Three-Valued Asynchronous Distributed Runtime Verification

Torben Scheffel

Institute for Software Engineering and Programming Languages
University of Lübeck, Germany

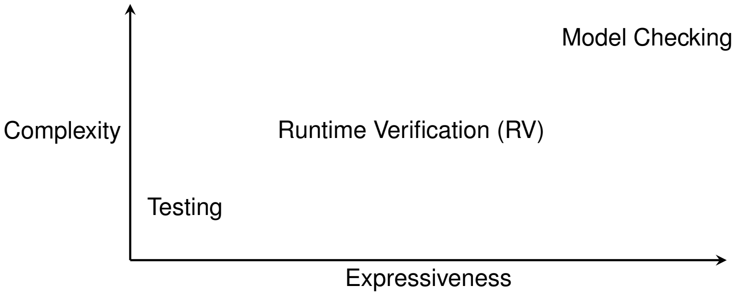
scheffel@isp.uni-luebeck.de

October 19, 2014

Table of Contents

Table of Contents

Introduction



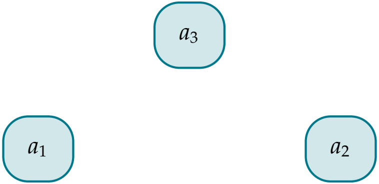
Challenges of Distributed RV in Asynchronous Systems

There are various encountered when doing RV in asynchronous distributed systems, for example:

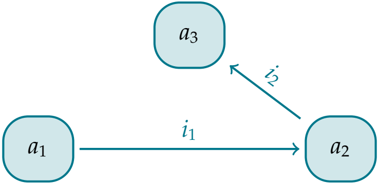
- different execution speed of agents
- inherent non-determinism in execution order
- information have to reach the monitor (communication overhead)
- one centralized or many decentralized monitors?

Table of Contents

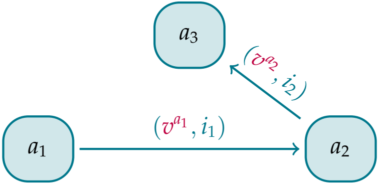
System Model



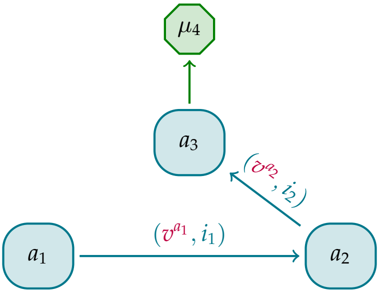
System Model



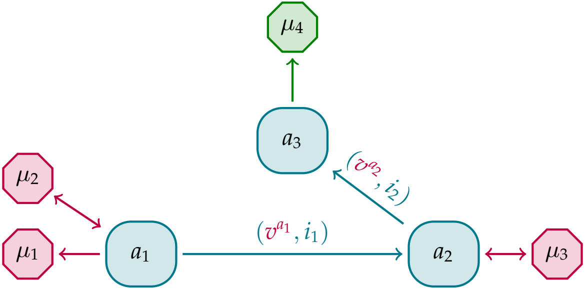
System Model



System Model



System Model



System Model

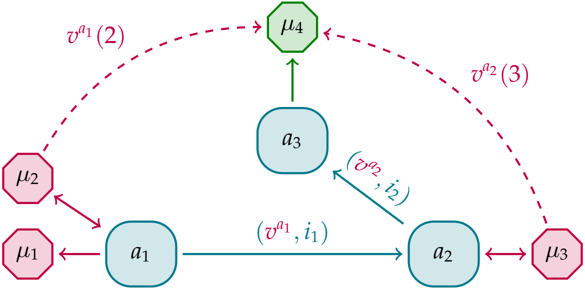


Table of Contents

Linear Temporal Logic (LTL) and Past Operators

$w = w_0 w_1 w_2 w_3 w_4 \cdots \in \Sigma^\omega$ execution trace (word)

Set of propositions and boolean operators negation ($\neg$) and or ($\vee$).

Future operators:

- Next ($\bigcirc$)
- Until ($\mathcal{U}$)

Past operators:

- Previous ($\ominus$)
- Since ($\mathcal{S}$)

Three-valued LTL over finite traces (LTL_3)

A. Bauer, M. Leucker, and C. Schallhart,

"Runtime Verification for LTL and TLTL"

$$[\![w \models \varphi]\!]_{\text{LTL}_3} = \begin{cases} \top & \text{if } \forall u \in \Sigma^\omega : wu \models_{\text{LTL}} \varphi \\ \bot & \text{if } \forall u \in \Sigma^\omega : wu \not\models_{\text{LTL}} \varphi \\ ? & \text{else} \end{cases}$$

The output of the LTL_3 semantics is only $\top$ or $\bot$ if every infinite extension of the trace is a model (not a model resp.) of the formula in LTL.

Past-Time Distributed Temporal Logic (ptDTL)

K. Sen, A. Vardhan, G. Agha, and G. Rosu,

"Efficient Decentralized Monitoring of Safety in Distributed Systems"

An Additional @-operator is used to spread properties over different agents.

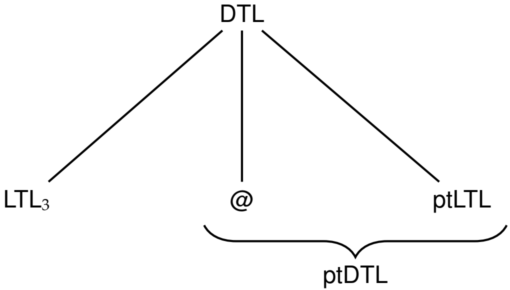
Example:

$$@_{a_1}(p\,\mathcal{S}\,@_{a_2}q)$$

$$\begin{array}{llllll} a_1: & \{p\} & \{p\} & \{p\} & \{\} & \{p\} \\ a_2: & \{\} & \{q\} & \{q\} & \{q\} & \{q\} \end{array}$$

Only safety properties monitorable with ptDTL

Distributed Temporal Logic (DTL)



DTL syntax

$$
\begin{array}{rccccccccc}
\chi ::= & @^{\mathrm{ft}}_{a_1}\varphi & \mid & @^{\mathrm{pt}}_{a_1}\psi & & & & & \\
\varphi ::= & \mathrm{true} & \mid & p & \mid & \neg\varphi & \mid & \varphi \vee \varphi & \mid \\
 & \bigcirc\varphi & \mid & \varphi\,\mathcal{U}\,\varphi & \mid & @^{\mathrm{ft}}_{a_2}\varphi & \mid & @^{\mathrm{pt}}_{a_2}\psi &
\end{array}
$$

DTL semantics

$@_a^{\mathrm{pt}}\varphi$ formulas are evaluated with ptDTL semantics.

$@_a^{\mathrm{ft}}\varphi$ formulas are evaluated similar to LTL_3 with DTL_ω replacing LTL.

DTL_ω works as follows:

- all operators besides $@_a^{\mathrm{ft}}$ and $@_a^{\mathrm{pt}}$ are evaluated as in LTL
- a subformula surrounded by $@_a^{\mathrm{pt}}$ is evaluated on agent a as in ptDTL
- a subformula surrounded by $@_a^{\mathrm{ft}}$ is evaluated on agent a as in DTL

Values from other agents are delivered using messages whose send and receiving points are marked in the runs of the agents.

DTL Advantages

The main advantages of DTL are:

- future and past operators

  $\Rightarrow$ higher succinctness

- three-valued semantics

  $\Rightarrow$ many more properties monitorable

- knowledge-vector and message symbols

  $\Rightarrow$ precise theoretical evaluation possible

Monitor Construction

Monitors for past formulas of DTL: algorithm from

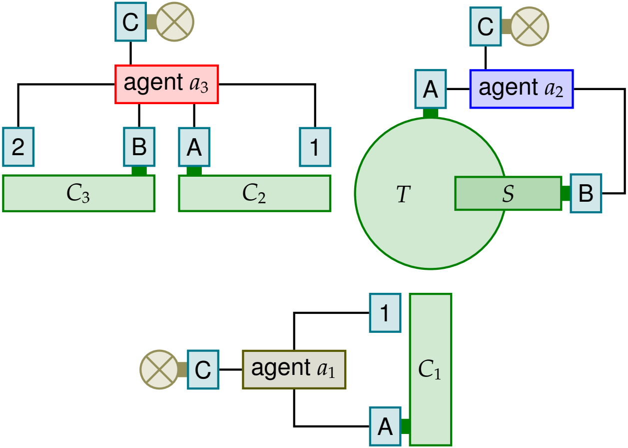
K. Havelund and G. Rosu, “Synthesizing monitors for safety properties”

Monitors for future formulas of DTL: deterministic Moore machines (DMM) constructed as follows:

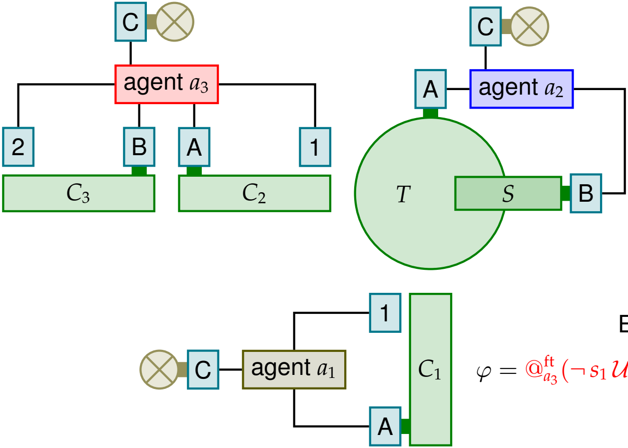
$$
\begin{tikzcd}
& \text{LTL} & \text{ABA} & \text{NBA} & \text{NFA} & \text{DFA} & \text{DMM} \\
& \varphi \arrow[r] & A_\varphi \arrow[r] & \overline{A}_\varphi \arrow[r] & \hat{A}_\varphi \arrow[r] & \tilde{A}_\varphi \arrow[dr] & \\
\varphi \arrow[ur] \arrow[dr] & & & & & & D_\varphi \\
& \neg\varphi \arrow[r] & A_{\neg\varphi} \arrow[r] & \overline{A}_{\neg\varphi} \arrow[r] & \hat{A}_{\neg\varphi} \arrow[r] & \tilde{A}_{\neg\varphi} \arrow[ur] &
\end{tikzcd}
$$

Table of Contents

Case Study



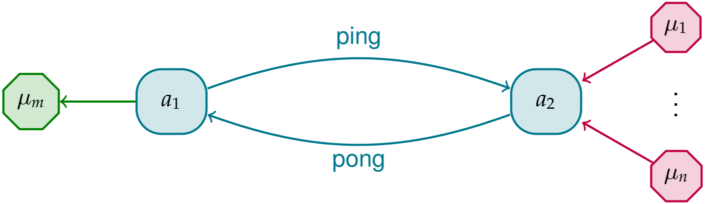
Case Study



Example:

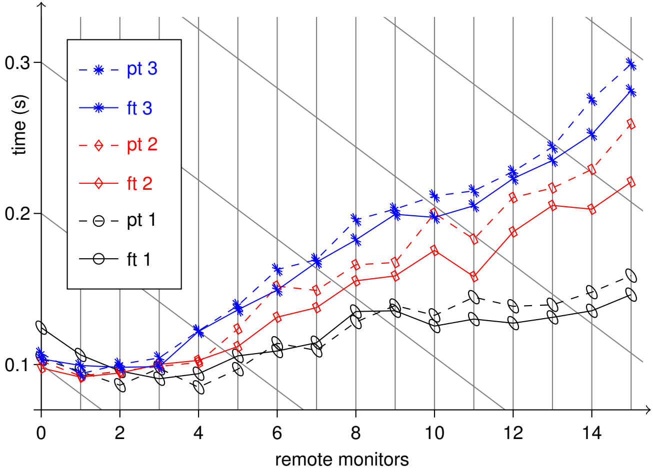
$$\varphi = @^{\text{ft}}_{a_3}(\neg s_1 \,\mathcal{U}\, @^{\text{pt}}_{a_2}(m_A \wedge \ominus @^{\text{pt}}_{a_1} \diamondminus s_1))$$

Benchmark



- Main monitor μ_m evaluates a formula of the form $@^{\text{ft}}_{a_1}(\varphi_1\,\mathcal{U}(\varphi_2\,\mathcal{U}(\ldots\mathcal{U}\,\varphi_n)))$ or $@^{\text{pt}}_{a_1}(\varphi_1\,\mathcal{S}(\varphi_2\,\mathcal{S}(\ldots\mathcal{S}\,\varphi_n)))$ for future or past case respectively.
- Every φ_i has the form $@^{\text{pt}}_{a_2}(p_{i_0}\,\mathcal{S}(p_{i_1}\,\mathcal{S}\,p_{i_2}))$ with the atomic propositions p_{i_0}, p_{i_1} and p_{i_2} and is evaluated by μ_i.

Benchmark


pt 3
ft 3
pt 2
ft 2
pt 1
ft 1
time (s)
0.3
0.2
0.1
0
2
4
6
8
10
12
14
remote monitors

Conclusion

We

- developed a system model which describes the distribution of monitoring data through messages,
- developed a new temporal logic DTL for distributed RV with a greater set of monitorable properties as ptDTL,
- programmed the transformation of DTL formulas into DMMs,
- used the created monitors for a case study to monitor a LEGO Mindstorms assembly line.