Using BDDs to capture data in Runtime verification (RV) [HP18]

Per Ove Ringdal

November 29, 2019

Contents

Motivation

Syntax and semantics of QTL

QTL Example

An Efficient Algorithm Using BDDs

Summary

References

Verifying file operations

Problem: We have a program that writes data to files, and we want to verify that some property always holds.

Assume file API which yields the following events:
$open(f)$: file f was open
$write(f, d)$: data d was written to file f
$close(f)$: file f was closed

Property:
A file should be open when writing data to it.

Runtime Verification - Definition

What is Runtime Verification?

- Lightweight formal method that complements classical exhaustive verification techniques [Bar+18]
- Analyse a single execution trace of a system
- At the price of limited execution coverage, we get precise information on the runtime behavior

Runtime Verification - Analysing execution traces

We analyse the system against a property, yielding an alarm when the property is violated. [HP18]

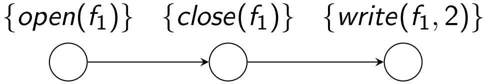
The property for the file API can be written as: "A file can only be written to if it has been opened in the past, and not closed since then."

Or in Quantified Temporal Logic (QTL), which will be explained later:
$\forall f((\exists d \; write(f,d)) \rightarrow \neg close(f) \; S \; open(f))$

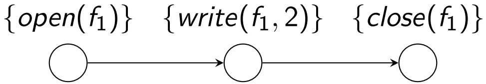
Execution trace examples

Example

not valid:



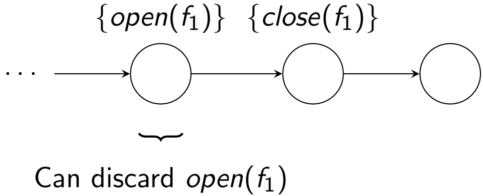
valid:



Data reclamation

When data cannot affect the rest of the execution we want to discard this data.

For instance, when a file is closed, we can forget that it was opened before that



Syntax and semantics of QTL

Syntax and semantics of QTL - Assignment

Definition

Let X be a finite set of *variables*. An *assignment* over a set of variables $W \subseteq X$ maps each variable $x \in W$ to a value from its associated domain $domain(x)$.

Syntax and semantics of QTL - Assignment

Definition

Let X be a finite set of *variables*. An *assignment* over a set of variables $W \subseteq X$ maps each variable $x \in W$ to a value from its associated domain $domain(x)$.

Example

$[x \mapsto 5, y \mapsto "abc"]$ maps x to 5 and y to 'abc'.

Syntax and semantics of QTL - Predicate names

Definition

Let T be a set of *predicate names*, where each predicate name p is associated with some domain $domain(p)$. A predicate is constructed from a predicate name and a variable or a constant of the same type. Predicates over constants are called *ground predicates*.

Syntax and semantics of QTL - Predicate names

Definition

Let T be a set of *predicate names*, where each predicate name p is associated with some domain $domain(p)$. A predicate is constructed from a predicate name and a variable or a constant of the same type. Predicates over constants are called *ground predicates*.

Example

If the predicate name p and the variable x are associated with the domain of strings, we have predicates like *p("gaga")* and *p(x)*,

Syntax and semantics of QTL - Events

Definition

An *event* is a finite set of ground predicates.

Syntax and semantics of QTL - Events

Definition

An *event* is a finite set of ground predicates.

Example

If $T = \{p, q, r\}$, then $\{p(\text{"}xyzzy\text{"}), q(3)\}$ is a possible event.

Syntax and semantics of QTL - Events

Definition

An *event* is a finite set of ground predicates.

Example

If $T = \{p, q, r\}$, then $\{p(''xyzzy''), q(3)\}$ is a possible event.

Definition

An *execution trace* $\sigma = s_1, s_2, \ldots$ is a finite sequence of events.

Syntax and semantics of QTL - Formulas

Definition

The formulas of QTL are defined by the following grammar.

$$
\begin{aligned}
\varphi ::= &\ \mathit{true} \\
&\mid p(a) \\
&\mid p(x) \\
&\mid \varphi \wedge \varphi \\
&\mid \neg\varphi \\
&\mid \varphi\ S\ \varphi \\
&\mid \ominus\ \varphi \\
&\mid \exists x\ \varphi
\end{aligned}
$$

Syntax and semantics of QTL - Formulas

Definition

The formulas of QTL are defined by the following grammar.

$\varphi ::= \mathit{true}$

$\mid p(a)$ holds when a is a constant in *domain(p)* and *p(a)* occurs in the most recent event

$\mid p(x)$

$\mid \varphi \wedge \varphi$

$\mid \neg\varphi$

$\mid \varphi \; S \; \varphi$

$\mid \ominus \; \varphi$

$\mid \exists x \; \varphi$

Syntax and semantics of QTL - Formulas

Definition

The formulas of QTL are defined by the following grammar.

$\varphi ::= \mathit{true}$

$\mid\ p(a)$	holds when a is a constant in *domain(p)* and *p(a)* occurs in the most recent event
$\mid\ p(x)$	holds with a binding of x to value a if *p(a)* occurs in the most recent event
$\mid\ \varphi \wedge \varphi$	
$\mid\ \neg\varphi$	
$\mid\ \varphi\ S\ \varphi$	
$\mid\ \ominus\ \varphi$	
$\mid\ \exists x\ \varphi$	

Syntax and semantics of QTL - Formulas

Definition

The formulas of QTL are defined by the following grammar.

$\varphi ::=$	*true*	
	$\mid p(a)$	holds when a is a constant in *domain(p)* and *p(a)* occurs in the most recent event
	$\mid p(x)$	holds with a binding of x to value a if *p(a)* occurs in the most recent event
	$\mid \varphi \wedge \varphi$	
	$\mid \neg\varphi$	
	$\mid \varphi \; S \; \varphi$	for $\varphi \; S \; \psi$, ψ held in past or now, and since then φ has been true
	$\mid \ominus \; \varphi$	
	$\mid \exists x \; \varphi$	

Syntax and semantics of QTL - Formulas

Definition

The formulas of QTL are defined by the following grammar.

$\varphi ::=$	$true$	
	$\mid p(a)$	holds when a is a constant in $domain(p)$ and $p(a)$ occurs in the most recent event
	$\mid p(x)$	holds with a binding of x to value a if $p(a)$ occurs in the most recent event
	$\mid \varphi \wedge \varphi$	
	$\mid \neg\varphi$	
	$\mid \varphi\ S\ \varphi$	for $\varphi\ S\ \psi$, ψ held in past or now, and since then φ has been true
	$\mid \ominus\ \varphi$	φ is true in the previous event
	$\mid \exists x\ \varphi$	

Syntax and semantics of QTL - Formulas derived

The following formulas can be derived from the definition:

$$\begin{aligned}
\mathit{false} &= \neg \mathit{true} \\
\varphi \ \vee \ \psi &= \neg(\neg\varphi \wedge \neg\psi) \\
\varphi \rightarrow \psi &= \neg\varphi \wedge \psi \\
P\ \varphi &= \mathit{true}\ S\ \varphi \\
H\ \varphi &= \neg P\ \neg\varphi \\
\forall x\ \varphi &= \neg\exists x\ \neg\varphi
\end{aligned}$$

Syntax and semantics of QTL - Formulas derived

The following formulas can be derived from the definition:

$$
\begin{aligned}
\mathit{false} &= \neg \mathit{true} \\
\varphi \ \vee \ \psi &= \neg(\neg\varphi \wedge \neg\psi) \\
\varphi \rightarrow \psi &= \neg\varphi \wedge \psi \\
P\ \varphi &= \mathit{true}\ S\ \varphi \qquad \varphi \text{ held in the past or now} \\
H\ \varphi &= \neg P\ \neg\varphi \\
\forall x\ \varphi &= \neg\exists x\ \neg\varphi
\end{aligned}
$$

Syntax and semantics of QTL - Formulas derived

The following formulas can be derived from the definition:

$$
\begin{aligned}
\mathit{false} &= \neg \mathit{true} \\
\varphi \ \vee \ \psi &= \neg(\neg\varphi \wedge \neg\psi) \\
\varphi \rightarrow \psi &= \neg\varphi \wedge \psi \\
P\ \varphi &= \mathit{true}\ S\ \varphi && \varphi \text{ held in the past or now} \\
H\ \varphi &= \neg P\ \neg\varphi && \varphi \text{ always true in the past and now} \\
\forall x\ \varphi &= \neg\exists x\ \neg\varphi
\end{aligned}
$$

Syntax and semantics of QTL - *free*, *hide*

Definition

Let *free*(φ) be the set of free (i.e., unquantified) variables of a subformula φ.

Syntax and semantics of QTL - *free*, *hide*

Definition

Let *free*(φ) be the set of free (i.e., unquantified) variables of a subformula φ.

Definition

Let A_1 and A_2 be sets of assignments. The intersection $A_1 \cap A_2$ is defined like a database 'join' operator. The union $A_1 \cup A_2$ is defined as the operator dual of intersection.

Syntax and semantics of QTL - *free*, *hide*

Definition

Let $free(\varphi)$ be the set of free (i.e., unquantified) variables of a subformula φ.

Definition

Let A_1 and A_2 be sets of assignments. The intersection $A_1 \cap A_2$ is defined like a database 'join' operator. The union $A_1 \cup A_2$ is defined as the operator dual of intersection.

Definition

Let Γ be a set of assignments over a set of variables W. We denote by $hide(\Gamma, x)$ the sets of assigments over $W \setminus \{x\}$, obtained from Γ by removing the assignment to x for each element of Γ.

Syntax and semantics of QTL - $\mathrm{I}[\varphi, \sigma, i]$

Definition

$A_{free(\varphi)}$ is the set of all possible assignments of values to the variables that appear free in φ.

Syntax and semantics of QTL - $\mathrm{I}[\varphi, \sigma, i]$

Definition

$A_{free(\varphi)}$ is the set of all possible assignments of values to the variables that appear free in φ.

Definition

Let $\mathrm{I}[\varphi, \sigma, i]$ be the semantic function, defined below. It returns the set of assignments that satisfy φ after the ith event of the exection σ. The empty set of assignments $\emptyset$ behaves as the Boolean constant 0 and the singleton set that contains an assignment over an empty set of variables $\{\epsilon\}$ behaves as the Boolean constant 1.

Syntax and semantics of QTL - $\mathrm{I}[\varphi, \sigma, i]$ cont.

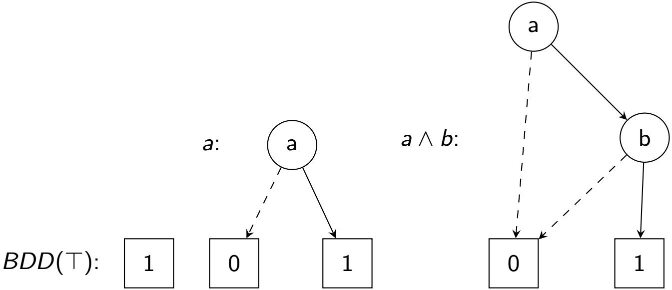
$$
\begin{aligned}
\mathrm{I}[\varphi, \sigma, 0] &= \emptyset \\
\mathrm{I}[\mathit{true}, \sigma, i] &= \{\epsilon\} \\
\mathrm{I}[p(a), \sigma, i] &= \text{if } p(a) \in \sigma[i] \text{ then } \{\epsilon\} \text{ else } \emptyset \\
\mathrm{I}[p(x), \sigma, i] &= \{[x \mapsto a] \mid p(a) \in \sigma[i]\} \\
\mathrm{I}[\varphi \wedge \psi, \sigma, i] &= \mathrm{I}[\varphi, \sigma, i] \ \cap \ \mathrm{I}[\psi, \sigma, i] \\
\mathrm{I}[\neg\varphi, \sigma, i] &= A_{\mathit{free}(\varphi)} \setminus \mathrm{I}[\varphi, \sigma, i] \\
\mathrm{I}[\varphi \ S \ \psi, \sigma, i] &= \mathrm{I}[\psi, \sigma, i] \ \cup \ (\mathrm{I}[\varphi, \sigma, i] \ \cap \ \mathrm{I}[\varphi \ S \ \psi, \sigma, i-1]) \\
\mathrm{I}[\ominus \ \varphi, \sigma, i] &= \mathrm{I}[\varphi, \sigma, i-1] \\
\mathrm{I}[\exists x \ \varphi, \sigma, i] &= \mathit{hide}(\mathrm{I}[\varphi, \sigma, i], x)
\end{aligned}
$$

QTL Example

An Efficient Algorithm Using BDDs

Boolean functions as Binary Decision Diagrams

Here Ordered Binary Decision Diagrams (OBDD) are used.
BDDs are a way of efficiently representing a boolean function ($f : 2^n \to 2$) as a directed asyclic graph.



Algorithm for monitoring QTL

1. Initially, for each subformula φ, $now(\varphi) := BDD(\bot)$
2. Observe a new event (as set of ground predicates) s as input
3. Let $pre := now$
4. Make the following updates for each subformula. If φ is a subformula of ψ then $now(\varphi)$ is updated before $now(\psi)$
   - $now(true) := BDD(\top)$
   - $now(p(a)) :=$ if $p(a) \in s$ then $BDD(\top)$ else $BDD(\bot)$
   - $now(p(x)) := build(x, V)$ where $V = \{a \mid p(a) \in s\}$
   - $now(\varphi \land \psi) := and(now(\varphi), now(\psi))$
   - $now(\neg\varphi) := not(now(\varphi))$
   - $now(\varphi\ S\ \psi) := or(now(\psi), and(now(\varphi), pre(\varphi\ S\ \psi)))$
   - $now(\ominus\ \varphi) := pre(\varphi)$
   - $now(\exists x\ \varphi) := exists(\langle x_0, \ldots, x_{k-1}\rangle, now(\varphi))$
5. Goto step 2

Summary

- First-order past time temporal logic properties (QTL)
- The properties contains data (ground predicates) over infinite domains

References I

Ezio Bartocci et al. "Introduction to Runtime Verification". In: *Lectures on Runtime Verification: Introductory and Advanced Topics*. Ed. by Ezio Bartocci and Yliès Falcone. Cham: Springer International Publishing, 2018, pp. 1–33. ISBN: 978-3-319-75632-5. DOI: 10.1007/978-3-319-75632-5_1. URL: https://doi.org/10.1007/978-3-319-75632-5_1.

Klaus Havelund and Doron Peled. "BDDs on the Run". In: *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice*. Ed. by Tiziana Margaria and Bernhard Steffen. Cham: Springer International Publishing, 2018, pp. 58–69. ISBN: 978-3-030-03427-6. URL: https://link.springer.com/chapter/10.1007%2F978-3-030-03427-6_8.