A semantically-derived subset of English for hardware verification

Alexander Holt and **Ewan Klein**
HCRC Language Technology Group
Division of Informatics
University of Edinburgh

alexander.holt@ed.ac.uk ewan.klein@ed.ac.uk

Abstract

To verify hardware designs by model checking, circuit specifications are commonly expressed in the temporal logic CTL. Automatic conversion of English to CTL requires the definition of an appropriately restricted subset of English. We show how the limited semantic expressibility of CTL can be exploited to derive a *hierarchy* of subsets. Our strategy avoids potential difficulties with approaches that take existing computational semantic analyses of English as their starting point—such as the need to ensure that all sentences in the subset possess a CTL translation.

1 Specifications in Natural Language

Mechanised formal specification and verification tools can significantly aid system design in both software and hardware (Clarke and Wing, 1996). One well-established approach to verification, particularly of hardware and protocols, is *temporal model checking*, which allows the designer to check that certain desired properties hold of the system (Clarke and Emerson, 1981). In this approach, specifications are expressed in a temporal logic and systems are represented as finite state transition systems.[1] An efficient search method determines whether the desired property is true in the model provided by the transition system; if not, it provides a counterexample. Despite the undoubted success of temporal model checking as a technique, the requirement that specifications be expressed in temporal logic has proved an obstacle to its take-up by circuit designers and therefore alternative interfaces involving graphics and natural language have been explored. In this paper, we address some of the challenges raised by converting English specifications into temporal logic as a prelude to hardware verification.

One general approach to this kind of task exploits existing results in the computational analysis of natural language semantics, including contextual phenomena such as anaphora and ellipsis, in order to bridge the gap between informal specifications in English and formal specifications in some target formalism (Fuchs and Schwitter, 1996; Schwitter and Fuchs, 1996; Pulman, 1996; Nelken and Francez, 1996). English input sentences are initially mapped into a general purpose semantic formalism such as Discourse Representation Theory (Kamp and Reyle, 1993) or the Core Language Engine's quasi logical form (Alshawi, 1992) at which point context dependencies are resolved. The output of this stage then undergoes a further mapping into the application-specific language which expresses formal specifications. One system which departs from this framework is presented by Fantechi et al. (1994), whose grammar contains special purpose rules for recognising constructions that map directly into ACTL formulas,[2] and can trigger clarification dialogues with the user in the case of a one-to-many mapping.

Independently, the interface may require the user to employ a controlled language, in which syntax and lexicon are restricted in order to minimise ambiguity with respect to the formal specification language (Macias and Pulman, 1995; Fuchs and Schwitter, 1996; Schwitter and Fuchs, 1996). The design of a controlled language is one method of addressing the key problem pointed out by Pulman (1996, p. 235), namely to ensure that *an English input has a valid translation into the target formalism*; this is the problem that we focus on here. Inevitably, we need to pay some attention to

[1] In practice, it turns out to be preferable to use a symbolic representation of the state model, thereby avoiding the *state explosion problem* (Macmillan, 1993).

[2] ACTL is an action-based branching temporal logic which, despite the name, is not directly related to the CTL language that we discuss below.

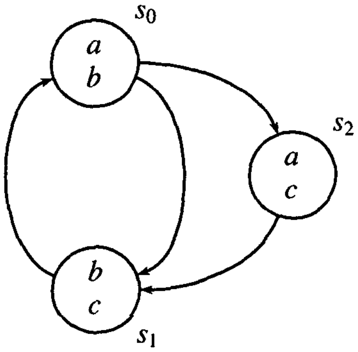

Figure 1: A CTL structure

the syntactic and semantic properties of our target formalism and this is the topic of the next section.

2 CTL Specification and Model Checking

While early attempts to use temporal logics for verification had explored both linear and branching models of time, Clarke et al. (1986) showed that the branching temporal logic CTL (*Computation Tree Logic*) allowed efficient model-checking in place of laborious proof construction methods.[3] In models of CTL, the temporal order relation $<$ defines a tree which branches towards the future. As pointed out by Thomason (1984), branching time provides a basis for formalising the intuition that statements of necessity and possibility are often non-trivially tensed. As we move forward through time, certain possible worlds (i.e., paths in the tree) are eliminated, and thus what was possible at t is no longer available as an option at some t' later than t.

CTL uses formulas beginning with A to express necessity. AGf is true at a time t just in case f is true along all paths that branch forward from the tree at t (true globally). AFf holds when, on all paths, f is true at some time in the future. AXf is true at t when f is true at the next time point, along all paths. Finally, $A[f\ U\ g]$ holds if, for each path, g is true at some time, and from now until that point f is true.

Figure 1, from Clarke et al. (1986), illustrates a CTL model structure, with the relation $<$ represented by arrows between circles (states), and the atomic propositions holding at a state being the letters contained in the circle. A CTL structure gives rise to an infinite computation tree, and Figure 2 shows the initial part of such a tree corresponding to Figure 1, when s_0 is selected as the initial state. States correspond to points of time in the course of a computation, and branches represent non-determinism. Formulas of CTL are either true or false with respect to any given model; see Table 1 for three examples interpreted at s_0 in the Figure 1 structure.

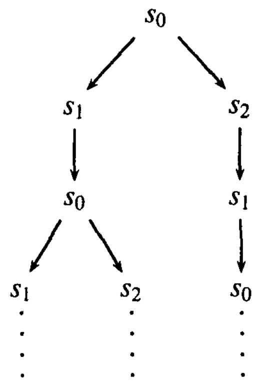

Figure 2: Computation tree

3 Data

One of our key tasks has been to collect an initial sample of specifications in English, so as to identify linguistic constructions and usages typical of specification discourse. We currently have a corpus of around a hundred sentences, most of which were elicited by asking suitably qualified respondents to describe the behaviour manifested by timing diagrams. An example of such a diagram is displayed in Figure 3, which is adapted from one of Fisler's (1996, p. 5).

The horizontal axis of the diagram indicates the passing of time (as measured by clock cycles) and the vertical axis indicates the transition of signals between the states of high and low. (A signal is

formula	*sense*	*at s_0*
AXc	for all paths, at the next state c is true	true
AGb	for all paths, globally b is true	false
$AF(AX(a \wedge b))$	for all paths, eventually there is a state from which, for all paths, at the following state a and b are true	true

Table 1: Interpretation of CTL formulas

[3] Subsequently, model-checking methods which use linear temporal logic have been developed. While theoretically less efficient that those based on CTL, they may turn out to be effective in practice (Vardi, 1998).

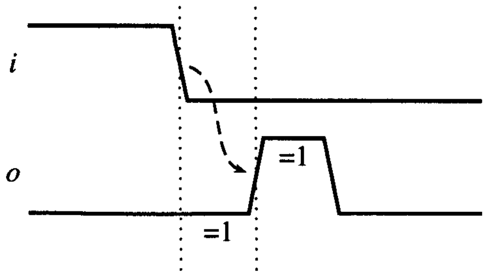

Figure 3: Timing diagram for pulsing circuit

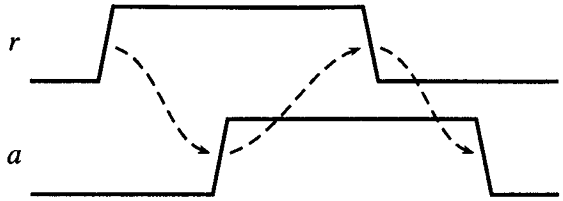

Figure 4: Timing diagram for handshaking protocol

a time-varying value present at some point in the circuit.) In Figure 3, the input signal *i* makes a transition from high to low which after a one-cycle delay triggers a unit-duration pulse on the output signal *o*.

(1a–b) give two possible English descriptions of the regularity illustrated by Figure 3,

(1) a. A pulse of width one is generated on the output *o* one cycle after it detects a falling edge on input *i*.

b. If *i* is high and then is low on the next cycle, then *o* is low and after one cycle becomes high and then after one more cycle becomes low.

while (2) is a CTL description.

(2) $AG(i \rightarrow AX(\neg i \rightarrow (\neg o \wedge AX(o \wedge AX \neg o))))$

A noteworthy difference between the two English renderings is that the first is clearly more abstract than the second. Description (1b) is closer to the CTL formula (2), and consequently easier to translate into CTL.[4]

For another example of the same phenomenon, consider the timing diagram in Figure 4. As before, sentences (3a–b) give two possible English descriptions of the regularity illustrated by Figure 4,

(3) a. Every request is eventually acknowledged and once a request is acknowledged the request is eventually deasserted and eventually after that the acknowledge signal goes low.

b. If *r* rises then after one cycle eventually *a* rises and then after one cycle eventually *r* falls and then after one cycle eventually *a* falls.

which can be rendered in CTL as (4).

(4) $AG(\neg r \wedge AXr \rightarrow AF(\neg a \wedge AX(a$
$\wedge AF(r \wedge AX(\neg r \wedge AF(a \wedge AX \neg a))))))$

Example (3b) parallels (1b) in being closer to CTL than its (a) counterpart. Nevertheless, (3b) is *ontologically richer* than CTL in an important respect, in that it makes reference to the event predicates *rise* and *fall.*

4 Defining a Controlled Language

Even confining our attention to hardware specifications of the level of complexity examined so far, we can conclude there are some kinds of English locutions which will map rather directly into CTL, whereas others have a much less direct relation. What is the nature of this indirect relation? Our claim in this paper is that we can give semantically-oriented characterisations of the relation between complexity in English sentences and their suitability for inclusion in a controlled language for hardware verification. Moreover, this semantic orientation yields a hierarchy of subsets of English. (This hierarchy is a theoretical entity constructed for our specific purposes, of course, not a general linguistic hypothesis about English.)

Our first step in developing an English-to-CTL conversion system was to build a prototype based on the Alvey Natural Language Tools Grammar (Grover et al., 1993). The Alvey grammar is a broad coverage grammar of English using GPSG-style rules, and maps into a event-based, unscoped semantic representation.

For this application, we used a highly restricted lexicon and simplified the grammar in a number of ways (for example: fewer coordination rules; no deontic readings of modals). Tidhar (1998) reports an initial experiment in taking the semantic output generated from a small set *S* of English specifications, and converting it into CTL. Given

[4] Our system does not yet resolve anaphoric references, as in (1a). There are existing English-to-CTL systems which do, however, such as that of Nelken and Francez (1996).

that the Alvey grammar will produce plausible semantic readings for a much larger set S', the challenge is to characterise an intermediate set S, with $S \subset \mathbf{S} \subset S'$, that would admit a translation ϕ into formulas of CTL. Let's assume that we have a reverse translation ϕ^{-1} from CTL to English; then we would like $\mathbf{S} = range(\phi^{-1})$.

4.1 Transliteration

Now suppose that ϕ^{-1} is a *literal* translation from CTL to English. That is, we recurse on the formulas of CTL, choosing a canonical lexical item or phrase in English as a direct counterpart to each constituent of the CTL formula. In fact, we have implemented such a translation as a DCG `ctl2eng`. To illustrate, `ctl2eng` maps the formula (2) into (5):

(5) globally if i is high then after 1 cycle if i is low then o is low and after 1 cycle o is high and after 1 cycle o is low

Let ϕ_{l}^{-1} be the function defined by `ctl2eng`; then we call $\mathcal{L}_{\mathrm{l}} = range(\phi_{\mathrm{l}}^{-1})$ the *canonical transliteration* level of English. We can be confident that it is possible to build a translation ϕ_{l} which will map any sentence in $\mathcal{L}_{\mathrm{l}}$ into a formula of CTL. $\mathcal{L}_{\mathrm{l}}$ can be trivially augmented by adding near-synonymous lexical and syntactic variants. For example, *i is high* can be replaced by *signal i holds*, and *after 1 cycle ...* by *1 cycle later ...* . This adds no semantic complexity. We call the this language (notated $\mathcal{L}_{\mathrm{l}}^{+}$) the *augmented transliteration* level.

One potential problem with defining ϕ_{l} in this way is that the sentences generated by `ctl2eng` soon become structurally ambiguous. We can solve this either by generating unambiguous paraphrases, or by analysing the relevant class of ambiguities and making sure that ϕ_{l} is able to provide all relevant CTL interpretations.

These languages contain only sentences. Hardware specifications often have the form of multi-sentence discourses, however. Such discourses, and the additional phenomena they introduce, occur at higher levels of our language hierarchy, and we presently lack any detailed analysis of them in the terms of this paper.

4.2 Compositional indirect semantics

We'll say that an English input expression has *compositional indirect* semantics just in case

1. there is a compositional mapping to CTL, but where
2. the semantics of the English is ontologically richer than the intended CTL translation.

The best way to explain these notions is by way of some examples. First, consider expressions like the nouns *pulse, edge* and the verbs *rise, fall.* These refer to certain kinds of event. For example, an *edge* denotes the event where a signal changes between two distinct states; from high at time t to low at time $t+1$ or conversely. In CTL, the notion of an edge on signal i corresponds approximately to the following expression:[5]

(6) $(i \wedge AX\neg i) \vee (\neg i \wedge AXi)$

Similarly, a *pulse* can be analysed in terms of a rising edge followed by a falling edge.

What do we mean by saying that there is a *compositional* mapping of locutions at this level to CTL? Our claim is that they can be algorithmically converted into pure CTL without reference to unbounded context. What do we mean by saying that these English expressions involve a richer ontology than CTL? If compositional mapping holds, then clearly we are not forced to augment the standard models for CTL in order to interpret them (although this route might be desirable for other reasons). Rather, we are saying that the 'natural' ontology for these expressions is richer than that allowed for CTL, even if reduction is possible.[6]

4.3 Non-compositional indirect semantics

We consider the conversion to involve *non-compositional indirect semantics* when there is some aspect of non-locality in the domain of the translation function. That is, some form of inference is required—probably involving domain-specific axioms or general temporal axioms—in order to obtain a CTL formula from the English expression.

Here are two examples. The first comes from sentence (3a), where the use of *eventually* might normally be taken to correspond directly to the CTL operator *AF*. However because of the domain of (3a)—a handshaking protocol, evidenced by the use of the verbs *acknowledge* and *request*—it is in fact more accurate to require an extra *AX* in the CTL.

[5] Approximately, in the sense that one cannot simply substitute this expression arbitrarily into a larger formula, as it depends on the syntactic context—for example, whether it occurs in the antecedent or consequent of an implication.

[6] There is a further kind of ontological richness in English at this level, involving the *relation* between events, rather than the events themselves. Space prohibits a closer examination here.

level	*expressiveness*	*examples*
$\mathcal{L}_1$	pure CTL	*i is high; after 1 cycle*
$\mathcal{L}_1^+$	pure CTL	*i holds; 1 cycle later*
$\mathcal{L}_2$	extended CTL	*i rises; there is a pulse of unit duration*
$\mathcal{L}_3$	full SR?	*r is eventually acknowledged*

Table 2: Language hierarchy

This ensures that the three transitions cannot occur at the same time.

We see here an example of domain-specific interpretation conventions that our system needs to be aware of. Clearly, it must incorporate them in such a way that users are still able to reliably predict how the system will react to their English specifications.

The second example is

(7) From one cycle after *i* changes until it changes again *x* and *y* are different.

In this case there is an interaction between a non-local linguistic phenomenon and something specific to the CTL conversion, namely how to make the right connection between the first and the second changes.

4.4 Language hierarchy

Table 2 summarises the main proposals of this section. The left-hand column lists the hierarchy of postulated sublanguages, in increasing order of semantic expressiveness. The middle column tries to calibrate this expressiveness. By 'extended CTL', we mean a superset of CTL which is syntactically augmented to allow formulas such as $rise(p)$, $fall(p)$, discussed earlier, and $pulse(p, v, n)$, where p is an atom, v is a Boolean indicating a high or low value, and n is a natural number indicating duration. The semantic clauses would have to be correspondingly augmented—as carried out for example by Nelken and Francez (1996), for $rise(p)$ and $fall(p)$. By 'full SR', we are hypothesising that it would be necessary to invoke a general semantic representation language for English.

We have constructed a context-free grammar for $\mathcal{L}_2$, in order to obtain a concrete approximation to a controlled subset of English for expressing specifications. There are two cautionary observations. First, as just indicated, $\mathcal{L}_2$ maps directly not into CTL, but into extended CTL. Second, our grammar for $\mathcal{L}_2$ ignores some subtleties of English syntax and morphology. For example, subject-verb agreement; modal auxiliary subcategorisation; varieties of verb phrase modification by adverbs; and forms of anaphora.

These defects in our CFG for $\mathcal{L}_2$ are not fundamental problems, however. The device of using the `ctl2eng` mapping to define a sublanguage is a specific methodology for finding a semantically motivated sublanguage. As such it is only an approximation to the language that we wish our system to deal with. This CFG is *not* the grammar used by our parser (which can, in fact, deal with many of the details of English syntax just mentioned). We may, therefore, introduce a language $\mathcal{L}_2^+$ which corrects the grammatical errors of $\mathcal{L}_2$ and extends it with some degree of anaphora and ellipsis.

We note that it would be useful to have a firmer theoretical grasp on the relations between our sublanguages; we have ongoing work in this area.

5 Conclusion

Much work on controlled languages has been motivated by the ambition to "find the right trade-off between expressiveness and processability" (Schwitter and Fuchs, 1996). An alternative, suggested by what we have proposed here, is to bring into play a *hierarchy* of controlled languages, ordered by the degree to which they semantically approximate the target formalism. Each point in the hierarchy brings different trade-offs between expressiveness and tractability, and evaluating their different merits will depend heavily on the particular task within a generic application domain, as well as on the class of users.

As a final remark, we wish to point out that there may be advantages in identifying plausible restrictions on the target formalism. Dwyer et al. (1998a; 1998b) have convincingly argued that users of formal verification languages make use of recurring *specification patterns*. That is, rather than drawing on the full complexity of languages such as CTL, documented specifications tend to fall into much simpler formulations which express commonly desired properties. In future work, we plan to investigate specification patterns as a further source of constraints that propagate backwards into the controlled English, perhaps providing additional mechanisms for dealing with apparent ambiguity in user input.

Acknowledgements

The work reported here has been carried out as part of PROSPER (Proof and Specification Assisted Design Environments), ESPRIT Framework IV LTR 26241, `http://www.dcs.gla.ac.uk/prosper/`. Thanks to Marc Moens, Claire Grover, Mike Fourman, Dirk Hoffman, Tom Melham, Thomas Kropf, Mike Gordon, and our ACL reviewers.

References

Hiyan Alshawi, editor. 1992. *The Core Language Engine*. MIT Press.

Edmund M. Clarke and E. Allen Emerson. 1981. Synthesis of synchronization skeletons for branching time temporal logic. In *Logic of Programs: Workshop, Yorktown Heights, NY, May 1981*, volume 131 of *Lecture Notes in Computer Science*. Springer-Verlag.

Edmund M. Clarke and Jeanette M. Wing. 1996. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643.

Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263.

Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1998a. Patterns in property specifications for finite-state verification. Technical Report KSU CIS TR-98-9, Department of Computing and Information Sciences, Kansas State University.

Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1998b. Property specification patterns for finite-state verification. In M. Ardis, editor, *Proceedings of the Second Workshop on Formal Methods in Software Practice*, pages 7–15.

A. Fantechi, S. Gnesi, G. Ristori, M. Carenini, M. Marino, and P. Moreschini. 1994. Assisting requirement formalization by means of natural language translation. *Formal Methods in System Design*, 4:243–263.

Kathryn Fisler. 1996. *A Unified Approach to Hardware Verification through a Heterogeneous Logic of Design Diagrams*. Ph.D. thesis, Department of Computer Science, Indiana University.

Norbert E. Fuchs and Rolf Schwitter. 1996. Attempto Controlled English (ACE). In *CLAW 96: First International Workshop on Controlled Language Applications*. Centre for Computational Linguistics, Katholieke Universiteit Leuven, Belgium.

Claire Grover, John Carroll, and Ted Briscoe. 1993. The Alvey Natural Language Tools Grammar (4th release). Technical Report 284, Computer Laboratory, University of Cambridge.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Number 42 in Studies in Linguistics and Philosophy. Kluwer.

Benjamin Macias and Stephen G. Pulman. 1995. A method for controlling the production of specifications in natural language. *The Computer Journal*, 38(4):310–318.

Kenneth L. Macmillan. 1993. *Symbolic Model Checking*. Kluwer.

Rani Nelken and Nissim Francez. 1996. Translating natural language system specifications into temporal logic via DRT. Technical Report LCL-96-2, Laboratory for Computational Linguistics, Technion, Israel Institute of Technology.

Stephen G. Pulman. 1996. Controlled language for knowledge representation. In *CLAW 96: Proceedings of the First International Workshop on Controlled Language Applications*, pages 233–242. Centre for Computational Linguistics, Katholieke Universiteit Leuven, Belgium.

Rolf Schwitter and Norbert E. Fuchs. 1996. Attempto — from specifications in controlled natural language towards executable specifications. In *GI EMISA Workshop*. Natürlichsprachlicher Entwurf von Informations-systemen, Tutzing, Germany.

Richmond H. Thomason. 1984. Combinations of tense and modality. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic. Volume II: Extensions of Classical Logic*, volume 146 of *Synthese Library*, chapter II.3, pages 89–134. D. Reidel.

Dan Tidhar. 1998. ALVEY to CTL translation — A preparatory study for finite-state verification natural language interface. Msc dissertation, Department of Linguistics, University of Edinburgh.

Moshe Y. Vardi. 1998. Linear vs. branching time: A complexity-theoretic perspective. In *LICS'98: Proceedings of the Annual IEEE Symposium on Logic in Computer Science*. Indiana University.