

ITL as a Frontend and Backend Formalism for Discrete Linear Time

Ben Moszkowski

Software Technology Research Laboratory

De Montfort University

Leicester

Great Britain

email: **benm@dmu.ac.uk**

<http://www.tech.dmu.ac.uk/~benm>

Introduction

- **Intervals** and **discrete linear state sequences** offer a compellingly natural and flexible way to model computational processes involving hardware or software.

Finite or infinite state sequence: • • • • • •

- **Interval Temporal Logic (ITL)** is an established formalism (over 25 years old) for reasoning about such phenomena.
- It includes operators for **sequentially** combining formulas.
- For example, if A and B are formulas, so are following

$A; B$ (“**chop**”) A^* (“**chop-star**”).

- ITL can express various **imperative programming** constructs (e.g., **while-loops**) and has **executable subsets**.
- The **Duration Calculus (DC)** is a **real-time** extension of ITL for hybrid systems.

Some Sample Uses of ITL and Intervals

We will briefly discuss some instances of ITL as a **frontend** and **backend**:

- KIV interactive theorem prover
- Model checking with Duration Calculus
- Dynamic authorisation policies
- Concurrent Separation Logic: **temporal separation**
- In IEEE Standard 1647: **temporal 'e'**
- Our own current work

ITL and the Interactive Theorem Prover KIV

- KIV is an interactive theorem prover based at the Univ. of Augsburg, Germany.
- KIV uses ITL as a calculus for symbolic execution of concurrent systems.
- ITL variants can be used directly as a **frontend** notation.
- ITL also serves as a **backend** for UML, Statecharts and other notations.
- Supports compositionality and rely-guarantee conditions.

Sample recent paper:

S. Bäuml et al. *Proving Linearizability with Temporal Logic*.
FACS Journal, 2009.

Comments by KIV group

From S. Bäumlner et al. (2009):

Our **ITL** variant supports classic temporal logic operators as well as program operators.

The interactive verifier KIV allows us to directly verify parallel programs in a **rich programming language** using the **intuitive proof principle of symbolic execution**. An additional translation to a special normal form (as e.g. in TLA) using explicit program counters is not necessary.

Model checking with Duration Calculus

Duration Calculus is an extension of ITL to real time.

First proposed by Zhou, Hoare and Ravn in early 90s.

Recent book by E.-R. Olderog and H. Dierks:

REAL-TIME SYSTEMS: Formal Specification and Automatic Verification.

Cambridge University Press, 2008.

- Describes embedded safety-critical applications.
- Uses real-time specification techniques.
- Book combines **logic** with **automata**: **Duration Calculus**, timed automata, and PLC-automata.
- Go from **Duration Calculus** down to **source code** for hardware platforms of embedded systems.

Some Comments on New Book

- Kim Guldstrand Larsen, Aalborg University, Denmark:

At last we have the book on the design of real-time systems which successfully combine mathematical models of systems with that of their practical implementation and automatic verification using state-of-the-art model checking.

- Tony Hoare, Microsoft Research, Cambridge:

[The book] shows how theories can be elegantly unified, and how tools can be soundly based on theory, and seamlessly integrated.

- Amir Pnueli, Weizmann Inst. of Science and New York Univ.:

[The book] is the first text that presents a complete and well-rounded process for the systematic development of real-time systems supported by formal methods and automated tools, whenever possible.

Dynamic Authorisation Policies

Moritz Becker

Microsoft Research, Cambridge

Goal: Specification of dynamic authorisation policies, i.e., rules governing actions that may depend on and update the authorisation state.

Uses **Transaction Logic (TR)** – developed in early 90s by Bonner and Kifer for logic programming.

TR contains ITL's **chop** – called **sequential conjunction**.

Here ITL serves as a **backend**.

Article by M. Becker and S. Nanz to appear in *ACM Transactions on Information and System Security*.

Concurrent Separation Logic – Temporal Separation

Tony Hoare, Microsoft Research, Cambridge

Peter O'Hearn, Queen Mary, University of London

Separation Logic already has a notion of **spatial separation**.

Hoare and O'Hearn consider **separation in time** in the article

Separation Logic Semantics for Communicating Processes.

They use state sequences and sequential operator similar to ITL's.

Quote from Zhou Chaochen in recent book:

Moreover, in a personal communication, Tony Hoare drew my attention to the possible correspondence between the chop operator in **Interval Logic** and the separating (spatial) conjunction in **Separation Logic**. I believe that **Interval Logic** deserves an important role in computing science.

Temporal 'e'

The **'e' language** (IEEE Standard 1647) supports functional simulation-based verification of digital devices such as processor cores, DSPs, microcontrollers and memory interfaces.

Temporal 'e' can specify events typically representing outputs from a hardware device under test.

Contains temporal operators for combining events into complex temporal expressions, including variant of **ITL's chop**.

It has a rigorous mathematical semantics.

Both **'e'** and **temporal 'e'** were developed by Verisity, Inc. which was later acquired by Cadence Design Systems, Inc., a leading firm in electronic design automation (EDA) technologies.

See also use of **chop** in IEEE Standard 1850 (**PSL/Sugar**) and part of IEEE Standard 1800 (**System Verilog Assertions (SVA)**).

Our Own Current Work

- Solve longstanding open problem: Proof of **axiomatic completeness** for **propositional ITL with infinite time**.
 - This also serves as a unifying basis for some extensions and other related logics sharing **nonelementary** computational complexity.
 - We dispense with practice of **almost 40 years** of **embedding intricate proofs** involving **complementing nondeterministic automata** for **infinite words**. Bound variables not needed.
- An ITL-based generalised Hoare logic with **temporal assertions** and **nestable** sequential and parallel triples.
- Could combine these to reason about reactive systems which do not necessarily terminate.
- ITL appears to offer other significant features such as acting a basis for a **“logical physics” of discrete linear time**.

Conclusions

We have presented several instances of ITL as a **frontend** or **backend** formalism.

We are optimistic that ITL will find other uses as well since ...

- Intervals, state sequences and sequential composition are fundamental concepts.
- ITL and Duration Calculus are basic logics for them.

So is Zhou Chaochen:

I believe that **Interval Logic** deserves an important role in computing science.

For more information about ITL:

ITL webpages – maintained by Antonio Cau:

<http://www.tech.dmu.ac.uk/STRL/ITL/>

Google search: **Interval Temporal Logic**