

Functionals for the Semantic Specification of Temporal Formulas for Model Checking

Raymond Boute and Hannes Verlinde

INTEC, Ghent University, Belgium,
boute@intec.UGent.be hannes.verlinde@UGent.be,
WWW home page:

<http://www.intec.UGent.be/groupsites/formal/BoutHomeFram.htm>

Abstract. Specifying behaviour by temporal formulas is facilitated by the following approach. So-called *temporal patterns* (defined via *generic functionals*) allow describing temporal relationships by expressions that closely reflect the informal word statements. Equivalence of these expressions with the temporal formulas meant for model checking can be proven formally in a functional predicate calculus. This increases confidence as to whether the temporal formulas indeed reflect the desired behaviour.

1 Introduction

In model checking, formal specification of behaviour is often expressed by temporal logic formulas [9–11]. The problem addressed in this paper was inspired by a tutorial [6] on Bandera, a system for model checking Java source code [1], but directly extends to temporal specification in general and is treated as such.

In the cited tutorial [6], the lecturer made the interesting observation that reading and writing temporal formulas may be difficult, nonintuitive, and requires considerable expertise. Only for simple examples the intuitive interpretation is straightforward, for instance, interpreting the LTL-formula $\Box\Diamond p$ as expressing that p happens infinitely often [10]. Yet, even then a formal proof remains an instructive exercise that paves the way for the sequel.

The Bandera group introduced so-called *temporal specification patterns* [6–8] that are more readable because they reflect the informal word specification in a rather direct way. Yet, two shortcomings remain: (a) the design is not as orthogonal and compositional as it could be, and (b) establishing that certain temporal formulas do express the same behaviour as the patterns still requires expertise, since no systematic method was given for this task.

This gap can be bridged by expressing the specification in a compositional way using formulas that closely reflect the informal word statements about the desired temporal behaviour. We describe important (often-used) temporal relationships in a functional predicate calculus [3] and using *generic functionals* [5] as auxiliary functions. This complements an earlier *functional temporal calculus* [2] with a comprehensive collection of calculation rules. The resulting expressions are called *temporal patterns*, reflecting analogy with [6–8], and avoiding confusion with other uses of the term “pattern” in software engineering.

Correspondence with temporal logic formulas can be established via their semantics [2, 4], expressed in the same predicate calculus, and can be used in various ways: proving equality between a temporal formula and a temporal pattern, or transforming one into the other. Either increases confidence as to whether the temporal formulas used for model checking indeed reflect the desired behaviour.

2 Functional mathematics: principle and key conventions

Functional Mathematics [3] is an approach to structure formalisms by conceiving mathematical objects as functions whenever convenient — which is quite more often than common practice reflects. Here we only provide a rather compact summary of the key conventions used in the sequel. Full details are given in [3].

A *function* f is fully defined by its domain $\mathcal{D}f$ and its mapping (image for every domain element). Some functions can be denoted by an *abstraction* of the form $x : X \wedge p . e$ ($\wedge p$ optional). Writing f for $v : X \wedge p . e$, the domain axiom is $d \in \mathcal{D}f \equiv d \in X \wedge p[d]_d^v$ and the mapping axiom $d \in \mathcal{D}f \Rightarrow f d = e[d]_d^v$. Here $e[d]_d^v$ is e with d substituted for v . Example: $n : \mathbb{Z} . 2 \cdot n$ doubles integers. Another example is the *constant function definer* \bullet with $X \bullet e = v : X . e$ (taking v not free in e).

Predicates are \mathbb{B} -valued functions ($\mathbb{B} = \{0, 1\}$). The *quantifiers* \forall and \exists are defined as predicates over predicates: $\forall P \equiv P = \mathcal{D}P \bullet 1$ and $\exists P \equiv P \neq \mathcal{D}P \bullet 0$. Writing predicates as abstractions yields familiar expressions such as $\forall P \equiv \forall x : \mathcal{D}P . Px$ and $\forall x : \mathbb{R} . x^2 \geq 0$. Calculation rules are given in [3].

Generic functionals are functions over arbitrary functions. Here we use only a few from a rather extensive collection [5]. The *range* operator \mathcal{R} has axiom $e \in \mathcal{R}f \equiv \exists x : \mathcal{D}f . fx = e$. Using $\{-\}$ as synonym for \mathcal{R} synthesises set notations such as $\{m : \mathbb{N} \mid m < n\}$, with the general convention that $x : X \mid p$ is shorthand for the abstraction $x : X \wedge p . x$. Expressions like $\{e, e', e''\}$ also have their usual meaning. The *function arrow* operator \rightarrow is defined by $f \in X \rightarrow Y \equiv \mathcal{D}f = X \wedge \mathcal{R}f \subseteq Y$, for any sets X and Y . The generic *set filtering* operator \downarrow is defined by $X \downarrow P = \{x : X \cap \mathcal{D}P \mid Px\}$ for any set X and predicate P .

A *sequence* is any function with domain $\square n$, with $n : \mathbb{N}$ or $n := \infty$. The *block* operator \square is defined by $\square n = \{m : \mathbb{N} \mid m < n\}$, so $\square 0 = \emptyset$, $\square 2 = \mathbb{B}$ and $\square \infty = \mathbb{N}$. The *length* operator $\#$ is defined by $\#x = n \equiv \mathcal{D}x = \square n$, and the *contiguous integer set* operator $_$ by $m_ n = \{k : \mathbb{Z} \mid m \leq k < n\}$. An array of length n over set A is a function of type $\square n \rightarrow A$, written A^n . The set of lists over A is $\bigcup n : \mathbb{N} . A^n$, written A^* , and A^+ is the set of nonempty lists.

3 Temporal specification patterns, temporal logic and functional temporal calculus

3.1 Temporal patterns: classification and functional definition

Classification Temporal patterns reflect temporal relationships or properties. The usual distinction between *safety* and *liveness* properties is well-known but rather coarse, and a more refined classification is based on the syntactic structure

of the temporal formulas expressing the properties [10]. Bandera uses a higher level of abstraction, more closely corresponding to expressing properties in a quasi-natural language. The classes are hierarchically subdivided in 2 groups:

- Occurrence Patterns*: Absence, Existence, Bounded Existence, Universality,
- Order Patterns*: Precedence, Response, Chain Precedence, Chain Response.

Conventions Let \mathbb{T} be a time domain with order \leq , and A a set of atomic propositions taking values in \mathbb{B} at any time instant. The *state* at a given time instant is a function of type $A \rightarrow \mathbb{B}$, and behaviour then has type $\mathbb{T} \rightarrow A \rightarrow \mathbb{B}$. Alternatively, as in [4], a *signal* associated with an element of A is a function of type $\mathbb{T} \rightarrow \mathbb{B}$, and behaviour then has type $A \rightarrow \mathbb{T} \rightarrow \mathbb{B}$. Either formulation is useful, and switching is easy by swapping arguments using the generic *transposition* functional [5]. Henceforth, $\mathbb{T} = \mathbb{N}$, and we write $\mathbb{N} \rightarrow \mathbb{B}$ as Pred .

The *scope* of a temporal pattern is the set of subsets (*regions*) of \mathbb{N} to which it pertains. For uniformity, we specify regions by predicates in Pred . The generic set filtering operator allows expressing a region as $\mathbb{N} \downarrow s$, abbreviated \mathbb{N}_s .

Occurrence patterns are of type $\text{Pred} \rightarrow \text{Pred} \rightarrow \mathbb{B}$ or $\text{Pred} \rightarrow \mathbb{N} \rightarrow \text{Pred} \rightarrow \mathbb{B}$. For any predicate $p : \text{Pred}$, region $s : \text{Pred}$, number $k : \mathbb{N}$, we define

- *Absence* (is false) by p is false $s \equiv \forall n : \mathbb{N}_s . \neg(pn)$.
- *Existence* (becomes true) by p becomes true $s \equiv \exists n : \mathbb{N}_s . pn$.
- *Bounded Existence* (occurs) by p occurs $_k$ $s \equiv \exists f : \mathcal{F}_{\text{inj}} . \mathcal{D}f = \square k \wedge \mathcal{R}f \subseteq \{n : \mathbb{N}_s \mid pn \wedge (n-1 \in \mathbb{N}_s \Rightarrow \neg p(n-1))\}$.
- *Universality* (is true) by p is true $s \equiv \forall n : \mathbb{N}_s . pn$.

Order patterns are of type $\text{Pred}^+ \rightarrow \text{Pred} \rightarrow \mathbb{B}$ or variants. For any predicates p and q in Pred , predicate lists p' and q' in Pred^+ , region $s : \text{Pred}$, we define

- *Precedence* (precedes) by p precedes q $s \equiv \forall n : \mathbb{N}_s . qn \Rightarrow \exists m : \mathbb{N}_s . pm \wedge m < n$.
- *Response* (responds to) by p responds to q $s \equiv \forall n : \mathbb{N}_s . qn \Rightarrow \exists m : \mathbb{N}_s . pm \wedge m \geq n$.
- *Chain Precedence* (precedes) by p' precedes q' $s \equiv \forall n : \mathbb{N}_s^{\#q'} . (\forall i : \mathcal{D}q' . q'i(ni) \wedge (1 \leq i \Rightarrow n(i-1) < ni)) \Rightarrow \exists m : \mathbb{N}_s^{\#p'} . (\forall i : \mathcal{D}p' . p'i(mi) \wedge (1 \leq i \Rightarrow m(i-1) < mi)) \wedge m(\#p'-1) < n0$.
- *Chain Response* (responds to) by p' responds to q' $s \equiv \forall n : \mathbb{N}_s^{\#q'} . (\forall i : \mathcal{D}q' . q'i(ni) \wedge (1 \leq i \Rightarrow n(i-1) < ni)) \Rightarrow \exists m : \mathbb{N}_s^{\#p'} . (\forall i : \mathcal{D}p' . p'i(mi) \wedge (1 \leq i \Rightarrow m(i-1) < mi)) \wedge m0 \geq n(\#q'-1)$.

Pattern scope These operators are of type $(\text{Pred} \rightarrow \mathbb{B}) \rightarrow \text{Pred}^* \rightarrow \mathbb{B}$ or variants. In typical use, their first argument (of type $\text{Pred} \rightarrow \mathbb{B}$) is a temporal pattern with the region s omitted, as in p occurs $_2$ and (p, q) precedes r . Possible other arguments further specify the scope. For any $p : \text{Pred} \rightarrow \mathbb{B}$ and q and r in Pred , we define

- *Globally* by p globally $\equiv p(\mathbb{N} \bullet 1)$.
- *Before* by p before $q \equiv (\exists n : \mathbb{N} . q n) \Rightarrow \exists n : \mathbb{N}_q . (\forall m : \mathbb{N}_q . m \geq n) \wedge p(< n)$.
- *Since* by p since $q \equiv (\exists n : \mathbb{N} . q n) \Rightarrow \exists n : \mathbb{N}_q . (\forall m : \mathbb{N}_q . m \geq n) \wedge p(\geq n)$.
- *Between* by p between q and $r \equiv$
 $\forall n : \mathbb{N}_q . (\exists m : \mathbb{N}_r . m \geq n) \Rightarrow \exists m : \mathbb{N}_r . m \geq n \wedge (\forall i : n_m . \neg(r i)) \wedge p(\in n_m)$.
- *Since-until* by p since q until $r \equiv p$ between q and $r \wedge$
 $\forall n : \mathbb{N}_q . (\forall m : \mathbb{N}_r . m \leq n) \Rightarrow$
 $\exists m : \mathbb{N}_q . (\forall i : \mathbb{N}_q . i < m \Rightarrow \exists j : i_m . r j) \wedge p(\geq m)$.

3.2 Linear Time Temporal logic (LTL)

Syntactically, assume a set V of variables and a set B of propositional expressions built from variables and atomic propositions. The set L of LTL expressions is recursively defined as one of the following: $\llbracket b \rrbracket$, $\llbracket (\neg p) \rrbracket$, $\llbracket (p \vee q) \rrbracket$, $\llbracket (\bigcirc p) \rrbracket$, $\llbracket (p \mathcal{U} q) \rrbracket$, where b is in B , and p and q in L . Derived operators are defined by

$$\begin{aligned} \llbracket (p \rightarrow q) \rrbracket &= \llbracket ((\neg p) \vee q) \rrbracket & \llbracket (\diamond p) \rrbracket &= \llbracket (1 \mathcal{U} p) \rrbracket \\ \llbracket (p \wedge q) \rrbracket &= \llbracket (\neg(\neg(p \rightarrow \neg q))) \rrbracket & \llbracket (\square p) \rrbracket &= \llbracket (\neg(\diamond(\neg p))) \rrbracket \\ & & \llbracket (p \mathcal{W} q) \rrbracket &= \llbracket ((\square p) \vee (p \mathcal{U} q)) \rrbracket \end{aligned}$$

We also adopt the usual conventions for omitting parentheses.

The semantics can be defined by a state-oriented model, with state space $S := V \rightarrow \mathbb{B}$ and state sequence space $\mathbb{N} \rightarrow S$. Letting $\mathcal{E} : B \rightarrow S \rightarrow \mathbb{B}$ be the meaning function for B , we define the meaning function $\mathcal{M} : L \rightarrow (\mathbb{N} \rightarrow S) \rightarrow \mathbb{B}$ for L as usual: for any $\sigma : \mathbb{N} \rightarrow S$, $b : B$, $p : L$, $q : L$ and $n : \mathbb{N}$,

$$\begin{aligned} \mathcal{M} \llbracket b \rrbracket \sigma n &\equiv \mathcal{E} b(\sigma n) \\ \mathcal{M} \llbracket (\neg p) \rrbracket \sigma n &\equiv \neg(\mathcal{M} \llbracket p \rrbracket \sigma n) \\ \mathcal{M} \llbracket (p \vee q) \rrbracket \sigma n &\equiv (\mathcal{M} \llbracket p \rrbracket \sigma n) \vee (\mathcal{M} \llbracket q \rrbracket \sigma n) \\ \mathcal{M} \llbracket (\bigcirc p) \rrbracket \sigma n &\equiv \mathcal{M} \llbracket p \rrbracket \sigma(n+1) \\ \mathcal{M} \llbracket (p \mathcal{U} q) \rrbracket \sigma n &\equiv \exists m : \mathbb{N}_{\geq n} . \mathcal{M} \llbracket q \rrbracket \sigma m \wedge \forall i : n_m . \mathcal{M} \llbracket p \rrbracket \sigma i \end{aligned}$$

Alternatively, in a signal-oriented model [4], we replace the state sequence space $\mathbb{N} \rightarrow V \rightarrow \mathbb{B}$ by a signal space $V \rightarrow \mathbb{N} \rightarrow \mathbb{B}$. Correspondingly, the function $\mathcal{M} : L \rightarrow (\mathbb{N} \rightarrow V \rightarrow \mathbb{B}) \rightarrow \mathbb{B}$ becomes $\mathcal{M}' : L \rightarrow (V \rightarrow \mathbb{N} \rightarrow \mathbb{B}) \rightarrow \mathbb{B}$. As expected, $\mathcal{M}' e \nu = \mathcal{M} e \nu^\top$ or, directly, for any $\nu : V \rightarrow \mathbb{N} \rightarrow \mathbb{B}$, $b : B$, $p : L$ and $n : \mathbb{N}$, define $\mathcal{M}' \llbracket b \rrbracket \nu n \equiv \mathcal{E} b(\nu^\top n)$ and $\mathcal{M}' \llbracket (\neg p) \rrbracket \nu n \equiv \neg(\mathcal{M}' \llbracket p \rrbracket \nu n)$ and so on.

Since the syntax of LTL has logical operators, we can assume $B = V \cup A$ and replace $\mathcal{M}' \llbracket b \rrbracket \nu n \equiv \mathcal{E} b(\nu^\top n)$ by $\mathcal{M}' \llbracket a \rrbracket \nu n \equiv \kappa a$ and $\mathcal{M}' \llbracket v \rrbracket \nu n \equiv \nu v n$.

3.3 Functional Temporal Calculus (FTC)

The Functional Temporal Calculus [2] facilitates calculation by obviating the cumbersome and repetitive meaning function. To this effect, LTL-expressions are identified with their interpretation as signals of type $\mathbb{N} \rightarrow \mathbb{B}$, i.e., Pred . Hence

def \neg : Pred \rightarrow Pred **with** $\neg p n \equiv \neg(p n)$
def \vee : Pred \rightarrow Pred \rightarrow Pred **with** $(p \vee q) n \equiv p n \vee q n$
def \circ : Pred \rightarrow Pred **with** $(\circ p) n \equiv p(n + 1)$
def \mathcal{U} : Pred \rightarrow Pred \rightarrow Pred **with** $(p \mathcal{U} q) n \equiv \exists m : \mathbb{N}_{\geq n} . q m \wedge \forall i : n..m . p i$

Alternatively, we can omit the first two definitions and use $\bar{\neg}$ and $\hat{\vee}$ instead, based on the generic *direct extension* operator [5]. In a broad context, this is even advisable, but in the restricted context of temporal logic it matters little.

4 Formal verification

Temporal patterns and temporal formulas With the preceding definitions, we can formally establish correspondence between our formalized versions of Bandera temporal specification patterns and temporal formulas, for instance

$$\begin{aligned}
p \text{ responds to } q \text{ globally} & : \Box (q \rightarrow \Diamond p) \\
p \text{ becomes true before } q & : \neg q \mathcal{W} (p \wedge \neg p) \\
p \text{ is false since } q & : \Box (q \rightarrow \Box (\neg p)) \\
p \text{ is true between } q \text{ and } r & : \Box ((q \wedge \neg r \wedge \Diamond r) \rightarrow (p \mathcal{U} r))
\end{aligned}$$

In the formal proofs, we use the conventions of FTC and the calculation rules of the functional predicate calculus [3]. Often the least element axiom for natural numbers is necessary: $\exists (n : \mathbb{N} . P n) \equiv \exists n : \mathbb{N} . P n \wedge \forall i : \mathbb{N} . P i \Rightarrow i \geq n$.

A formal proof example Of the preceding correspondences, we prove the most complicated one, leaving the others to the reader. We shall use two lemmata, both for arbitrary $n : \mathbb{N}$, $p : \text{Pred}$ and $r : \text{Pred}$, and proven using the rules from [3].

Lemma 1. $r n \Rightarrow \exists m : \mathbb{N} . m \geq n \wedge r m \wedge \forall i : n..m . p i$.

Lemma 2. $\exists (m : \mathbb{N} . m \geq n \wedge r m \wedge \forall i : n..m . p i) \equiv \exists m : \mathbb{N} . m \geq n \wedge r m \wedge (\forall i : n..m . \neg(r i)) \wedge \forall i : n..m . p i$.

Theorem 1. p is true between q and $r \equiv \Box ((q \wedge \neg r \wedge \Diamond r) \rightarrow (p \mathcal{U} r)) 0$

Proof. $\Box ((q \wedge \neg r \wedge \Diamond r) \rightarrow (p \mathcal{U} r)) 0$
 $\equiv \langle \text{Definition } \Box \rangle \forall n : \mathbb{N}_{\geq 0} . ((q \wedge \neg r \wedge \Diamond r) \rightarrow (p \mathcal{U} r)) n$
 $\equiv \langle \text{Definition } \rightarrow \rangle \forall n : \mathbb{N} . (q \wedge \neg r \wedge \Diamond r) n \Rightarrow (p \mathcal{U} r) n$
 $\equiv \langle \text{Defin. } \wedge, \mathcal{U} \rangle \forall n : \mathbb{N} . q n \wedge \neg r n \wedge \Diamond r n \Rightarrow \exists m : \mathbb{N}_{\geq n} . r m \wedge \forall i : n..m . p i$
 $\equiv \langle \text{Defin. } \neg, \Diamond \rangle$
 $\forall n : \mathbb{N} . q n \wedge \neg(r n) \wedge \exists (m : \mathbb{N}_{\geq n} . r m) \Rightarrow \exists m : \mathbb{N}_{\geq n} . r m \wedge \forall i : n..m . p i$
 $\equiv \langle \text{Shunting, trading } \exists \rangle \forall n : \mathbb{N} . q n \Rightarrow \exists (m : \mathbb{N} . m \geq n \wedge r m) \Rightarrow \neg(r n) \Rightarrow$
 $\exists m : \mathbb{N} . m \geq n \wedge r m \wedge \forall i : n..m . p i$
 $\equiv \langle \text{Lemma 1, } (x \Rightarrow y) \wedge (\neg x \Rightarrow y) \equiv y \rangle$
 $\forall n : \mathbb{N} . q n \Rightarrow \exists (m : \mathbb{N} . m \geq n \wedge r m) \Rightarrow \exists m : \mathbb{N} . m \geq n \wedge r m \wedge \forall i : n..m . p i$
 $\equiv \langle \text{Trading } \forall, \text{ trading } \exists, \text{ lemma 2} \rangle \forall n : \mathbb{N}_q . \exists (m : \mathbb{N}_r . m \geq n) \Rightarrow$
 $\exists m : \mathbb{N} . m \geq n \wedge r m \wedge \forall (i : n..m . \neg(r i)) \wedge \forall i : n..m . p i$

$$\begin{aligned}
&\equiv \langle \text{Trading } \exists, n_m = \mathbb{N}_{\in n_m} \rangle \\
&\forall n : \mathbb{N}_q. \exists (m : \mathbb{N}_r. m \geq n) \Rightarrow \exists m : \mathbb{N}_r. m \geq n \wedge \forall (i : n_m. \neg(r\ i)) \wedge \forall i : \mathbb{N}_{\in n_m}. p\ i \\
&\equiv \langle \text{Definition } \textit{Universality} \rangle \quad \forall n : \mathbb{N}_q. \exists (m : \mathbb{N}_r. m \geq n) \Rightarrow \\
&\exists m : \mathbb{N}_r. m \geq n \wedge \forall (i : n_m. \neg(r\ i)) \wedge p \text{ is true } (\in n_m) \\
&\equiv \langle \text{Definition } \textit{Between} \rangle \quad p \text{ is true between } q \text{ and } r \quad \square
\end{aligned}$$

5 Conclusion and future work

Temporal specification patterns were introduced by the Bandera group in the context of model checking Java code. We have shown how the correspondence between temporal patterns and LTL-formulas can be proved formally. The approach is not restricted to LTL and has more general applications than model checking of software, such as proving equivalence between formulas in different temporal logics. Such proofs often involve manipulating complicated predicate formulas, which can be tedious. To address this matter, we have experimented with Isar/HOL as a proof assistant. Further generalization and automation of our approach and improving compositionality by generic functionals will be the main objects of future research.

References

1. *Bandera Home Page*. <http://www.cis.ksu.edu/~santos/bandera>
2. Raymond Boute, “A calculus for reasoning about temporal phenomena”, *NGI-SION Symposium 1986*, pp. 405-411, April 1986.
3. Raymond Boute, *Functional Mathematics: a Unifying Declarative and Computational Approach to Systems, Circuits and Programs — Part I*. Course notes, Ghent University, 2002.
4. Raymond Boute, “Functional characterization of discrete systems”, chapter M-04 in: *Formele Systeemmodellen*. Course notes, Ghent University, 2002.
5. Raymond Boute, “Concrete Generic Functionals: Principles, Design and Applications”, in: Jeremy Gibbons and Johan Jeuring, eds., *Generic Programming*, pp. 89-119, Kluwer, 2003.
6. Matthew B. Dwyer and John Hatcliff, *Bandera Temporal Specification Patterns*, tutorial presentation at ETAPS’02 (Grenoble) and SMF’02 (Bertinoro), 2002. <http://www.cis.ksu.edu/~santos/bandera/Talks/SMF02/02-SML-Patterns.ppt>
7. Matthew B. Dwyer, George S. Avrunin and James C. Corbett, “Property Specification Patterns for Finite-State Specification”, in: Mark Ardis, editor, *Proc. FMSP’98, Second Workshop on Formal Methods in Software Practice*, pp. 7-15, Clearwater Beach, FL, March 1998.
8. Matthew B. Dwyer, George S. Avrunin and James C. Corbett, “Patterns in Property Specification for Finite-State Specification”, in: *Proc. Twenty-First Intl. Conf. on Software Engineering*, pp. 411-420, Los Angeles, May 1999.
9. Gerard J. Holzmann, *Design and Validation of Computer Protocols*, Prentice Hall, 1990.
10. Zohar Manna and Amir Pnueli, *The Temporal Logic of Reactive and Concurrent Systems: Safety*, Springer-Verlag, 1995.
11. P. S. Thiagarajan, *Introduction to Temporal Logics*, lectures 4-5: LTL, National University of Singapore, 2002. <http://www.comp.nus.edu.sg/~cs5236>