Visualization of Interval Temporal Logic*

Arun Chakrapani Rao,
Antonio Cau and Hussein Zedan,
Software Technology Research Laboratory,
SERCentre, De Montfort University,
The Gateway, Leicester, LE1 9BH, England
Email : {arunc,acau,hzedan}@strl.dmu.ac.uk

Abstract

A graphical notation and an associated tool to specify the behaviour of systems is introduced and is based on 'Interval Temporal Logic' (ITL). It is a part of the so-called 'lean approach' to formal methods.

1 Introduction

The reasons for the lack of widespread acceptability of any particular formal method usually includes the level/kind of mathematics involved and/or the lack of proper tool support. While the scare from the level of mathematics involved is usually a myth[1], we feel that more adequate and appropriate tool support is required to ease the use and understanding of the mathematics involved in dealing with formal specifications. This is the basis for the so-called 'lean formal approach'.

ITL is a flexible notation for both propositional and first-order reasoning about periods of time found in descriptions of hardware and software systems. It can handle both sequential and parallel composition unlike most temporal logics. It offers powerful and extensible specification and proof techniques for reasoning about properties involving safety, liveness and projected time. Tempura[3] provides an executable framework for developing and experimenting with suitable ITL specifications. Therefore, we have chosen ITL for our lean formal approach.

Visual ways of communicating are often better. In engineering, drawings are used to communicate precisely and effectively. In chemical engineering, we find process flow diagrams used to describe the behaviour of entire chemical plants.

Visual notations for describing system behaviour could be more intuitive than textual ways. This is particularly the case when temporal logics are involved ; their many different constructs with various semantics are often confusing. Therefore, a visual notation with proper semantics would be very convenient, particularly if it is very simple, intuitive and based on a formalism like ITL. Moreover, if it is supported by convenient tools, it would go a long way in making formal methods of specification more accessible to a wide spectrum of users.

We also wish to mention here that a visual notation for ITL would greatly aid in the understanding of the refinement steps carried out on a specification. It would greatly help the user in performing a better analysis of the specification being refined by allowing him/her to concentrate better on the relevant portions of the specification (which are often very huge). This aspect of our visual notations is better demonstrated through examples and a suitable tool. In Sect. 3, we will elaborate a bit more on refinement.

2 A visual notation for ITL

ITL in brief. An interval σ is defined as a (in)finite sequence of states σ_0, σ_1, σ_2, .. where σ_i is a mapping from the set of variables 'Var' to the set of values 'Val'. The length of σ is one less than the number of states in the interval.

The syntax of ITL is defined in Fig.1 where μ is an integer value, a is a static variable (doesn't change within an interval), A is a state variable (can change within an interval), v a static or state variable, g is a function symbol and p is a predicate symbol. [2] gives more details on ITL.

Expressions
$e ::= \mu \mid a \mid A \mid g(e_1, \ldots, e_n) \mid \iota a{:}\, f$
Formulae
$f ::= p(e_1, \ldots, e_n) \mid \neg f \mid f_1 \wedge f_2 \mid \forall v \bullet f \mid \mathsf{skip} \mid f_1 ; f_2 \mid f^*$

Figure 1: Syntax of ITL

The syntax. The figures, fig.2 and fig.3, depict the syntax of our visual notation for ITL. Most of the symbols

* in Proceedings of The Fifth Joint Conference on Information Sciences(JCIS 2000), NJ, USA, Feb.27-Mar.03, 2000.

used are self-explanatory. As the figures imply, expressions are enclosed in circles while the formulae are enclosed in rectangular boxes. The corresponding ITL textual notations are also shown in the figure. The following can be noted:

- The visual notation for the 'and composition' is the same as in Statecharts[4].
- The visual notations for 'and' and 'chop' are simple and intuitive.

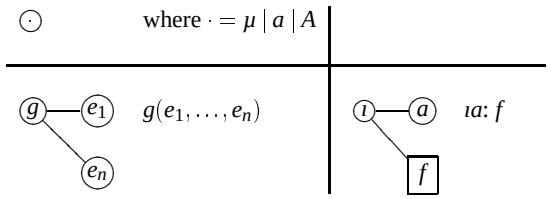

Figure 2: Visual notation for ITL expressions

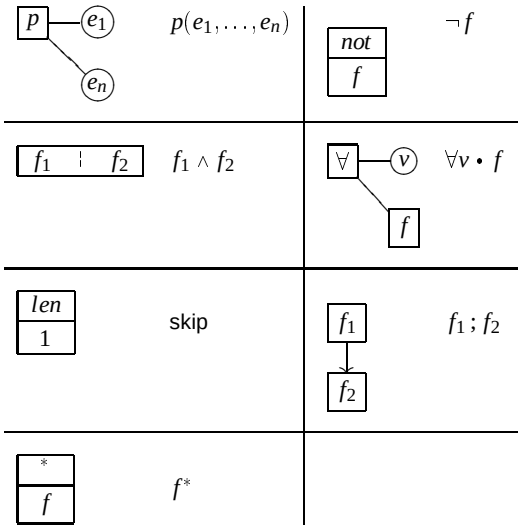

Figure 3: Visual notation for ITL formulae

Some frequently used ITL abbreviations visually. The abbreviated notations help in making the specifications more concise. Some simple and straightforward visual notations for constructs like 'always', 'sometimes' and so on are shown in Fig.4. They help avoid confusion, particularly for users accustomed to other notations in other formalisms.

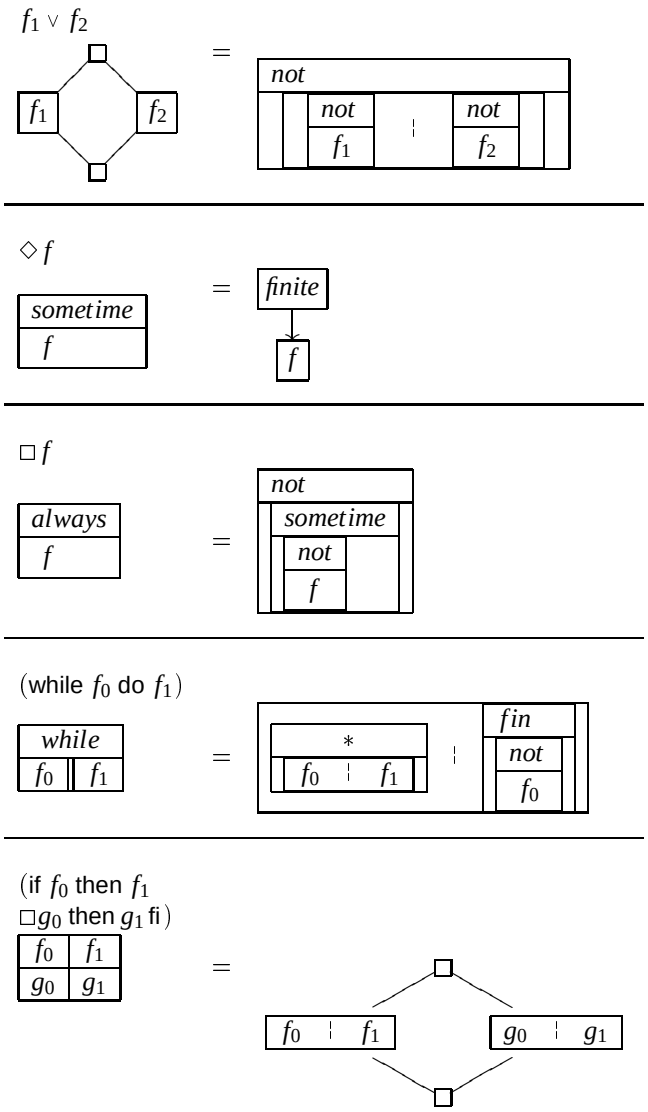

Figure 4: Visual notations for some frequently used ITL abbreviations

3 Refinement

Let $f_0 \sqsubseteq f_1$ denote that f_0 is *refined* by f_1 and be defined as follows:

$$f_1 \supset f_0$$

Visually refinement corresponds to replacing $\boxed{f_0}$ by $\boxed{f_1}$. Some examples of refinement rules are shown in Fig.5 where f_i $(i = 0, 1, 2, 3)$ represents an ITL formula.

The following is a simpler abstract version of a case study showing refinement in [5]. As Fig.6 shows, the abstract specification involving '*' is refined to a 'while loop' using refinement laws of the type just described. Furthermore the 'or composition' and the 'and composition' are replaced by the "if f_1 then $f_2 \Box g_1$ then g_2 fi"

concrete construct. The refinement using visual notation for ITL is better demonstrated through a tool currently under development.

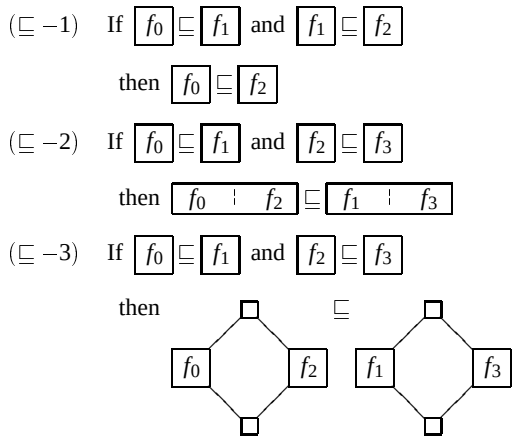

Figure 5: Refinement rules : some examples

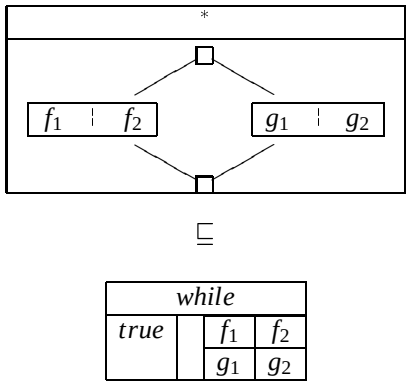

Figure 6: A refinement visually

4 The visual tool

The tool, fig.7, being written in Tcl/Tk, would allow us to do the following :

- Draw visual ITL specifications easily by selecting the visual constructs on the tool bar/menu. This drawing portion of the tool is an extended version of the 'tkpaint[6]' tool freely available.
- Convert a visual ITL specification to its textual ITL form by the click of a button.
- Pretty-print a visual ITL specification.
- Convert an ITL specification into its visual form.
- Refine a visual ITL specification.
- Include help features and examples.

5 Conclusion

We have introduced a simple visual notation and a tool for specifying system behaviour. The tool has many features planned for implementation, as discussed, together with the motivation for such features. In the next coming months, we hope to obtain feedback from various kinds of users to determine the usability of this tool and improve on it. A possible further work in this area includes the abstraction of an executable specification of ITL(viz., Tempura) to a high-level specification in visual form. This ability to abstract using the tool could have applications involving efforts to comprehend or re-engineer legacy code.

References

[1] Bowen, Jonathan P. and Hinchey, Michael G., "The Use of Industrial-Strength Formal Methods", Proceedings of the International Computer Software and Application Conference (COMPSAC'97), Washington D.C., USA, 13-15 August 1997, pages 332-337, IEEE Computer Society Press, 1997.

[2] ITL homepage on the internet, at "http://www.cse.dmu.ac.uk/~cau/itlhomepage".

[3] Moszkowski, Ben, "Executing Temporal Logic Programs", Cambridge University Press, 1986.

[4] Harel, D., "Statecharts : a visual formalism for complex systems", Science of Computer Programming, 8(1):231-274, 1987.

[5] Cau, A., Czarnecki, C., Zedan, H., "Designing a Provably Correct Robot Control System using a 'Lean' Formal Method", In the proceedings of the 5th International Symposium, FTRTFT'98, Lyngby, Denmark, September 1998, volume 1486 of Lecture Notes in Computer Science, pages 123-132.

[6] Tkpaint homepage on the internet, at "http://www.netanya.ac.il/~samy/tkpaint.html".

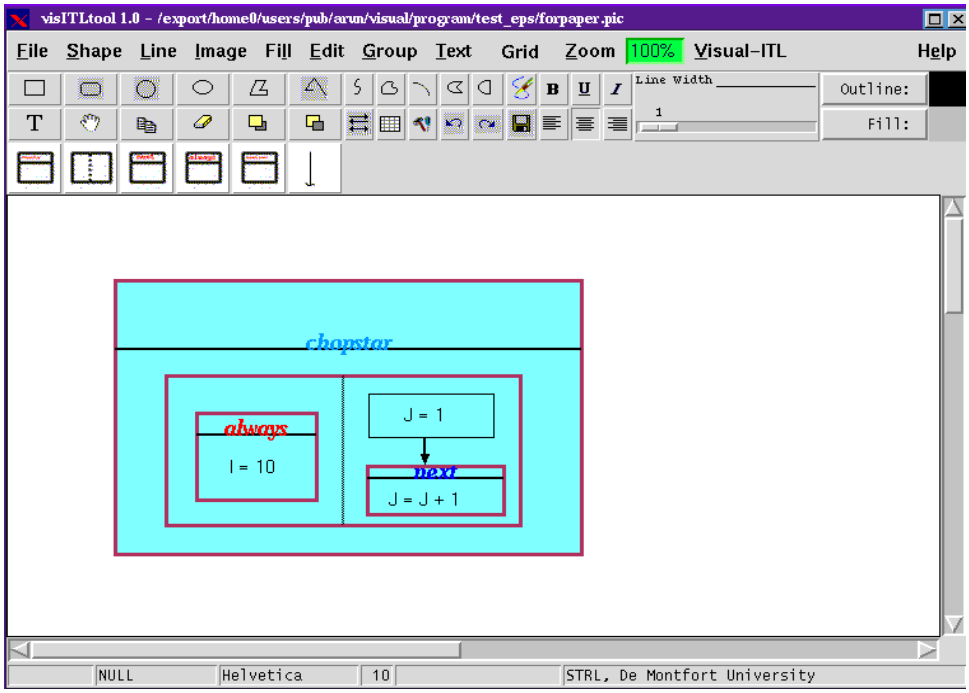

Figure 7: The visualization tool for ITL