Information Processing Letters 113 (2013) 350–353

Contents lists available at SciVerse ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Interconnections between classes of sequentially compositional temporal formulas

Ben Moszkowski

Software Technology Research Laboratory, De Montfort University, Leicester, UK

ARTICLE INFO

Article history:
Received 12 November 2012
Received in revised form 30 January 2013
Accepted 6 February 2013
Available online 8 February 2013
Communicated by J.L. Fiadeiro

Keywords:
Formal methods
Interval Temporal Logic
Compositionality

ABSTRACT

Interval Temporal Logic (ITL) is an established formalism for reasoning about time periods. We elucidate here the relationship between various kinds of compositional propositional ITL formulas. Several are closed under conjunction and the standard temporal operator known as "box" and "always".

1. Introduction

Intervals and discrete linear state sequences offer a compellingly natural and flexible way to model computational processes involving hardware or software. *Interval Temporal Logic* (ITL) [1], an established formalism for reasoning about such phenomena, has operators for sequentially combining formulas. If A and B are formulas, so are $A;B$ (*"chop"*) and A^* (*"chop-star"*). These are related to the concatenation and Kleene star operators for regular expressions. Time is modelled as in conventional discrete linear-time temporal logic using finite and infinite sequences of one or more states.

We recently introduced and investigated **2-to-1** formulas for compositional reasoning [2]. A formula A is 2-to-1 if the implication $(A;A) \supset A$ is valid, so if two portions of a system ensure such a formula's behaviour, then their sequential composition is guaranteed to as well. This work builds on our earlier compositional techniques surveyed in [3] to facilitate inference rules for combining concurrent systems sequentially and in parallel. A sample rule is later presented in Section 3. The 2-to-1 formulas are closed under conjunction and the conventional temporal operator $\Box$ ("always") which examines *suffix* subintervals. This helps modularly obtain 2-to-1 safety and liveness formulas such as the standard temporal formulas $\Box p$ ("always p") and $\Box(p \supset \Diamond q)$ ("p always leads to q"), where p and q are propositional variables. The propositional version of ITL (PITL) used here to formalise 2-to-1 formulas is decidable and has a complete axiomatisation [4], but our results are also applicable to first-order ITL (along the lines of [3]).

The versatile class of 2-to-1 formulas has useful subclasses (e.g., $*$**-to-1**: $A^* \supset A$) and other variants, also closed under conjunction and sometimes closed under $\Box$ as well. Our presentation here mainly concerns interconnections between these to elucidate the nature of such formulas and obtain new ones. We systematically and incrementally apply properties proved about some formulas to later proofs for others to avoid redundant reasoning.

2. Propositional Interval Temporal Logic

We now describe the version of (quantifier-free) PITL used here. More about PITL can be found in [4] (see also [1] and the ITL web pages [5]).

Here is a BNF syntax of PITL formulas, with p any propositional variable:

$$A ::= \mathit{true} \mid p \mid \neg A \mid A \vee A \mid \mathit{skip} \mid A;A \mid A^*. \tag{1}$$

E-mail address: benm@dmu.ac.uk.

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2013.02.005

Table 1
Some useful derived PITL operators.

$\bigcirc A \stackrel{\text{def}}{\equiv} skip; A$	Next	$more \stackrel{\text{def}}{\equiv} \bigcirc true$	Two or more states
$empty \stackrel{\text{def}}{\equiv} \neg more$	One state	$A? \stackrel{\text{def}}{\equiv} empty \wedge A$	One state with test
$inf \stackrel{\text{def}}{\equiv} true; false$	ω states	$finite \stackrel{\text{def}}{\equiv} \neg inf$	Finite interval
$\Diamond A \stackrel{\text{def}}{\equiv} finite; A$	Eventually	$\Box A \stackrel{\text{def}}{\equiv} \neg\Diamond\neg A$	Henceforth (always)
$fin\ A \stackrel{\text{def}}{\equiv} \Box(empty \supset A)$	Final state (weak)	$A^\omega \stackrel{\text{def}}{\equiv} inf \wedge (finite \wedge A)^*$	Chop-omega
$\diamondsuit A \stackrel{\text{def}}{\equiv} A; true$	Initial (prefix) subinterval	$\boxdot A \stackrel{\text{def}}{\equiv} \neg\diamondsuit\neg A$	All initial (prefix) subintervals

Boolean operators *false*, $A \wedge B$, $A \supset B$ and $A \equiv B$ are defined as usual.

Time within PITL is modelled by discrete, linear state sequences. The *set of states* Σ is the power set 2^V, where V is the set of propositional variables. Each state in Σ therefore sets every propositional variable $p, q, \ldots$ to *true* or *false*. The associated (standard) version of PITL with such state-based variables (instead of interval-based ones) is called *local PITL*. An *interval* σ is any element of $\Sigma^+ \cup \Sigma^\omega$ and has states $\sigma^0, \sigma^1, \ldots$.

The notation $\sigma \models A$, defined shortly by induction on formula A's syntax, denotes that interval σ *satisfies* A. If all intervals satisfy A, denoted $\models A$, it is *valid*. Below are the semantics of the first five PITL constructs in (1):

$\sigma \models true$ for any σ

$\sigma \models p$ iff $p \in \sigma^0$ (initially p)

$\sigma \models \neg A$ iff $\sigma \not\models A$

$\sigma \models A \vee B$ iff $\sigma \models A$ or $\sigma \models B$

$\sigma \models skip$ iff $\sigma \in \Sigma^2$ (two states).

The cases below for chop and chop-star involve *subintervals*:

- Chop: $\sigma \models A; B$ iff for some σ' and σ'', $\sigma' \models A$ and $\sigma'' \models B$

  or $\sigma \in \Sigma^\omega$ and $\sigma \models A$,

  where $\sigma' \in \Sigma^+$ is a finite *prefix* subinterval of σ (perhaps even σ itself if $\sigma \in \Sigma^+$), and σ'' is the adjacent *suffix* subinterval of σ with one shared state (i.e., the last state of σ'). Chop here is *weak* (like the weak version of the temporal operator Until) for potentially nonterminating programs which ignore B. *Strong* chop (and chop-star) is derivable.
- Chop-star: $\sigma \models A^*$ iff one of the following holds: (1) σ has only one state (i.e., $\sigma \in \Sigma$). (2) σ either itself satisfies A or can be split into a finite number of subintervals which share end-states (like chop) and all satisfy A. (3) $\sigma \in \Sigma^\omega$ and can be split into ω finite-length intervals sharing end-states (like chop) and each satisfying A.

Consider a sample 5-state interval σ with the following alternating values for the variable p: $p, \neg p, p, \neg p, p$. Here are formulas satisfied by σ: p, $skip; \neg p$, $p \wedge (true; \neg p)$ and $(p \wedge (skip; skip))^*$. For instance, $skip; \neg p$ is true since σ's prefix subinterval $\sigma^0\sigma^1$ satisfies *skip* and the adjacent suffix subinterval $\sigma^1 \ldots \sigma^4$ satisfies $\neg p$ because $p \notin \sigma^1$. The formula $(p \wedge (skip; skip))^*$ is true since σ's subintervals $\sigma^0\sigma^1\sigma^2$ and $\sigma^2\sigma^3\sigma^4$ both satisfy $p \wedge (skip; skip)$. The interval σ does not satisfy the formulas $\neg p$, $skip; p$ and $true; (\neg p \wedge \neg(true; p))$.

Table 1 shows useful derived PITL operators.

Let w and w' denote *state formulas* with no temporal operators. Let *PTL* denote the PITL subset of conventional *Propositional Linear-Time Temporal Logic* with just the (derived) operators $\bigcirc$ and $\Diamond$ in Table 1.

Here are some sample valid PITL formulas:

$$A \supset \diamondsuit A \quad skip^* \quad inf \equiv \Box more$$

$$(w \wedge A); B \equiv w \wedge (A; B) \quad A \equiv (empty; A).$$

As another example, the equivalence $(w \wedge A) \equiv (empty \wedge w); A$ is valid since for any interval σ, $\sigma \models w \wedge A$ iff σ's first state satisfies w and $\sigma \models A$.

3. 2-to-1 formulas

A PITL formula A is **2-to-1** iff $(A; A) \supset A$ is valid. For example, *true*, p, *empty* and $B?$ (for any B) are 2-to-1, as are PTL formulas $\Box p$ and $\Box(p \supset \Diamond q)$, but not *skip*. The next sample semantic inference rule uses a 2-to-1 formula A with systems *Sys* and *Sys*$'$ and pre- and post-conditions w, w' and w'':

$$\models w \wedge Sys \supset A \wedge fin\ w',$$
$$\models w' \wedge Sys' \supset A \wedge fin\ w''$$
$$\Rightarrow \models w \wedge (Sys; Sys') \supset A \wedge fin\ w''.$$

We now discuss properties of 2-to-1 formulas. This class is later used with other classes such as $*$**-to-1** formulas (i.e., $\models A^* \supset A$) for iteration in Section 4. Our systematic analysis incrementally obtains formulas in the classes. Included are many PTL formulas for which it also seems computationally feasible to check membership in the classes, but this is left for future work.

Theorem 1. *(See Moszkowski [2].)* 2*-to-*1 *formulas are closed under* $\wedge$ *and* $\Box$.

Proof. Here is a chain of valid implications for $\wedge$: $(A \wedge B); (A \wedge B) \supset (A; A) \wedge (B; B) \supset (A \wedge B)$. For $\Box$, if $\sigma \models (\Box A); (\Box A)$, then every suffix of σ satisfies $A; A$ or A and hence A so $\sigma \models \Box A$. Therefore, $\models (\Box A); (\Box A) \supset \Box A$. $\Box$

The $\Box$-closure for 2-to-1 formulas nicely generalises to the next semantic inference rule for any C and C': $\models (C'; C) \supset C \Rightarrow \models ((\Box C'); \Box C) \supset \Box C$.