, V.L., et al.: Concept inventories in computer science for the topic discrete mathematics. ACM SIGCSE Bull. **38**(4), 132–145 (2006). https://doi.org/10.1145/1189136.1189182

2. Alur, R., Bansal, S., Bastani, O., Jothimurugan, K.: A framework for transforming specifications in reinforcement learning. CoRR abs/2111.00272 (2021). https://arxiv.org/abs/2111.00272

3. Amram, G., Bansal, S., Fried, D., Tabajara, L.M., Vardi, M.Y., Weiss, G.: Adapting behaviors via reactive synthesis. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12759, pp. 870–893. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_41

4. Antoniotti, M., Mishra, B.: Discrete events models + temporal logic = supervisory controller: automatic synthesis of locomotion controllers. In: ICRA, pp. 1441–1446. IEEE (1995). https://doi.org/10.1109/ROBOT.1995.525480

5. Araki, B., Li, X., Vodrahalli, K., DeCastro, J.A., Fry, M.J., Rus, D.: The logical options framework. In: ICML, vol. 139, pp. 307–317. PMLR (2021). http://proceedings.mlr.press/v139/araki21a.html

6. Armoni, R., et al.: The ForSpec temporal logic: a new temporal property-specification language. In: Katoen, J.-P., Stevens, P. (eds.) TACAS 2002. LNCS, vol. 2280, pp. 296–311. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46002-0_21

7. Bansal, S., Li, Y., Tabajara, L.M., Vardi, M.Y., Wells, A.: Model checking strategies from synthesis over finite traces. In: André, É., Sun, J. (eds.) ATVA 2023. LNCS,

vol. 14215, pp. 227–247. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-45329-8_11

8. Beer, I., Ben-David, S., Eisner, C., Fisman, D., Gringauze, A., Rodeh, Y.: The temporal logic sugar. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 363–367. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44585-4_33

9. Beer, I., Ben-David, S., Eisner, C., Rodeh, Y.: Efficient detection of vacuity in ACTL formulas. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 279–290. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63166-6_28

10. Bhatia, A., Kavraki, L.E., Vardi, M.Y.: Sampling-based motion planning with temporal goals. In: ICRA, pp. 2689–2696. IEEE (2010). https://doi.org/10.1109/ROBOT.2010.5509503

11. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. J. Comput. Syst. Sci. **78**(3), 911–938 (2012). https://doi.org/10.1016/j.jcss.2011.08.007

12. Brunello, A., Montanari, A., Reynolds, M.: Synthesis of LTL formulas from natural language texts: state of the art and research directions. In: TIME, vol. 147, pp. 17:1–17:19. Schloss Dagstuhl (2019). https://doi.org/10.4230/LIPIcs.TIME.2019.17

13. Camacho, A., McIlraith, S.A.: Strong fully observable non-deterministic planning with LTL and LTLf goals. In: IJCAI, pp. 5523–5531. ijcai.org (2019). https://doi.org/10.24963/IJCAI.2019/767

14. Chockler, H., Strichman, O.: Easier and more informative vacuity checks. In: MEMOCODE, pp. 189–198. IEEE Computer Society (2007). https://doi.org/10.1109/MEMCOD.2007.371225

15. Choi, W., Vazirani, M., Santolucito, M.: Program synthesis for musicians: a usability testbed for temporal logic specifications. In: Oh, H. (ed.) APLAS 2021. LNCS, vol. 13008, pp. 47–61. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-89051-3_4

16. Ciccio, C.D., Montali, M.: Declarative process specifications: reasoning, discovery, monitoring. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. LNBIP, vol. 448, pp. 108–152. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08848-3_4

17. Cohen, J.: A coefficient of agreement for nominal scales. Educ. Psychol. Measur. **20**(1), 37–46 (1960). https://doi.org/10.1177/001316446002000104

18. Cosler, M., Hahn, C., Mendoza, D., Schmitt, F., Trippel, C.: nl2spec: Interactively translating unstructured natural language to temporal logics with large language models. In: Enea, C., Lal, A. (eds.) CAV 2023. LNCS, vol. 13965, pp. 383–396. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-37703-7_18

19. Czepa, C., Zdun, U.: On the understandability of temporal properties formalized in linear temporal logic, property specification patterns and event processing language. IEEE Trans. Softw. Eng. **46**(1), 100–112 (2020). https://doi.org/10.1109/TSE.2018.2859926

20. De Giacomo, G., De Masellis, R., Grasso, M., Maggi, F.M., Montali, M.: Monitoring business metaconstraints based on LTL and LDL for finite traces. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 1–17. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10172-9_1

21. De Giacomo, G., Iocchi, L., Favorito, M., Patrizi, F.: Restraining bolts for reinforcement learning agents. In: AAAI, pp. 13659–13662. AAAI Press (2020).https://doi.org/10.1609/AAAI.V34I09.7114

22. De Giacomo, G., Maggi, F.M., Marrella, A., Patrizi, F.: On the disruptive effectiveness of automated planning for LTLf-based trace alignment. In: Artificial Intelligence, pp. 1–7. AAAI (2017). https://doi.org/10.1609/aaai.v31i1.11020

23. De Giacomo, G., Masellis, R.D., Montali, M.: Reasoning on LTL on finite traces: insensitivity to infiniteness. In: AAAI, pp. 1027–1033. AAAI Press (2014). https://doi.org/10.1609/AAAI.V28I1.8872

24. De Giacomo, G., Rubin, S.: Automata-theoretic foundations of FOND planning for LTLf and LDLf goals. In: IJCAI, pp. 4729–4735. ijcai.org (2018). https://doi.org/10.24963/IJCAI.2018/657

25. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI, pp. 854–860. AAAI Press (2013). https://doi.org/10.5555/2540128.2540252

26. DeMillo, R.A., Lipton, R.J., Perlis, A.J.: Social processes and proofs of theorems and programs. CACM **22**(5), 271–280 (1979). https://doi.org/10.1145/359104.359106

27. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: ICSE, pp. 411–420. ACM (1999). https://doi.org/10.1145/302405.302672

28. Eisner, C., Fisman, D.: A Practical Introduction to PSL. Springer, New York (2006). https://doi.org/10.1007/978-0-387-36123-9

29. Fainekos, G.E., Kress-Gazit, H., Pappas, G.J.: Temporal logic motion planning for mobile robots. In: ICRA, pp. 2020–2025. IEEE (2005). https://doi.org/10.1109/ROBOT.2005.1570410

30. Fuggitti, F., Chakraborti, T.: NL2LTL – a Python package for converting natural language (NL) instructions to linear temporal logic (LTL) formulas. In: AAAI Conference on Artificial Intelligence, vol. 37, no. 13, pp. 16428–16430 (2023). https://doi.org/10.1609/aaai.v37i13.27068

31. Geck, G., Ljulin, A., Peter, S., Schmidt, J., Vehlken, F., Zeume, T.: Introduction to Iltis: an interactive, web-based system for teaching logic. In: ITiCSE, pp. 141–146. ACM (2018). https://doi.org/10.1145/3197091.3197095

32. Geck, G., et al.: Iltis: teaching logic in the Web. CoRR abs/2105.05763 (2021)

33. Glaser, B., Strauss, A.: The Discovery of Grounded Theory: Strategies for Qualitative Research. Sociology Press, Mill Valley (1967)

34. Greenman, B., et al.: Artifact for misconceptions in finite-trace and infinite-trace linear temporal logic (2024). https://doi.org/10.5281/zenodo.12770102

35. Greenman, B., Saarinen, S., Nelson, T., Krishnamurthi, S.: Little tricky logic: misconceptions in the understanding of LTL. Programming **7**(2), 7:1–7:37 (2023). https://doi.org/10.22152/programming-journal.org/2023/7/7

36. Grunske, L.: Specification patterns for probabilistic quality properties. In: ICSE. ACM (2008). https://doi.org/10.1145/1368088.1368094

37. Gundana, D., Kress-Gazit, H.: Event-based signal temporal logic synthesis for single and multi-robot tasks. IEEE Robot. Autom. Lett. **6**(2), 3687–3694 (2021). https://doi.org/10.1109/LRA.2021.3064220

38. Haisjackl, C., et al.: Understanding Declare models: strategies, pitfalls, empirical results. Softw. Syst. Model. **15**(2), 325–352 (2016). https://doi.org/10.1007/S10270-014-0435-Z

39. Hestenes, D.: Toward a modeling theory of physics instruction. Am. J. Phys. **55**(5), 440–454 (1987). https://doi.org/10.1119/1.15129

40. Hestenes, D., Wells, M., Swackhamer, G.: Force concept inventory. Phys. Teach. **30**(3), 141–158 (1992). https://doi.org/10.1119/1.2343497

41. Hoskote, Y.V., Kam, T., Ho, P., Zhao, X.: Coverage estimation for symbolic model checking. In: Design Automation Conference, pp. 300–305. ACM (1999). https://doi.org/10.1145/309847.309936

42. Kantaros, Y., Zavlanos, M.M.: STyLuS*: a temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. Int. J. Robot. Res. **39**(7), 812–836 (2020). https://doi.org/10.1177/0278364920913922

43. Konrad, S., Cheng, B.H.C.: Real-time specification patterns. In: ICSE, p. 372–381. ACM (2005). https://doi.org/10.1145/1062455.1062526

44. Kupferman, O., Vardi, M.Y.: Vacuity detection in temporal model checking. Int. J. Softw. Tools Technol. Transf. **4**(2), 224–233 (2003). https://doi.org/10.1007/s100090100062

45. Lahijanian, M., Almagor, S., Fried, D., Kavraki, L., Vardi, M.: This time the robot settles for a cost: a quantitative approach to temporal logic planning with partial satisfaction. In: AAAI, pp. 3664–3671. AAAI Press (2015). https://shaull.github.io/pub/LAFKV15.pdf

46. Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, Boston (2002)

47. Lee, E.A., Seshia, S.A.: Introduction to Embedded Systems—A Cyber–Physical Systems Approach, 2nd edn. MIT Press, Cambridge (2017)

48. Loizou, S.G., Kyriakopoulos, K.J.: Automatic synthesis of multi-agent motion tasks based on LTL specifications. In: CDC, pp. 153–158. IEEE (2004). https://doi.org/10.1109/CDC.2004.1428622

49. Manna, Z., Wolper, P.: Synthesis of communicating processes from temporal logic specifications. TOPLAS **6**(1), 68–93 (1984). https://doi.org/10.1145/357233.357237

50. Menghi, C., Tsigkanos, C., Pelliccione, P., Ghezzi, C., Berger, T.: Specification patterns for robotic missions. IEEE Trans. Softw. Eng. **47**(10), 2208–2224 (2021). https://doi.org/10.1109/TSE.2019.2945329

51. Nathan, M.J., Koedinger, K.R., Alibali, M.W.: Expert blind spot: when content knowledge eclipses pedagogical content knowledge. In: International Conference on Cognitive Sciences, pp. 644–648 (2001). http://pact.cs.cmu.edu/koedinger/pubs/2001_NathanEtAl_ICCS_EBS.pdf

52. Nathan, M.J., Petrosino, A.: Expert blind spot among preservice teachers. Am. Educ. Res. J. **40**(4), 905–928 (2003). https://www.jstor.org/stable/3699412

53. Nelson, T., et al.: Forge: a tool and language for teaching formal methods. PACMPL **8**(OOPSLA1), 1–31 (2024). https://doi.org/10.1145/3649833

54. O'Connor, L., Wickström, O.: Quickstrom: property-based acceptance testing with LTL specifications. In: PLDI, pp. 1025–1038. ACM (2022). https://doi.org/10.1145/3519939.3523728

55. Pnueli, A.: The temporal logic of programs. In: FOCS, pp. 46–57. IEEE (1977). https://doi.org/10.1109/SFCS.1977.32

56. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: POPL, pp. 179–190. ACM (1989). https://doi.org/10.1145/75277.75293

57. Rajhans, A., Mavrommati, A., Mosterman, P.J., Valenti, R.G.: Specification and runtime verification of temporal assessments in simulink. In: Feng, L., Fisman, D. (eds.) RV 2021. LNCS, vol. 12974, pp. 288–296. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88494-9_17

58. Saarinen, S.: Query strategies for directed graphical models and their application to adaptive testing. Ph.D. thesis, Brown University (2021). https://repository.library.brown.edu/studio/item/bdr:kgyft3b4/

59. Saarinen, S., Krishnamurthi, S., Fisler, K., Tunnell Wilson, P.: Harnessing the wisdom of the classes: classsourcing and machine learning for assessment instrument generation. In: SIGCSE, pp. 606–612. ACM (2019). https://doi.org/10.1145/3287324.3287504

60. Shah, A., Kamath, P., Shah, J.A., Li, S.: Bayesian inference of temporal task specifications from demonstrations. In: NeurIPS, pp. 3808–3817 (2018). https://proceedings.neurips.cc/paper/2018/hash/13168e6a2e6c84b4b7de9390c0ef5ec5-Abstract.html

61. Sim, J., Wright, C.C.: The kappa statistic in reliability studies: use, interpretation, and sample size requirements. Phys. Ther. **85**(3), 257–268 (2005). https://doi.org/10.1093/ptj/85.3.257

62. Tabajara, L.M., Vardi, M.Y.: LTLf synthesis under partial observability: from theory to practice. In: GandALF, pp. 1–17. Open Publishing Association (2020). https://doi.org/10.4204/eptcs.326.1

63. Taylor, C.B., Zingaro, D., Porter, L., Webb, K.C., Lee, C.B., Clancy, M.J.: Computer science concept inventories: past and future. Comput. Sci. Educ. **24**(4), 253–276 (2014). https://doi.org/10.1080/08993408.2014.970779

64. Tracy II, T., Tabajara, L.M., Vardi, M., Skadron, K.: Runtime verification on FPGAs with LTLf specifications. In: FMCAD, pp. 36–46 (2020). https://doi.org/10.34727/2020/isbn.978-3-85448-042-6_10

65. Umili, E., Capobianco, R., De Giacomo, G.: Grounding LTLf specifications in images. In: KR, pp. 45–63. ACM (2023).https://doi.org/10.24963/kr.2023/65

66. Vardi, M.Y.: Branching vs. linear time: final showdown. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 1–22. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45319-9_1

67. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification (preliminary report). In: LICS, pp. 332–344. IEEE Computer Society (1986)

68. Wickström, O.: Linear temporal logic visualizer. https://quickstrom.github.io/ltl-visualizer

69. Wilke, T.: Classifying discrete temporal properties. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 32–46. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49116-3_3

70. Wongpiromsarn, T., Ulusoy, A., Belta, C., Frazzoli, E., Rus, D.: Incremental temporal logic synthesis of control policies for robots interacting with dynamic agents. In: IROS, pp. 229–236. IEEE (2012). https://doi.org/10.1109/IROS.2012.6385575

71. Zhu, S., Tabajara, L.M., Li, J., Pu, G., Vardi, M.Y.: Symbolic LTLf synthesis. In: IJCAI, pp. 1362–1369 (2017). https://doi.org/10.24963/ijcai.2017/189